HIGHLAND TECHNOLOGY

V120/V124
PCI Express VME/VXI
Crate Controller

Technical Manual

March 24, 2023

650 Potrero Avenue San Francisco, CA 94110 www.highlandtechnology.com tel: 415 551-1700 fax: 415 551-5129

Contents

1. Introduction

The V120 is a single-width 6U VME crate controller. It provides a configurable bridge that translates reads and writes in the PCI Express memory space of a host computer to reads and writes on the VMEbus over a 1GB/s cabled PCI Express link. Ethernet and USB interfaces provide additional means of accessing VME, and a suite of additional control and diagnostic functions allow the V120 to not only provide access to a working VME system but to quickly investigate and solve a malfunctioning one.

The V124 is a C-size VXI module that includes all the features of the V120, and adds the VXI trigger and clock functions. All references to V120 features and registers apply to the V124 as well.

Features include:

- PC memory-mapped VME/VXI Crate Controller in single-wide 6U format
- Cabled PCIe connects to PC over up to 7 meter PCI Express cable
- Provides 8192 16kB VME mapping pages; individual page descriptors support address mapping/modifiers, speed options, dynamic endian swapping
- Includes concurrent Ethernet and USB ports
- Privilege, endianness, and transaction control per page, including adjusting cycle timing to tame unruly VME modules
- Selectable crate ID per V120, allowing up to 16 VME crates to be controlled from a single host without relying on potentially unstable PCI bus numbering
- VME multimastering as either a slot 1 master/arbiter or a secondary master
- VME interrupt handling through either polling or true host interrupts
- Free open-source Linux driver and application library, usable on both open and proprietary projects with no licensing requirements or fees. Full source code and extensive documentation are provided.
- Diagnostics:
    - Crate power supply, temperature sensing, and air flow monitors
    - Reads back DTACK speed, address/data, and errors for any transaction
    - VME transaction reporting with 4 programmable event monitors
    - VME utility bus reporting including SYSCLK monitoring
    - Front panel LEDs and bus signal scope monitors are provided
- Optional DMA, VME64, and VXIbus extension support
- No proprietary VME chips: designed for long-term availability
- Clean, simple architecture: no jumpers and no setup needed for basic VME I/O

2. Specifications: Model V120 / V124

2.1. General Specifications

FUNCTION	VME/VXI crate controller
DEVICE TYPE	VME master: A32: A24: A16: D08: D16: D32 VME system controller: BTO(64), IH(1-7) Optional VME64
OPERATING TEMPERATURE	0 to 60°C; extended MIL/COTS ranges available
POWER	Standard VME supplies: + 5 V, 2 A max +12 V, 50 mA max -12 V, 50 mA max
CONNECTORS	VME64 P1/P2 Four front-panel SMB test connectors One front-panel SMB CLOCK connector (V124 only) One PCI Express 4 lane connector One type B USB connector One RJ45 100 MHz Ethernet connector
INDICATORS	LEDs indicate VME access, CPU activity, error conditions; additional LED is user programmable 7-segment display indicates unit number
PACKAGING	V120 : 6U single-wide VME module V124: 6U C-size VXI module
CONFORMANCE	ANSI/VITA 1-1994 (R2002) VMEbus spec VXI-1 4.0 PCI Express Base Specification 1.1, Gen1 PCI Express External Cabling Specification 1.0

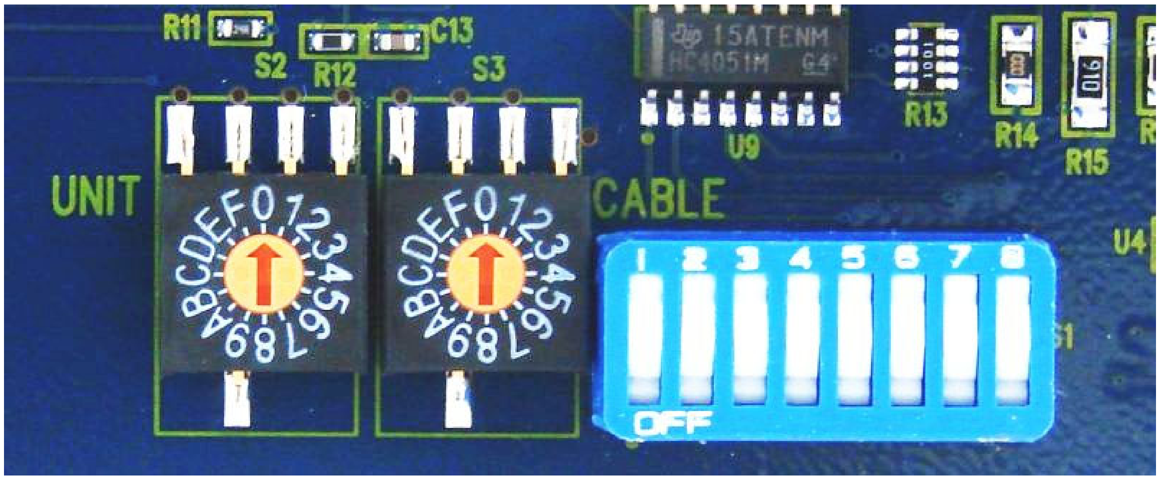
2.2. Electrical Specifications

W, X, Y, Z INPUT	V_H 2.2V – 3.6V V_L -0.3V – 0.6V
W, X, Y, Z OUTPUT	V_H 3.3V typ (100 µA) 2.0V min (16 mA) V_L 0.0V typ (100 µA) 1.1V max (16 mA) Output impedance 50Ω typ
CLK INPUT (V124)	Sine 0.5V RMS min Square 1.2V P-P min Limits -3V – 7V Impedance 50Ω typ when termination enabled 845Ω AC typ when termination disabled
CLK OUTPUT (V124)	V_H 3.3V typ (100 µA) 2.0V min (16 mA) V_L 0.0V typ (100 µA) 1.1V max (16 mA) Output impedance 50Ω typ

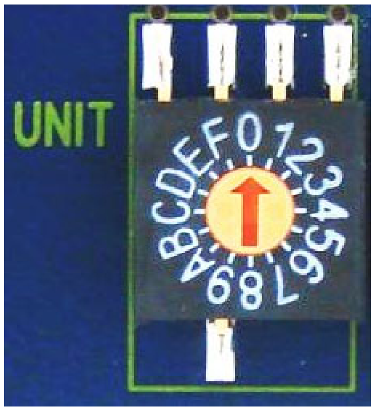
3. Connectors and Installation

3.1. DIP Switches

An internal set of DIP switches is provided to establish startup settings.



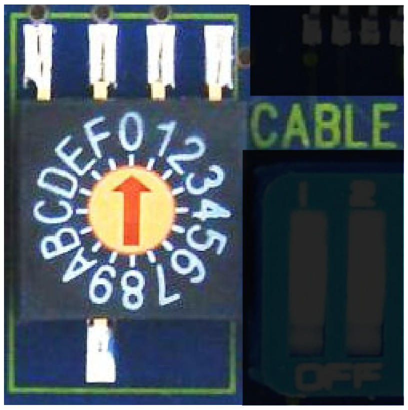
3.1.1. *Unit Number (UNIT Rotary Hex)*



The UNIT rotary switch declares a unit number from 0 to 15 (0xF). The unit number identifies crates in multi-crate installations. Unit number is displayed on the front-panel 7-segment display and is readable with other switch settings in the **DIPS** register.

When using the Highland Technology provided Linux drivers, the unit number dictates the location of the V120 devices, which will be in the form of `/dev/v120_cX` and `/dev/v120_vX`, where X is the decimal UNIT code 0-15.

3.1.2. *PCIe Cable Length (CABLE Rotary Hex)*



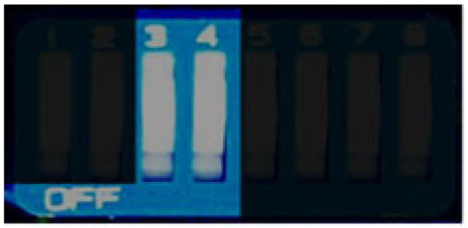
The CABLE rotary switch sets the equalization options for the PCI Express cable drivers/receivers by informing the V120 of the cable length. Positions 1 through 7 select cable lengths of 1 through 7 meters. Positions 0 and 8-15 are reserved for factory testing, and should not be selected by users as unexpected results may occur.

3.1.3. *Ethernet IP Address (1 and 2)*

Sections 1 and 2 of the blue dipswitch determine how the Ethernet IP address is set at device power-on. Regardless of the setting of this switch pair, the Ethernet address can be changed using the IP command over the Ethernet or USB interfaces.

sw1	**sw2**	**IP Address**	
OFF	OFF	Static IP	Default is 192.168.254.183, reprogrammable with IP command
OFF	ON	192.168.254.(100+U)	U = UNIT switch, 0-15
ON	OFF	192.168.254.(1+U)	U = UNIT switch, 0-15
ON	ON	DHCP	Defaults to the static IP setting if no DHCP server can be found.

3.1.4. *First Slot (3 and 4)*

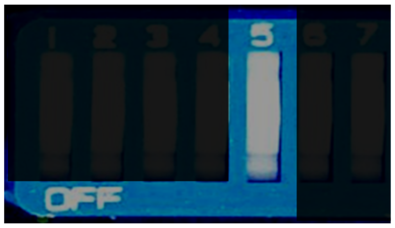


Sections 3 and 4 of the blue dipswitch determine whether the crate controller is the primary system controller/arbiter (slot 1 in VME or slot 0 in VXI) or a secondary controller.

The VXI-4 specification makes an express recommendation against the use of FSD on VXI modules. This is due to concerns that the module in question may unwittingly be inserted in a VXI crate that is several VXI-VXI repeaters down from the primary master, and may therefore fail to correctly detect the upstream master due to the excessive propagation delay of this system. FSD does work correctly in individual VXI crates without repeaters. Highland Technology, therefore, leaves to the user's discretion whether FSD is appropriate for their situation.

sw3	sw4	Function
OFF	OFF	Primary/arbiter
OFF	ON	VME BG3* first slot detect (FSD)
ON	OFF	Secondary

3.1.5. *Firmware Fallback (5)*



Section 5 of the blue dipswitch determines whether or not the crate controller will boot into upgraded firmware.

sw5	Function
OFF	Boot using upgrade firmware if present
ON	Only boot factory firmware

Controllers are shipped with CABLE = 2, UNIT = 0, switches 1-2 on, and switches 3-8 off, selecting the module as a system controller using DHCP that will boot upgraded firmware if present.

Switch settings are read only at power-up.

3.2. Installation

The controller is usually installed in the leftmost (arbiter) slot of a VME or VXI crate. Initial units do not have non-arbiter (secondary master) capability.

Always remove crate power when inserting or removing any VME module. Secure the front-panel mounting screws before applying crate power.

3.3. PCI Express Cable

A 4-lane PCI Express cable is used to connect the host PC to the V120. This may be hot-plugged if supported by the OS and BIOS of the system.

3.4. USB

A type B USB connector is provided on the front panel. It enumerates as a 115.2 kbaud serial port and uses the common FTDI USB/serial converter chip.

3.5. Ethernet

A standard shielded-shell RJ45 connector on the front panel provides 10/100 Ethernet access. The Ethernet interface autonegotiates between 10BaseT and 100BaseT, and supports Auto-MDIX for automatic detection of point-to-point links without requiring a crossover cable.

3.6. Clock Connector

A front-panel CLOCK SMB connector is provided on the V124 only. This connector can be programmed to output the V124's internal 10 MHz reference clock (which is the frequency base for the VXI 10 MHz CLK10 and 100 MHz LCLK), or to accept an external 10 MHz signal which will be used as the frequency base. See sections 5.2.22 and 5.2.23 for details.

3.7. Test Connectors

Four SMB test connectors are provided, named W, X, Y, and Z. They default to active-high TTL outputs as follows:

Connector	Function
W	VMEbus address strobe (AS)
X	Logical OR of VMEbus data strobe (DS0/DS1)
Y	VMEbus data transfer acknowledge (DTACK)
Z	PCIe activity indication

The functions of each test connector can be changed via control registers. See sections 5.2.24 and 5.2.25 for details.

3.8. Front-Panel Indicators

There are four front-panel LED indicators.

The blue VME LED flashes whenever the module accesses the VME bus. Intensity is a rough indication of traffic density.

The green CPU LED flashes about once a second to indicate CPU activity

The red ERR LED will flash to indicate errors:

Pattern	Description
Solid ON	µP startup failure
Long blinks	Fatal startup error
Two blinks	VME power supply error
Four blinks	FPGA error

The range USR LED displays a user-defined blink pattern. See sections 5.2.12, 8.4 and 9.

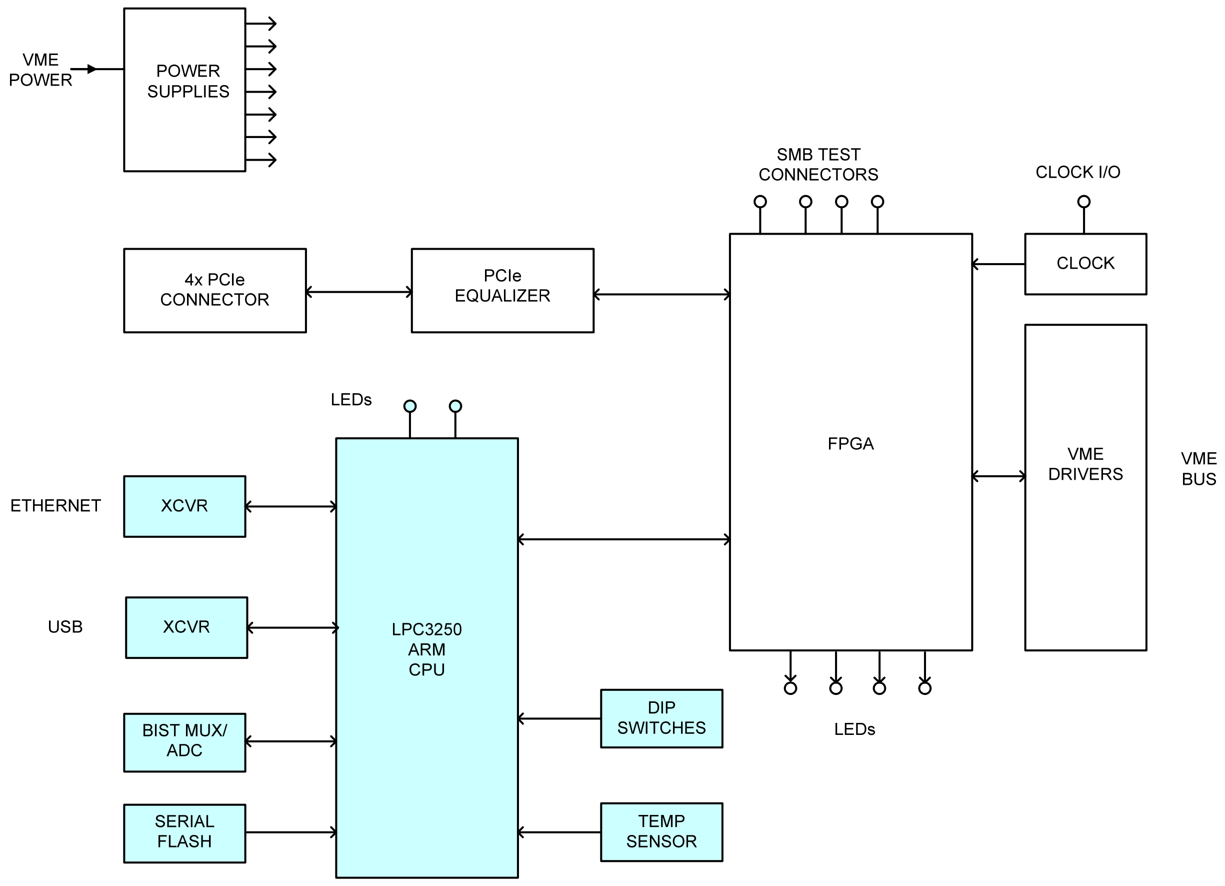
There is an additional LED on the PCB surface which illuminates green when the FPGA is properly configured.

The blue 7-segment display reports the crate unit number in hex. The decimal point of this display flashes to indicate PCI Express activity.

4. Architecture and Registers

4.1. Overview and Block Diagram

The block diagram of the V120 is as follows:



4.2. PCI Configuration Space

The crate controller provides a standard 4kB PCI Express configuration space, of which the first 64 bytes are the standard PCI Type 0 device header.

Byte 3	Byte 2	Byte 1	Byte 0	Dword Offset
Device ID		Vendor ID (0x1C32)		0
Status		Command		1
Class Code			Revision ID	2
BIST	Header Type	Latency Timer	Cache Line Size	3
BAR0				4
BAR1				5
BAR2				6
BAR3 (Unused)				7
BAR4 (Unused)				8
BAR5 (Unused)				9
Cardbus CIS Pointer (Unused)				10
Subsystem ID (0x0000)		Subsystem Vendor ID (0x0000)		12
Expansion ROM Base Address (Unused)				12
▨			Capabilities Pointer	13
▨				14
Max_Lat (0x00)	Min_Gnt (0x00)	Interrupt Pin	Interrupt Line	15

Device ID is 22120 decimal (0x5668) for both the V120 and V124. Vendor ID is 0x1C32, Highland's registered PCI vendor number. Class code is 0x11, Data Acquisition Device, subclass 0x80.

Revision ID is 0x03[1]. If any changes are made that would affect drivers, the ID will be revised. Only BAR0, BAR1, and BAR2are used.

[1] Revision IDs prior to 0x03 used an older register layout. In this layout, BAR0 is 128 MB and contains both the control space (the first 128 kB) and the VME mapping region (everything past this,

The BAR0-2 pointers in the configuration space are manipulated by the host computer's BIOS at initialization time to determine the memory block size and start addresses of the two V120 memory blocks. BAR0 is a 128 kB space used for interacting with the V120 itself. BAR1 is a 128 MB space which is mapped directly into VME by the page descriptor registers located in BAR0. BAR2 is a 128 byte space used for interacting with the V120 DMA controller. This is in a separate BAR than the normal control registers so that drivers can more easily provide access control that allows user-level access into the safe BAR0 without the security concerns inherent in the DMA controller.

4.3. V120 Control Space (BAR0)

After BIOS initialization, BAR0 will point to a 128 kB address block. The first half of the space (byte offsets 0x00000-0x10000) contains the page descriptor registers, used to control the mapping of the BAR1 space into VME. The second half (byte offsets 0x10000-0x20000) points to the V120 control registers.

4.3.1. *Page Descriptors and VME Pages*

The first 64 kilobyte region of BAR0 memory space is the page descriptor table. It contains 8,192 descriptors, each 64 bits long. Each descriptor defines the starting VMEbus address and attributes of its corresponding VME data page. A page descriptor is laid out as follows:

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
ADDR															

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDR															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR		▨		SP	E		▨	S		AM					

The AM bits are the VME address modifier field used to access VME through the associated page. For example, 0x2D specifies A16 supervisory access. For the full list of address modifiers, consult the VME specification[2].

The S bits specify the approximate VME cycle speed. The actual speed is a function of the response time of the slave module being addressed, but using speeds slower than the VME max adds additional delay to the VME cycle, above what is necessary to meet

starting with page 8). This layout was used in firmware revision A, which should be upgraded. Note that the control region is still in the same place (first 128 kB of BAR0), allowing upgrades over PCI Express to work with either layout.

[2] ANSI/VITA 1-1994 (R2002), Table 2-3, pp. 21-22

the specification. Additionally, the two slower speeds disable pipelining, also known as “address rot” and “data rot”. A compliant VME slave in a compliant VME backplane should support the maximum speed. The lower speeds exist, however, to allow access even to slaves whose VME implementation may not be 100% complaint.

S	Speed	DTACK timeout	Timeout	Pipelining
0	1 µs	100 µs	≥ 10 µs	No
1	500 ns	100 µs	≥ 30 µs	No
2	200 ns	50 µs	≥ 30 µs	Yes
3	VME max	10 µs	≥ 60 µs	Yes

The E bits specify one of four available endian translation modes. See section 7.1 for a complete discussion of the complex topic of endianness.

E	Mode	Description
0	AUTO	Swapping based on access size.
1	BYTE	Assumes all data is natively bytes.
2	WORD	Assumes all data is natively 16-bit words.
3	LONG	Assumes all data is natively 32-bit longs.

The SP (split) bit indicates that 32-bit longword transfers should be split into two 16-bit VME cycles. SP essentially declares that VME modules accessed by this page do not have P2 connectors and cannot access the D16...D31 VME data lines. If SP is set and a 32-bit read or write is performed, two VME cycles will be executed but only one PCI Express transaction will take place, increasing access speed. Split longword accesses must be longword aligned.

The ADDR bits specify the upper bits (14 and above) of the VMEbus address of the start of the page. Only the bits used in by the address mode specified in the AM field are actually valid for the transaction, an A16 transfer uses only bits 15-14, an A24 transfer uses only bits 23-14, etc.

4.3.2. ***Control Region***

The V120 control region begins at 0x10000 from the BAR0 base address. It controls global crate controller functions. All registers are 32 bits, and should only be accessed as such, avoiding any endian considerations in the control region. Users should not write to RO (read-only) registers, which are periodically refreshed by the microprocessor.

The control register space is further subdivided into peripherals representing functional regions.

Name	BAR0 Offset	Description
PAGEDESC	0x00000	PAGEDESC holds the 8,192 page descriptor registers discussed in 4.3.1.
CTLREG	0x10000	CTLREG holds assorted registers related to module function and identification.
MONITOR	0x14000	MONITOR provides 4 VME monitor blocks. Each block is individually programmable for the types of events it captures.
VXI_CTL	0x14200	VXI_CTL controls VXI clocking and triggering. While these registers are present on both the V120 and the V124, they serve no purpose on the V120 and should be ignored.
IRQHNDL	0x14400	IRQHNDL handles VME interrupt management. The PCIe IRQ flag is asserted whenever any interrupt-enabled IRQ goes high, and cleared when the host writes to PCIIRQ.
PCIE_MON	0x14800	PCIE_MON provides monitoring of the PCIe interface between the host system and the V120. This is necessary because the PCIe interface on most PCs is poorly characterized; this monitor allows for some inspection into what transfers are actually being requested.

The register map is below. Offsets are byte addresses relative to the BAR0 base address. Register numbers are calculated with respect to the peripheral base.

REG Name	REG#	Offset	R/W	Function
		CTLREG (0x10000)		
VXI MFR	0	0x10000	RO	VXI manufacturer ID
MODTYPE	1	0x10004	RO	Module type
MODREV	2	0x10008	RO	Module revision
SERIAL	3	0x1000C	RO	Unit serial number
DASH	4	0x10010	RO	Module dash (version) number
ROM ID	8	0x10020	RO	Firmware ID
ROM REV	9	0x10024	RO	Firmware revision

REG Name	REG#	Offset	R/W	Function
STAMP	10	0x10028	RO	Firmware build stamp
STATUS	16	0x10040	RO	Crate controller summary status
MCOUNT	17	0x10044	RO	Microprocessor IRQ update counter
UPTIME	18	0x10048	RO	Uptime, seconds
ULED	19	0x1004C	RW	User LED control
DIPS	20	0x10050	RO	Switch settings
MACRO	22	0x10058	RW	Macro command
MP0	23	0x1005C	RW	Macro parameters
MP1	24	0x10060	RW	…
MP2	25	0x10064	RW	…
MP3	26	0x10068	RW	…
VME ACC	32	0x10080	RO	Last VME access timing
VME WC	33	0x10084	RO	VME write counter
VME RC	34	0x10088	RO	VME read counter
UTILITY	35	0x1008C	RW	VME bus control
REQUESTER	36	0x10090	RW	VME bus requester control
ARBSTATE	37	0x10094	RO	VME bus arbitration status
CLOCKCTL	49	0x100C4	RW	10 MHz reference clock control[α]
CLOCKSTA	50	0x100C8	RO	10 MHz reference clock status[α]
FPCTL	51	0x100CC	RW	Front panel monitor control
FPMON	52	0x100D0	RO	Front panel monitor status

REG Name	REG#	Offset	R/W	Function
MACLO	60	0x100F0	RO	Reports Ethernet MAC address
MACHI	61	0x100F4	RO	Reports Ethernet MAC address
IP	62	0x100F8	RO	Reports IP address
SLOT	65	0x10104	RW	System controller information
PFLAGS	80	0x10140		Power and temperature flags
TEMP	81	0x10144		PCB temperature
P5V	83	0x1014C		Crate +5 voltage and noise
P12V	84	0x10150		Crate +12 voltage and noise
N12V	85	0x10154		Crate -12 voltage and noise
P24V	86	0x10158		Crate +24 voltage and noise [α β]
N24V	87	0x1015C		Crate -24 voltage and noise [α β]
P3.3V	88	0x10160		Crate +3.3 voltage and noise [γ]
N2V	89	0x10164		Crate -2 voltage and noise [α β]
N5.2V	90	0x10168		Crate -5.2 voltage and noise [α β]
P1.2X	91	0x1016C		Internal +1.2 voltage and noise
P2.5X	92	0x10170		Internal +2.5 voltage and noise
P3.3X	93	0x10174		Internal +3.3 voltage and noise
RAM	128	0x10200		Scratchpad RAM, 128 bytes
...	...			...

REG Name	REG#	Offset	R/W	Function
...	159	0x1027C		...
BUFFER	256	0x10400		Macro data buffer, 1024 bytes
...	...			...
...	511	0x107FC		...
		MONITOR (0x14000)		
MON_CTL0	0	0x14000	RW	Bus monitor 0 control register
MON_TRANS0	2	0x14008	RO	Monitor 0 latched request
MON_RESP0	3	0x1400C	RO	Monitor 0 latched response
ADDR_LO0	4	0x14010	RO	Monitor 0 latched VME address
ADDR_HI0	5	0x14014	RO	
DATA_LO0	6	0x14018	RO	Monitor 0 latched VME data
DATA_HI0	7	0x1401C	RO	
MON_CTL1	16	0x14040	RW	Bus monitor 1 control register
MON_TRANS1	18	0x14048	RO	Monitor 1 latched request
MON_RESP1	19	0x1404C	RO	Monitor 1 latched response
ADDR_LO1	20	0x14050	RO	Monitor 1 latched VME address
ADDR_HI1	21	0x14054	RO	
DATA_LO1	22	0x14058	RO	Monitor 1 latched VME data
DATA_HI1	23	0x1405C	RO	
MON_CTL2	24	0x14080	RW	Bus monitor 2 control register
MON_TRANS2	26	0x14088	RO	Monitor 2 latched request

REG Name	REG#	Offset	R/W	Function
MON_RESP2	27	0x1408C	RO	Monitor 2 latched response
ADDR_LO2	28	0x14090	RO	Monitor 2 latched VME address
ADDR_HI2	29	0x14094	RO	
DATA_LO2	30	0x14098	RO	Monitor 2 latched VME data
DATA_HI2	31	0x1409C	RO	
MON_CTL3	48	0x140C0	RW	Bus monitor 3 control register
MON_TRANS3	50	0x140C8	RO	Monitor 3 latched request
MON_RESP3	51	0x140CC	RO	Monitor 3 latched response
ADDR_LO3	52	0x140D0	RO	Monitor 3 latched VME address
ADDR_HI3	53	0x140D4	RO	
DATA_LO3	54	0x140D8	RO	Monitor 3 latched VME data
DATA_HI3	55	0x140DC	RO	
		VXI_CTL (0x14200) [α]		
TTLTRG_DAT0	0	0x14200	RW	Data for the TTLTRG0 line [α]
TTLTRG_DAT1	1	0x14204	RW	Data for the TTLTRG1 line [α]
TTLTRG_DAT2	2	0x14208	RW	Data for the TTLTRG2 line [α]
TTLTRG_DAT3	3	0x1420C	RW	Data for the TTLTRG3 line [α]
TTLTRG_DAT4	4	0x14210	RW	Data for the TTLTRG4 line [α]
TTLTRG_DAT5	5	0x14214	RW	Data for the TTLTRG5 line [α]
TTLTRG_DAT6	6	0x14218	RW	Data for the TTLTRG6 line [α]
TTLTRG_DAT7	7	0x1421C	RW	Data for the TTLTRG7 line [α]
TTLTRG_CTL	8	0x14220	RW	Configure the TTLTRG lines [α]
ECLTRG_CTL	9	0x14224	RW	Configure the ECLTRG lines [α]

REG Name	REG#	Offset	R/W	Function
ECLTRG_DAT0	10	0x14228	RW	Data for the ECLTRG0 line $^{\alpha}$
ECLTRG_DAT1	11	0x1422C	RW	Data for the ECLTRG1 line $^{\alpha}$
		IRQHNDL (0x14400)		
IRQSTATUS	0	0x14400	RO	VME interrupt line status
IRQEN	1	0x14404	RW	Control PCIe interrupts
IACKCFG	2	0x14408	RW	Configure IACK_VECTOR speed
PCIIRQ	3	0x1440C	RW	Control PCIe IRQ flag
IACK_VECTOR0	8	0x14420	RW	Unused
IACK_VECTOR1	9	0x14424	RO	Read IACK vector 1
IACK_VECTOR2	10	0x14428	RO	Read IACK vector 2
IACK_VECTOR3	11	0x1442C	RO	Read IACK vector 3
IACK_VECTOR4	12	0x14430	RO	Read IACK vector 4
IACK_VECTOR5	13	0x14434	RO	Read IACK vector 5
IACK_VECTOR6	14	0x14438	RO	Read IACK vector 6
IACK_VECTOR7	15	0x1443C	RO	Read IACK vector 7
		PCIE_MON (0x14800)		
PM_STATUS0	0	0x14800	RO	PCIe monitor 0 status
PM_ADDR0	1	0x14804	RO	PCIe monitor 0 address
PM_DL0	2	0x14808	RO	PCIe monitor 0 data low
PM_DH0	3	0x1480C	RO	PCIe monitor 0 data hi
PM_STATUS1	4	0x14810	RO	PCIe monitor 1 status
PM_ADDR1	5	0x14814	RO	PCIe monitor 1 address

REG Name	REG#	Offset	R/W	Function
PM_DL1	6	0x14818	RO	PCIe monitor 1 data low
PM_DH1	7	0x1481C	RO	PCIe monitor 1 data hi
…				
PM_STATUS127	508	0x14810	RO	PCIe monitor 127 status
PM_ADDR127	509	0x14814	RO	PCIe monitor 127 address
PM_DL127	510	0x14818	RO	PCIe monitor 127 data low
PM_DH127	511	0x1481C	RO	PCIe monitor 127 data hi

[α] V124 only
[β] VXI crates only
[γ] VME64 or VXI4 crates only

4.4. VME Data Pages (BAR1)

The 128 MB BAR1 space maps transactions through to the VME bus. Mapping between the PCIe address space and VME combines the bottom 14 bits of the local (PCIe) address with the upper 50 address bits from the page descriptor list as follows:

```
offset = PCIe_addr - BAR1_base_addr
page_idx = offset / 0x4000
vme_base_addr = page_descriptor[page_idx].A
page_offset = offset & 0x3FFF
vme_addr = vme_base_addr + page_offset
```

PAGEDESC #	BAR1 Start Offset	BAR1 End Offset
0	0x0000_0000	0x0000_3FFF
1	0x0000_4000	0x0000_7FFF
2	0x0000_8000	0x0000_BFFF
	…	
8189	0x07FF_4000	0x07FF_7FFF
8190	0x07FF_8000	0x07FF_BFFF
8191	0x07FF_C000	0x07FF_FFFF

So if PAGEDESC[3] = 0x0000_0000_0012_40F9, which is ADDR=0x124000, SP=0, E=AUTO, S=MAX, AM=0x39 (A24 user data access), then the host memory at BAR1 + 0xD040 (which is an offset of 0x1040 into page 3 at 0xC000) maps onto VME A24 address 0x1241040.

Multiple V120 pages can map onto the same VME address range without any problems. The most common use case for this is to map regions with different address modifiers, since the VME address spaces are non-overlapping, but this can also be used to differentiate D16/D32, access speeds, endianness, etc. The Highland provided Linux library uses this fact, along with the exceedingly large number of page mappings available on the V120, to simplify the allocation of pages onto VME modules by allocating each module's pages entirely to that module. So a card at A24 0xC000 will go into page 0, and the next card at A24 0xC200 will go into page 1 even though they could share the same page.

4.5. Default Power-Up States

At power-up, the default state of the crate controller is:

All control registers are cleared, and then read-only registers are loaded.

Page descriptors PDR8 through PDR11 are loaded such as to map the A16 address space through VME data pages 8...11. Supervisor mode is assumed, access speed 200ns, endian mode 0. That gives:

PDR8	0x 0000 0000 0000 00AD
PDR9	0x 0000 0000 0000 40AD
PDR10	0x 0000 0000 0000 80AD
PDR11	0x 0000 0000 0000 C0AD

Page descriptors 12...1,035 are loaded such as to map the A24 space through VME pages 12...1,035.

PDR12	0x 0000 0000 0000 00BD
PDR13	0x 0000 0000 0000 40BD
PDR14	0x 0000 0000 0000 80BD
PDR15	0x 0000 0000 0000 C0BD

and so on.

Page descriptors 1,036...8,191 are loaded such as to map the first 117 Mbytes of the A32 space through VME pages 1,036...8,191.

PDR1036	0x 0000 0000 0000 008D
PDR1037	0x 0000 0000 0000 408D
PDR1038	0x 0000 0000 0000 808D
PDR1039	0x 0000 0000 0000 C08D

and so on.

The boot and code startup sequence is described in section 9.

4.6. Macros

The 32-bit **MACRO** register allows any port (PCIe, USB, or Ethernet) to request commands to be executed by the microprocessor.

To execute a macro command:

1. Verify that bits 7...0 of **MACRO** are zero. Nonzero indicates that the macro processor is busy
2. Write any required parameters to the **MP0** ... **MP3** registers
3. Write a macro code to the **MACRO** register
4. Wait for **MACRO** bits 7...0 to clear, indicating that the operation is done. If bit 15 is set, there was a parameter or execution error.
5. Read any delivered parameters

Macro command codes are...

Value	Code	Description
0x00000080	NO-OP	If macro code 0x00000080 is written to the MACRO register, a dummy macro is executed and **MACRO** is cleared.
0x00000081	FLASH REPORT	See section 9.2 for details of the Flash macros.
0x00000082	RESET	If macro code 0x00000082 is written to the MACRO register, the module will reboot. The factory or upgrade code will be installed and run.
0x00000083	ERASE FLASH	See section 9.2 for details of the Flash macros.
0x00000084	WRITE FLASH	
0x00000086	FLASH UNLOCK	

4.7. Interrupts

4.7.1. *Basics of VME Interrupts*

VME modules can request interrupts by asserting (low) the open-drain IRQ lines (1-7). When a module asserts an interrupt, the interrupt responder module must perform an **IACK** *Interrupt Acknowledge* cycle to fetch the interrupt vector from the module; this is a (generally user-programmed) value that identifies the interrupting card, since multiple VME modules can interrupt on the same interrupt line.

From here, VME modules can be divided into two types. A **ROAK** module will *Release on AcKnowledge*, and thus let go of the IRQ line. A **RORA** module will *Release on Register Access*, meaning that a further action must be taken by the VME master to complete the interrupt cycle.

The V120 acts as both a bus master and an interrupt responder. However, since decisions about priority and how to handle RORA modules are policy questions that on a module by module and system by system basis, the generation of IACK cycles and any RORA completions are left to the user-space program.

To generate an IACK cycle, the host must perform a 32-bit read of one of the IACK_VECTOR[] registers (see section 5.5.5). A read to IACK_VECTOR[n] generates an IACK cycle with address n, which prompts the VME card interrupting on IRQn to provide a vector address in response. This vector address is returned as the result of reading the register, and it is then up to the user software to take appropriate action based on the vector returned.

As will be discussed further, the user has the option to either poll on the interrupt status or to configure individual IRQ lines to generate true PCIe MSI interrupts.

4.7.2. *IRQ Polling*

An application can periodically check the IRQSTATUS register (see section 5.5.1), which provides the current status of the IRQ lines. This is less efficient than using true interrupts, but easier and places fewer rules on how interrupts must be handled, as well as avoiding the need for communication between asynchronous tasks.

Code to do so (using the Highland provided V120 Linux library) would look similar to the following.

```
void process_interrupts(V120_IRQ *pIrq)
{
    int irq;
    do {
        uint32_t status = pIrq->irqstatus;
        for (int irq=7; irq>0; irq--) {
            if (status & (1 << irq)) {
                uint32_t vector = pIrq->iack_vector[irq];
                take_some_action(irq, vector);
                break;
            }
        }
    } while (irq>0);
}
```

4.7.3. *Host Interrupts*

More complicated but faster than polling is to use true interrupts. By setting bits 7-1 in the IRQEN register (see section 5.5.2), the V120 will be configured to generate MSI interrupts on the host system. Generally, these interrupts will need to be handled by a driver, such as the one provided by Highland for Linux systems.

The V120 signals an interrupt by raising the PCIIRQ flag (see section 5.5.4), generating an interrupt on the host system. The first thing the host must do is clear the PCIIRQ flag to acknowledge receipt of the interrupt. Then the host must service all interrupts selected in the IRQEN register (or simply all interrupts) until all of their lines are clear in the IRQSTATUS register. Clearing all IRQSTATUS lines resets the interrupt logic and allows it to generate the next interrupt.

4.7.4. *v120irqd*

To simplify interrupt handling, especially with multiple processes, Highland provides v120irqd, an application that is part of Highland's free Linux software suite. Full documentation about v120irqd and the API interfaces is provided in the V120 Linux Programmer's Manual, included with the software.

4.8. DMA Control Space (BAR2)

The 128 byte BAR2 space manages the V120 DMA engine, which can pull blocks of data between the host PC and VME without the processor having to move each word. This also allows the data to go into larger PCI Express packets than single word reads, greatly reducing overhead and speeding throughput.

The DMA engine maps VME orthogonally to the page descriptor mechanism in BAR0; giving each requested transfer its own (similarly structured) descriptor instead. This means that, for instance, a program that wanted to fetch 1 GB of waveform data from A32 space would not dynamically swap VME pages through BAR0 in order to access the entire space. Instead, the program would just construct a request for the data and fetch it by executing the single request.

DMA transfers have substantially higher throughput than individual I/O transfers. The exact numbers are highly situational, but as a rough number, individual D32 reads through BAR1 can move about 2.7 MB/s. Using DMA instead, the throughput jumps to 12.5 MB/s.

4.8.1. *DMA Theory*

To perform a DMA transfer, the host must create a scatter/gather list in local memory. This is a linked list of DMA descriptors, each describing a copy between a contiguous region of host memory and a contiguous region of VME space. The last descriptor in the list will point to address 0 to indicate the termination of the list.

Having built the descriptor chain in memory, the host will write the address of the first descriptor into NEXTDESC (section 5.7.3, page 55) and then set the GO bit in the DMA CONTROL register (section 5.7.1, page 54). This will cause the DMA engine to begin by fetching the first descriptor, then transferring both the described data and the additional descriptors in the chain as needed. While the transfer proceeds, the NEXTDESC, STATUS, and VMEADDR registers will be continually updated .

Successful completion of the final descriptor will terminate the DMA transfer. So will VME errors, or any problems with the DMA descriptors. In any case, the DMA controller will raise an interrupt. The software should check the DMA STATUS register (section 5.7.2, page 54) to determine the cause of termination.

4.8.2. *DMA Descriptors*

This descriptor (as defined in driver/v120_dma.c) is

```
struct v120_dma_hwdesc_t {
        u32 ctl;
        u32 len;
        u32 vme_addr[2];
        u32 bus_addr[2];
        u32 next_addr[2];
        u32 unused;
        u32 checksum;
};
```

The ctl word is the same as in 5.7.10, and defines transfer parameters such as VME address modifier, endianness swapping, and whether the transfer reads or writes from VME. This word is very similar to the lower word of the BAR0 page descriptors. One important difference, however, is that endianness AUTO is not allowed, endianness must be specified explicitly.

The len word is the number of bytes to transfer.

The vme_addr is the 64-bit address in VME space to transfer.

The bus_addr is the 64-bit address in the host PC to transfer. The next_addr is the 64-bit address in the host PC of the next descriptor. Both bus_addr and next_addr are bus addresses on the host PC. In a virtual memory system (practically any OS) it is important to note that these bus addresses are not the same as user-space virtual addresses. For that matter, large blocks of memory that are contiguous in virtual memory space may not be in physical space, or may not even be in RAM at the moment.

Checksum is the inverse of the wordwise sum of all other words in the struct, such that the sum of the entire structure is 0xFFFF_FFFF. This can be calculated with the following code:

```
static void set_dma_cksum(struct v120_dma_hwdesc_t *h)
{
        u32 *p;
        u32 sum;
        for (sum = 0, p = (u32 *)h; p < &h->checksum; ++p)
                sum += *p;

        h->checksum = ~sum;
}
```

The presence of a checksum, and specifically one that is valid for memory that is neither all zeros nor all ones, prevents an incorrect pointer in a next_addr leading to data corruption.

4.8.3. *DMA with libV120*

Using the Highland provided driver and interface library for Linux, using DMA from userspace becomes substantially easier. The transfer is described in a linked list of struct v120_dma_desc_t objects, which are the userspace analogue of the DMA descriptor chain, with virtual memory pointers. This is applied against an open V120_HANDLE as returned by v120_open by calling the v120_dma_xfr function, which blocks until completion and returns zero for success or a non-zero failure code.

Detailed information about this process is available in the V120 Linux Programmer's Manual, and examples examples/dmatest.c and examples/dmawrite.c are provided

5. Detailed Register Description

5.1. PAGEDESC Register Map

The PAGEDESC region holds the 8,192 PAGEDESC registers.

5.1.1. *PAGEDESC[n] (BAR0 + 0x8 * n) (Read-Write)*

Controls the operation of VME page n, a 16 kB region starting at address (BAR1 + 0x4000 * n). When working with this register using two 32-bit transfers, rather than a single atomic 64-bit transfer, note that bits 31-0 are located at the lower address and bits 63-32 at the higher address, in keeping with the PCIe little-endian convention.

63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48
ADDR															

47	46	45	44	43	42	41	40	39	38	37	36	35	34	33	32
ADDR															

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDR															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADDR		▨		SP	E		RO	S		AM					

Field	Range	Description
ADDR	[63:14]	VME address bits 63-14.
SP	[11]	Set to 1 to indicate that the target VME modules in this page do not support D32 operations, and 32-bit requests should be split into two sequential D16 transfers.
E	[10:9]	Page endian swapping control. AUTO 0 Swapping based on access size. BYTE 1 Assumes all data is natively bytes. WORD 2 Assumes all data is natively 16-bit words. LONG 3 Assumes all data is natively 32-bit longs.
RO	[8]	Set to 1 to mark this page as read only. Write attempts will immediately cause BERRs.
S	[7:6]	VME access speed. 3 is fastest, 0 is slowest.

Field	Range	Description
AM	[5:0]	VME address modifier.

5.2. CTLREG Register Map

5.2.1. *VXI_MFR (BAR0 + 0x10000) (Read-Only)*

The Highland VXI manufacturer number: 0xFEEE.

5.2.2. *MODTYPE (BAR0 + 0x10004) (Read-Only)*

The module identifier: either 22120 for the V120 or 22124 for the V124.

5.2.3. *MODREV (BAR0 + 0x10008) (Read-Only)*

The module hardware revision letter.

5.2.4. *SERIAL (BAR0 + 0x1000C) (Read-Only)*

The module serial number.

5.2.5. *DASH (BAR0 + 0x10010) (Read-Only)*

The module dash number.

5.2.6. *ROM_ID (BAR0 + 0x10020) (Read-Only)*

A number representing the ROM build ID: 22120.

5.2.7. *ROM_REV (BAR0 + 0x10024) (Read-Only)*

The revision information for this ROM.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	DRAFT							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	LETTER							

Field	Range	Description
DRAFT	[23:16]	The draft code of this revision. For production firmware, this field will be 0. (Unsigned)
LETTER	[7:0]	The ASCII character representing this revision, beginning with 0x41, "A" (Unsigned)

5.2.8. *STAMP (BAR0 + 0x10028) (Read-Only)*

The 32-bit ROM build stamp.

5.2.9. *STATUS (BAR0 + 0x10040) (Read-Only)*

Summary of crate controller status.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									UA				DHCP	LF	EA

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PE	FE	TE	UF	DF										VXI	UG

Field	**Range**	**Description**
UA	[22]	USB connected.
DHCP	[18]	Ethernet is configured using DHCP. DHCP name will be v120-nnnnn or v124-nnnnn, for a V120 or V124 respectively, where nnnnn is the five digit serial number.
LF	[17]	Ethernet link is up at 100 Mbps.
EA	[16]	Ethernet connected.
PE	[15]	Power supply failure flag; for more details see the PFLAGS register 5.2.29
TE	[13]	Temperature high/low warning flag.
UF	[12]	Upgrade firmware is present but failed.
DF	[11]	Firmware is a draft revision, rather than a final release product.
VXI	[1]	1 for a V124 VXI controller. 0 for a V120 VME controller.
UG	[0]	Upgrade firmware is running.

5.2.10. *MCOUNT (BAR0 + 0x10044) (Unsigned Read-Only)*

Microprocessor interrupt counter; incremented by the ARM processor at about 1 KHz.

5.2.11. *UPTIME (BAR0 + 0x10048) (Unsigned Read-Only)*

Microprocessor up-time in seconds.

5.2.12. *ULED (BAR0 + 0x1004C) (Read-Write)*

User LED control. An orange LED is provided on the front panel for user application. The **ULED** register allows user flash patterns to be loaded. An internal shift register is periodically loaded from the contents of the **ULED** register, and the bit 15 of this register operates the orange LED. The shift register is left-shifted every 125 milliseconds, and the register is reloaded every 16 shifts, namely every 2 seconds.

ULED pattern 0x00000000 turns the user LED off. Pattern 0x0000FFFF turns it steady on.

5.2.13. *DIPS (BAR0 + 0x10050) (Read-Only)*

The current DIP switch settings.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LINDIP								CABLE				UNIT			

Field	Range	Description
LINDIP	[15:8]	The 8 discrete dip switches.
CABLE	[7:4]	The PCIe cable length rotary hex switch.
UNIT	[3:0]	The unit ID rotary hex switch. This value will be displayed on the front panel indicator.

5.2.14. *MACRO (BAR0 + 0x10058) (Read-Write)*

Macro register. See section 4.6 - Macros for more information.

5.2.15. *MP0, MP1, MP2, MP3 (BAR0 + 0x1005C-0x1006C) (Read-Write)*

Macro parameter registers. The defined use of these parameters depends on the macro executed.

5.2.16. *VME_ACC (BAR0 + 0x10080) (Read-Only)*

Result of the last VME request from PCI Express.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMER															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨											AF	BTO	RETRY	BERR	DTACK

Field	Range	Description
TIMER	[31:16]	Number of 8ns clock ticks that the transaction took to complete. (Unsigned Read-Only)
AF	[4]	Set to 1 when a transaction cannot complete due to a failure to win arbitration on the VME bus. (Read-Only)
BTO	[3]	Set to 1 when a transaction does not complete before the bus timer expires. (Read-Only)
RETRY	[2]	Set to 1 when a transaction completes with a RETRY response. (Read-Only)
BERR	[1]	Set to 1 when a transaction completes with a BERR response. (Read-Only)
DTACK	[0]	Set to 1 when a transaction completes with a DTACK response. (Read-Only)

5.2.17. ***VME_WC (BAR0 + 0x10084) (Unsigned Read-Write)***

Counts the total number of executed VME write cycles. A write to either VME_WC or VME_RC will clear both.

5.2.18. ***VME_RC (BAR0 + 0x10088) (Unsigned Read-Write)***

Counts the total number of executed VME read cycles. A write to either VME_WC or VME_RC will clear both.

5.2.19. ***UTILITY (BAR0 + 0x1008C) (Read-Write)***

VME utility bus control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SYSCLK_KHZ															

[15:4]	3	2	1	0
▨	SYSCLK	ACFAIL	SYSFAIL	SYSRESET

Field	Range	Description
SYSCLK_KHZ	[31:16]	Reads back the frequency of the (nominally) 16 MHz SYSCLK line in kHz. (Unsigned Read-Only)
SYSCLK	[3]	Reads back the state of the 16 MHz SYSCLK line. (Read-Only)
ACFAIL	[2]	Reads back the state of the (active-low) ACFAIL line. (Read-Only)
SYSFAIL	[1]	Reads back the state of the (active-low) SYSFAIL line. (Read-Only)
SYSRESET	[0]	Reads back the state of the (active-low) SYSRESET line. Write a 1 to this bit to initiate a SYSRESET, which will hold SYSRESET low for the VME minimum of 200 ms.

5.2.20. ***REQUESTER (BAR0 + 0x10090) (Read-Write)***

Controls the VME requester logic in a multi-mastered system.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨											FAIR	▨		BR	

Field	Range	Description
FAIR	[4]	When set, enables the FAIR arbitration algorithm, or "request on no request". This helps in extremely multi-mastered system at the expense of 40ns of additional latency. The default is off.

Field	Range	Description
BR	[1:0]	Selects which of the 4 BR[3..0] levels to use when requesting the VMEbus. With a priority arbiter, 3 is the highest priority and 0 the lowest priority. Default is 3. (Unsigned)

5.2.21. *ARBSTATE (BAR0 + 0x10094) (Read-Only)*

Provides information about the state of the VME arbitration bus. Some of the information in this register is only valid when this module is the system controller, but much is read directly from the bus and is valid regardless.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
									AN		AG	GRANT	REQ	BCLR	BBSY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RSTIN				BGOUT				BGIN				BR			

Field	Range	Description
AN	[22:21]	When this module is the bus arbiter and AG=1, the BR level that currently has the bus grant. (Unsigned Read-Only)
AG	[20]	Set to 1 when this module is the bus arbiter, and any module currently has a bus grant. (Read-Only)
GRANT	[19]	Set to 1 when this module's VME requester currently holds the bus grant. (Read-Only)
REQ	[18]	Set to 1 when this module's VME requester is requesting the bus. (Read-Only)
BCLR	[17]	The current state of the (active-low) BCLR line. (Read-Only)
BBSY	[16]	The current state of the (active-low) BBSY line. (Read-Only)
RSTIN	[15:12]	The state of the (active-low) BGIN[3..0] lines latched at module initialization. The VME specification allows these lines to be used for the First Slot Detector (FSD) function as follows: if BGIN3 is low at power-on time then this module is in slot 1, and should be used as the system controller. Otherwise, the module is in a downstream slot, and should not.

Field	Range	Description
		Note: the VXI specification explicitly discourages the use of FSD in VXI systems. (Read-Only)
BGOUT	[11:8]	The current state of the (active-low) BGOUT[3..0] lines as driven by this module. (Read-Only)
BGIN	[7:4]	The current state of the (active-low) BGIN[3..0] lines coming into this module. (Read-Only)
BR	[3:0]	The current state of the (active-low) BR[3..0] lines. (Read-Only)

5.2.22. ***CLOCKCTL (BAR0 + 0x000100C4) (Read-Write)***

Controls the clocking mode for the VXI clocks. If the clock connector is an input, then the clocks will be derived from an externally provided 10 MHz reference clock. Otherwise the clocks will be derived from the onboard oscillator. V124 only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														MODE	

Field	Range	Description		
MODE	[1:0]	Clocking mode selection.		
		OFF	0	Internal clock, connector is high-impedance.
		OUT	1	Internal clock, 10 MHz TTL output on connector.
		INZ	2	External clock, high-impedance input.
		INT	3	External clock, 50 ohm input.

5.2.23. ***CLOCKSTA (BAR0 + 0x000100C8) (Read Only)***

Status of the front panel 10 MHz clock.

If MODE requests an external clock is requested but one is not present, ERR will be high and the internal clock will be used as a reference instead. This condition will automatically fix itself should an external clock become available. V124 only.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	ERR	DRV	EXTL	PRESENT

Field	Range	Description
ERR	[3]	1 if an external clock is requested but not present.
DRV	[2]	1 if the clock connector is being driven by the V120 (OUT mode).
EXTL	[1]	1 if the backplane clocks are derived from an external signal.
PRESENT	[0]	1 if a signal is present on the clock connector.

5.2.24. ***FPCTL (BAR0 + 0x000100CC) (Read-Write)***

Control the front panel monitor connectors. These connectors can be used to monitor events on the VME bus, or as basic GPIOs under software control.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Z								Y							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
X								W							

Field	Range	Description
Z	[3]	Control the Z connector.
Y	[2]	Control the Y connector.
X	[1]	Control the X connector.
W	[0]	Control the W connector.

For each connector, there is a list of signals that can be selected to drive from it

Mnemonic	Value	Description
INPUT	0	General purpose logic input (high-impedance)
LOW	1	Drive low
HIGH	2	Drive high
DS0	4	DS0*
DS1	5	DS1*
DSA	6	Either of DS[1:0]* is low
AS	7	AS*
WRITE	8	WRITE*
LWORD	9	LWORD*
DTACK	10	DTACK*
BERR	11	BERR*
RETRY	12	RETRY*
TIMO	13	The V120 timed out.
SYSCLK	14	16 MHz SYSCLK.
PCIIRQ	20	The PCIIRQ flag is set.
IACK	22	IACK*
IRQA	23	Any IRQ* line is low.
IRQ1	24	The IRQ1* line is low.
IRQ2	25	The IRQ2* line is low.
IRQ3	26	The IRQ3* line is low.
IRQ4	27	The IRQ4* line is low.
IRQ5	28	The IRQ5* line is low.
IRQ6	29	The IRQ6* line is low.
IRQ7	30	The IRQ7* line is low.
CLKREQ	32	0 if the onboard 10 MHz is requested, 1 for the V124 CLOCK IN.

Mnemonic	Value	Description
CLKACT	33	0 if the onboard 10 MHz is active, 1 for the V124 CLOCK IN.
CLK0BAD	34	1 if the onboard 10 MHz clock is bad.
CLK1BAD	35	1 if the V124 CLOCK IN is bad.
CLKLOCK	36	1 if the SYSCLK PLL is locked.
C1MHZ	37	1 MHz from the SYSCLK PLL.
CPPS	38	1 PPS from the SYSCLK PLL.
CLKSW	39	SYSCLK switch requested.
LBR	40	The V120 is requesting the data bus.
LBG	41	The V120 has been granted the data bus.
RBR	42	Another requester is requesting the data bus.
RBG	43	Another requester has been granted the data bus.

5.2.25. *FPMON (BAR0 + 0x000100D0) (Read-Only)*

Read the signals on the four front-panel monitor connectors.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
												Z	Y	X	W

Field	Range	Description
Z	[3]	Signal at the Z connector.
Y	[2]	Signal at the Y connector.
X	[1]	Signal at the X connector.
W	[0]	Signal at the W connector.

5.2.26. ***MACLO:HI (BAR0 + 0x100F0:0x100F4) (Unsigned Read-Only)***

MACLO is the 32 LSBs of the unit MAC address. MACHI is the 16 MSBs of the address. These can be read as a 64-bit integer to give the full 48-bit MAC address. Note that MAC addresses are stored in reverse order so as to appear correctly when treated as a 6-byte array. Therefore, if the unit has a MAC address of 0a:0b:0c:0d:0e:0f then MACLO will be 0x0D0C0B0A and MACHI will be 0x00000F0E, so the following bytes will appear at the following addresses:

Address	**Hex Value**
0x100F0	0A
0x100F1	0B
0x100F2	0C
0x100F3	0D
0x100F4	0E
0x100F5	0F
0x100F6	00
0x100F7	00

5.2.27. ***IP (BAR0 + 0x100F8) (Unsigned Read-Only)***

IP is the unit IP address. If the unit is configured for a static IP, this register will give that address regardless of connection status. If configured for DHCP, the register will yield the current IP address if the network is connected and DHCP has been successfully performed or 0 if there is no network address.

IP addresses are stored in reverse order so as to appear correctly when treated as a 4-byte array. Therefore, if a unit has an IP address of 1.2.3.4 then IP will be 0x04030201, so the following bytes will appear at the following addresses:

Address	**Hex Value**
0x100F8	01
0x100F9	02
0x100FA	03
0x100FB	04

5.2.28. *SLOT (BAR0 + 0x10104) (Read-Write)*

Controls whether this module is the system controller. This is also called slot 0 in a VXI system or slot 1 in VME, and generally refers to the leftmost card in the system.

Initially, all bits in this register will be set to 1 or 0 collectively, based on the power-on state of the DIP switches (see 3.1.4). In general, this should not be changed at run-time.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	IACK	▨	▨	▨	SYSCON

Field	Range	Description
IACK	[4]	If set to 1, provide interrupt handling. TODO: Interrupts not yet available.
SYSCON	[0]	If set to 1, generate the VME (and VXI for a V124) backplane clocks and provide bus arbitration (request/grant) services.

5.2.29. *PFLAGS (BAR0 + 0x10140) (Read-Only)*

Individual power/temperature error flags.

31	30	29	28	27	26	25	24
▨	▨	▨	▨	▨	▨	▨	▨

23	22	21	20	19	18	17	16
I3N	I3V	I2N	I2V	I1N	I1V	N5N	N5V

15	14	13	12	11	10	9	8
N2N	N2V	P3N	P3V	N24N	N24V	P24N	P24V

7	6	5	4	3	2	1	0
N12N	N12V	P12N	P12V	P5N	P5V	▨	TEMP

Field	Range	Description
I3N	[23]	Internal +3.3V noise out of spec. Indicates hardware failure.

Field	Range	Description
I3V	[22]	Internal +3.3V DC out of spec. Indicates hardware failure.
I2N	[21]	Internal +2.5V noise out of spec. Indicates hardware failure.
I2V	[20]	Internal +2.5V DC out of spec. Indicates hardware failure.
I1N	[19]	Internal +1.2V noise out of spec. Indicates hardware failure.
I1V	[18]	Internal +1.2V DC out of spec. Indicates hardware failure.
N5N	[17]	Crate -5.2V noise out of spec. V124 only.
N5V	[16]	Crate -5.2V DC out of spec. V124 only.
N2N	[15]	Crate -2V noise out of spec. V124 only.
N2V	[14]	Crate -2V DC out of spec. V124 only.
P3N	[13]	Crate +3.3V noise out of spec.
P3V	[12]	Crate +3.3V DC out of spec. +3.3V is only present on VXI4/VME64x and as such will not cause an error in the STATUS register if not present.
N24N	[11]	Crate -24V noise out of spec. V124 only.
N24V	[10]	Crate -24V DC out of spec. V124 only.
P24N	[9]	Crate +24V noise out of spec. V124 only.
P24V	[8]	Crate +24V DC out of spec. V124 only.
N12N	[7]	Crate -12V noise out of spec.
N12V	[6]	Crate -12V DC out of spec.
P12N	[5]	Crate +12V noise out of spec.
P12V	[4]	Crate +12V DC out of spec.
P5N	[3]	Crate +5V noise out of spec.
P5V	[2]	Crate +5V DC out of spec.
TEMP	[0]	Temperature outside 0-60 degrees C.

5.2.30. ***TEMP (BAR0 + 0x10144) (Signed Read-Only)***

PCB temperature, in units of 0.1 degrees C

5.2.31. *Voltage Monitors (Read-Only)*

All of the V120 voltage monitors use the same format to provide information about the DC voltage and noise on a given power rail.

Register	BAR0 Offset	Source	Description
P5V	0x1014C	VMEVXI	Crate +5 voltage and noise
P12V	0x10150	VME/VXI	Crate +12 voltage and noise
N12V	0x10154	VME/VXI	Crate -12 voltage and noise
P24V	0x10158	VXI	Crate +24 voltage and noise
N24V	0x1015C	VXI	Crate -24 voltage and noise
P3.3V	0x10160	VME64/ VXI4	Crate +3.3 voltage and noise
N2V	0x10164	VXI	Crate -2 voltage and noise
N5.2V	0x10168	VXI	Crate -5.2 voltage and noise
P1.2X	0x1016C	Internal	Internal +1.2 voltage and noise
P2.5X	0x10170	Internal	Internal +2.5 voltage and noise
P3.3X	0x10174	Internal	Internal +3.3 voltage and noise

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
AC															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DC															

Field	Range	Description
AC	[31:16]	Peak-to-peak noise voltage on the supply, in mV. (Unsigned)
DC	[15:0]	DC voltage on the supply, in mV. (Signed)

5.2.32. *RAM (BAR0 + 0x10200) (Read-Write)*

256 bytes of scratchpad RAM. The V120/V124 will not use this memory for anything, leaving it available for user data.

5.2.33. *BUFFER (BAR0 + 0x10400) (Read-Write)*

1 kB of macro data buffer. This space is primarily used by the flash macros, and the exact purpose varies by macro used.

5.3. MONITOR Register Map

The MONITOR peripheral provides 4 VME monitor blocks. Each block is individually programmable for the types of events that it captures. Captures will continue happening so long as the capture block is enabled. This means that, if VME accesses are continuing in the background, the capture block can be overwritten while it is being read. If this is a concern, each capture block also includes a one-shot function that will disable the block after it has fired.

The power-on default state of the monitors is for monitor 0 to capture all transactions, for monitor 1 to capture all irregular (non-DTACK) transactions, and for monitors 2-3 to be disabled.

Each monitor block is 64 bytes long, and contains several registers. Located at address BAR0 + 0x14000, the PCIe monitor can be expressed as

```
struct monitor {
    uint32_t mon_ctl;
    uint32_t __padding0;
    uint32_t mon_trans;
    uint32_t mon_resp;
    uint64_t addr;
    uint64_t data;
    uint32_t __padding1[8];
} MONITOR[4];
```

5.3.1. *MON_CTL[n] (BAR0 + 0x14000 + 0x40 * n) (Read-Write)*

Control the capture for this block. There are three different qualifier sets that the transaction must meet to be captured. First, the ENABLE bit must be set, allowing the block to capture anything at all. Secondly, the transaction type (READ/WRITE/IACK) must be set to capture transactions of that type. Finally, the response type (DTACK/BERR/BTO/RETRY) must be set to capture transactions with responses of that type.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
IACK		READ	WRITE												AF

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RETRY	BTO	BERR	DTACK										CLR	OS	ENABLE

Field	Range	Description
IACK	[31]	Set to 1 to capture interrupt acknowledge transactions.
READ	[29]	Set to 1 to capture read transactions.
WRITE	[28]	Set to 1 to capture write transactions.
AF	[16]	Set to 1 to capture transactions with an arbitration failure.
RETRY	[15]	Set to 1 to capture transactions with a RETRY response.
BTO	[14]	Set to 1 to capture transactions with a bus timeout.
BERR	[13]	Set to 1 to capture transactions with a BERR response.
DTACK	[12]	Set to 1 to capture transactions with a DTACK response.
CLR	[2]	Write a 1 to clear this monitor, setting all the read only values to zero. This bit will clear itself upon clear completion (this is quite fast, probably imperceptibly so).
OS	[1]	Set to 1 to set this block to one-shot mode. In one-shot mode, upon completing a data capture the ENABLE bit will be cleared, preventing further acquisition on this block until it is explicitly set again.
ENABLE	[0]	Set to 1 to enable capture on this block. The one-shot action of the OS bit will clear this bit after capture.

5.3.2. ***MON_TRANS[n] (BAR0 + 0x14008 + 0x40 * n) (Read-Only)***

Transaction information from the last captured VME transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	IACK	WRITE

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	SRC		▨	LWORD	DS1	DS0	▨	▨	AM					

Field	Range	Description
IACK	[17]	1 if the transaction was an interrupt acknowledge cycle. (Read-Only)
WRITE	[16]	1 if the transaction was a write. (Read-Only)

Field	Range	Description
SRC	[13:12]	Defines the source of the transaction. (Read-Only) PIO 0 PCI Express PIO COM 1 Serial communications: USB or Ethernet. DMA 2 PCI Express DMA
LWORD	[10]	1 if LWORD was asserted (low) for the transaction, indicating a 32-bit data transaction using all lanes D[31:0] (Read-Only)
DS1	[9]	1 if DS1 was asserted (low) for the transaction, signaling that D[15:8] should be active. (Read-Only)
DS0	[8]	1 if DS0 was asserted (low) for the transaction, signaling that D[7:0] should be active. (Read-Only)
AM	[5:0]	Address mode of the logged transaction. (Read-Only)

5.3.3. ***MON_RESP[n] (BAR0 + 0x1400C + 0x40 * n) (Read-Only)***

Response information from the last captured VME transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMER															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	AF	BTO	RETRY	BERR	DTACK

Field	Range	Description
TIMER	[31:16]	Number of 8ns clock ticks that the transaction took to complete. (Unsigned Read-Only)
AF	[4]	Set to 1 when a transaction cannot complete due to a failure to win arbitration on the VME bus. (Read-Only)
BTO	[3]	Set to 1 when a transaction does not complete before the bus timer expires. (Read-Only)
RETRY	[2]	Set to 1 when a transaction completes with a RETRY response. (Read-Only)

Field	Range	Description
BERR	[1]	Set to 1 when a transaction completes with a BERR response. (Read-Only)
DTACK	[0]	Set to 1 when a transaction completes with a DTACK response. (Read-Only)

5.3.4. *ADDR_LO[n] (BAR0 + 0x14010 + 0x40 * n) (Read-Only)*

VME bus address lines A[31:0]. As VME does not have an A0 line, this bit will always be 0.

5.3.5. *ADDR_HI[n] (BAR0 + 0x14014 + 0x40 * n) (Read-Only)*

VME bus address bits [63:32]. These are used only in multiplexed transfers in the A40 and A64 space; otherwise this register will be zero.

5.3.6. *DATA_LO[n] (BAR0 + 0x14018 + 0x40 * n) (Read-Only)*

VME bus data lines D[31:0].

5.3.7. *DATA_HI[n] (BAR0 + 0x1401C + 0x40 * n) (Read-Only)*

VME bus data bits D[63:32]. These bits exist only for multiplexed transfers; otherwise this register will be zero.

5.4. VXI_CTL Register Map

The VXI_CTL peripheral controls VXI clocking and triggering. While these registers are present on both the V120 and the V124, they serve no purpose on the V120 and should be ignored.

5.4.1. *TTLTRG_DAT[n] (BAR0 + 0x14200 + 0x4 * n) (Read-Write)*

Data for the TTLTRGn line. The meaning of this data is dependent on the corresponding field in TTLTRG_CTL.

5.4.2. *TTLTRG_CTL (BAR0 + 0x14220) (Read-Write)*

Defines the function of the TTLTRG lines. Settings here will control what impact the corresponding TTLTRG_DAT register has.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
T7				T6				T5				T4			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
T3				T2				T1				T0			

Field	Range	Description
T7	[31:28]	Sets the mode for TTLTRG7. See T0 field for definitions.
T6	[27:24]	Sets the mode for TTLTRG6. See T0 field for definitions.
T5	[23:20]	Sets the mode for TTLTRG5. See T0 field for definitions.
T4	[19:16]	Sets the mode for TTLTRG4. See T0 field for definitions.
T3	[15:12]	Sets the mode for TTLTRG3. See T0 field for definitions.
T2	[11:8]	Sets the mode for TTLTRG2. See T0 field for definitions.
T1	[7:4]	Sets the mode for TTLTRG1. See T0 field for definitions.
T0	[3:0]	Sets the mode for TTLTRG0. Unused modes are reserved for future use. STIN 0 Static input; DAT bit 0 reports the current line state. This is inverted from the electrical state, a 1 in DAT means that the line is pulled low. STST 1 VXI start/stop protocol. A 1 in the DAT register (bit 0) causes the line to be pulled low. All changes take place synchronously to CLK10, meeting the VXI requirements of a minimum 50ns setup and 15ns hold.

5.4.3. ***ECLTRG_CTL (BAR0 + 0x00014224) (Read-Write)***

Defines the function of the ECLTRG lines. Settings here will control what impact the corresponding ECLTRG_DAT register has.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
								E1				E0			

Field	Range	Description		
E1	[7:4]	Sets the mode for ECLTRG1. See T0 field for definitions.		
E0	[3:0]	Sets the mode for ECLTRG0. Unused modes are reserved for future use.		
		STIN	0	Static input; DAT bit 0 reports the current line state. This is also the electrical state, a 1 in DAT means the line is being pulled high (-0.8V), rather than floating low (-2.0V).
		STST	1	VXI start/stop VXI start/stop protocol. A 1 in the DAT register (bit 0) causes the line to be pulled high. All changes take place synchronously to CLK10, meeting the VXI requirements of a minimum 50 ns setup and 15 ns hold.

5.4.4. ***ECLTRG_DAT0 (BAR0 + 0x00014228) (Read-Write)***

Data for the ECLTRG0 line. The meaning of this data is dependent on the E0 field in ECLTRG_CTL.

5.4.5. ***ECLTRG_DAT1 (BAR0 + 0x0001422C) (Read-Write)***

Data for the ECLTRG1 line. The meaning of this data is dependent on the E1 field in ECLTRG_CTL.

5.5. IRQHNDL Register Map

The IRQHNDL peripheral handles VME interrupt management. The PCIe IRQ flag is asserted whenever any interrupt-enabled IRQ goes high, and cleared when the host writes to PCIIRQ.

5.5.1. ***IRQSTATUS (BAR0 + 0x14400) (Read-Only)***

Current status of the VME interrupt lines.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								IRQ							

Field	Range	Description
IRQ	[7:0]	VME interrupt status. This field is logically negated from the electrical level of the IRQ[7:1]* lines on the backplane, such that a value of 1 means the interrupt is asserted. Bit 0 of this field cannot be asserted, as the VME backplane has no IRQ[0]* line. These bits will also go high when the IRQEN.FAKE bits are set.

5.5.2. *IRQEN (BAR0 + 0x14404) (Read-Write)*

Control the assertion of PCI interrupts from VME interrupts.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FAKE								EN							

Field	Range	Description
FAKE	[15:8]	VME interrupts to fake. Set a bit in this field to set the corresponding bit in IRQSTATUS to simulate an IRQ of that level. Clear the bit to simulate removal of the interrupt. The LSB of this field is ignored.
EN	[7:0]	VME interrupt enable. Set a bit in this field to cause a PCI interrupt when the corresponding bit in IRQSTATUS goes true. Otherwise the interrupt is available only by polling the IRQSTATUS register. Bit 0 of this field is ignored.

5.5.3. *IACKCFG (BAR0 + 0x14408) (Read-Write)*

Configuration for each of the IACK_VECTOR registers.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
		IACKS7				IACKS6				IACKS5				IACKS4	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
		IACKS3				IACKS2				IACKS1					

Each IACKSn is the VME access speed for IACK cycles generated by reading the IACK_VECTOR[n] registers (see Section 5.5.5). 3 is fastest, 0 is slowest.

5.5.4. *PCIIRQ (BAR0 + 0x1440C) (Read-Write)*

The current state of the PCIe IRQ flag. Writing anything to this register clears the PCIe IRQ flag, but does not alter the value of the IRQSTATUS register. This should be done by the driver to acknowledge the interrupt.

Specifically, the flag is set by the logical or of ((IRQSTATUS.IRQ[n] and IRQEN.EN[n]) transitioning 0 -> 1) over all n 1 through 7.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	IRQ

Field	**Range**	**Description**
IRQ	[0]	The state of the flag. 1 is asserted.

5.5.5. *IACK_VECTOR[n] (BAR0 + 0x14420 + 4*n) (Read-Only)*

Read IACK_VECTOR[n] registers 1-7 to fetch the corresponding interrupt vector over VME by generating an IACK cycle with A[3:1] set to the vector index 1-7. IACK_VECTOR[0] generates a IACK cycle for address 0, which should never find a module willing to DTACK it and thus will timeout with a value of 0xFFFFFFFF.

If the VME interrupter currently asserting the vector is only D16 then only bits 15:0 will be valid, and 31:16 will be 0xFFFF. If the VME interrupter is only D8, then only the appropriate bits will be valid.

5.6. PCIE_MON Register Map

A PCIe transaction recorder is provided for help diagnosing PCI Express issues. Located at address BAR0 + 0x14800, the PCIe monitor can be expressed as

```
struct pcie_mon {
    uint32_t status;
    uint32_t address;
    uint64_t data;
} RECORDS[128];
```

Writing anywhere in this space will reset the recorder. After the reset, all transactions on the PCIe bus will be logged, with the first transaction in RECORDS[0], the second in RECORDS[1], until the end of the space is reached. Note that this space will also log transactions into this space, though the readback data will probably be incorrect as it won't have caught up in time.

5.6.1. ***STATUS[n] (BAR0 + 0x14800 + 0x10 * n) (Read-Write)***

Holds status information about the currently displayed transaction.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VALID	READ	BAR	START	TIMER											

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	COUNT							BYTE_EN							

Field	Range	Description
VALID	[31]	1 if this is a valid transaction. 0 if this record has not been written since the reset; in this case all other transaction data is undefined.
READ	[30]	1 if this transaction was a read, in which case DATA is read data. 0 if this transaction was a write, in which case DATA is write data.
BAR	[29]	1 for a transaction in BAR1. 0 for a transaction in BAR0.
START	[28]	1 if this starts a burst.
TIMER	[27:16]	Time of the transaction, in 8ns clock ticks. (Unsigned)
COUNT	[14:8]	Burst count, number of 64-bit words in this atomic transfer. (Unsigned)
BYTE_EN	[7:0]	Bus byte enables for this transfer.

5.6.2. ***ADDRESS (BAR0 + 0x14804) (Read-Write)***

Address of the first data in the burst. This will not increment as the burst proceeds.

5.6.3. ***DATA_LO (BAR0 + 0x14808) (Read-Write)***

Least significant dword of the data.

5.6.4. ***DATA_HI (BAR0 + 0x1480C) (Read-Write)***

Most significant dword of the data.

5.7. DMA Register Map

The DMA peripheral controls the PCIe to VME DMA scatter-gather DMA engine.

Start a DMA transfer by loading the address of the first DMA descriptor into NEXTDESC and setting CONTROL.RUN true.

The DESC_* registers hold the values read from the last descriptor fetched. They will update live during descriptor fetches, and so are only guaranteed atomic if the DMA engine is not running.

5.7.1. *CONTROL (BAR2 + 0x0000) (Read-Write)*

Control bits for the DMA controller.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
														IACK	RUN

Field	Range	Description
IACK	[1]	Write 1 to this bit to clear the DMA interrupt.
RUN	[0]	If RUN is clear, set it to have the DMA controller begin working from the descriptor pointed to by NEXTDESC. If RUN is set, clear it to have the DMA controller stop processing after the current descriptor is complete. This bit is cleared by the DMA controller when it begins the transfer for the final descriptor (the one for which the NEXTADDR is 0), or when it terminates abnormally.

5.7.2. *STATUS (BAR2 + 0x0004) (Read-Only)*

Status information for the DMA controller. All fields are cleared when RUN transitions from 0 to 1.

31-23	22	21	20	19	18	17	16
	IFLAG	VAERR	BAERR	LENERR	CHKERR	VMEERR	OK

15-8	7-0
DCOMP	DFETCH

Field	Range	Description
IFLAG	[22]	The state of the DMA controller interrupt flag.

Field	Range	Description
VAERR	[21]	The DMA chain terminated because the VME address was not divisible by the VME transfer size. ERRADDR will contain the address of the failed descriptor.
BAERR	[20]	The DMA chain terminated because the bus address was not divisible by the VME transfer size. ERRADDR will contain the address of the failed descriptor.
LENERR	[19]	The DMA chain terminated because the DMA descriptor had a zero length, or a length not divisible by the VME transfer size. ERRADDR will contain the address of the failed descriptor.
CHKERR	[18]	The DMA chain terminated because the DMA descriptor had a bad checksum. ERRADDR will contain the address of the failed descriptor.
VMEERR	[17]	The DMA chain terminated due to a VME error. ERRADDR will contain the address of the descriptor that failed. LASTVME will contain the VME address that failed. VME_ACC will have more information about the transfer.
OK	[16]	The DMA chain completed successfully.
DCOMP	[15:8]	Number of descriptors that have been completed. (Unsigned)
DFETCH	[7:0]	Number of descriptors that have been fetched. (Unsigned)

When any of the status bits in the upper halfword transition to 1, the interrupt flag will be set.

5.7.3. ***NEXTDESC_LO (BAR2 + 0x0008) (Read-Write)***

Next DMA descriptor location in PCIe memory space (NEXTDESC), bits 31:0. While CONTROL.RUN is clear, write to this register to tell the engine where to begin processing. While CONTROL.RUN is set, this register is updated by the DMA engine as the associated transfer for each descriptor begins to indicate the address of the next descriptor that will be executed. While CONTROL.RUN is set writes from the PCIe side are ignored.

5.7.4. ***NEXTDESC_HI (BAR2 + 0x000C) (Read-Write)***

NEXTDESC bits 63:32.

5.7.5. *ERRADDR_LO (BAR2 + 0x0010) (Read-Only)*

PCIe address of the DMA descriptor in which an error, indicated by either STATUS.VMEERR or STATUS.DMAERR, occurred (ERRADDR), bits 31:0

5.7.6. *ERRADDR_HI (BAR2 + 0x0014) (Read-Only)*

ERRADDR bits 63:32

5.7.7. *LASTVME_LO (BAR2 + 0x0018) (Read-Only)*

Address of the last VME access attempted (LASTVME), bits 31:0. This is fast moving status information while the DMA transfer is occurring, but will be the address of the VME failure if STATUS.VMEERR is set.

5.7.8. *LASTVME_HI (BAR2 + 0x001C) (Read-Only)*

LASTVME bits 63:32

5.7.9. *VME_ACC (BAR2 + 0x0020) (Read-Only)*

Status of the last VME access attempted. This is fast moving status information while the DMA transfer is occurring, but will be the information on the VME failure if STATUS.VMEERR is set.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TIMER															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	AF	BTO	RETRY	BERR	DTACK

Field	Range	Description
TIMER	[31:16]	Number of 8 ns clock ticks that the transaction took to complete. (Unsigned)
AF	[4]	Set to 1 when a transaction cannot complete due to a failure to win arbitration on the VME bus.
BTO	[3]	Set to 1 when a transaction does not complete before the bus timer expires.
RETRY	[2]	Set to 1 when a transaction completes with a RETRY response.
BERR	[1]	Set to 1 when a transaction completes with a BERR response.
DTACK	[0]	Set to 1 when a transaction completes with a DTACK response.

5.7.10. ***DESC_CTL (BAR2 + 0x0024) (Read-Only)***

Descriptor control word of the last DMA descriptor read.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	WRITE	HOLD

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
▨	▨	▨	▨	SPLIT	ENDIAN		▨	SPEED		AM					

Field	Range	Description
WRITE	*[17]*	Set to 1 for a write, 0 for a read.
HOLD	*[16]*	Set to 1 to use a constant VME address.
SPLIT	*[11]*	Set to 1 to force D16 access, 0 for D32.
ENDIAN	*[10:9]*	Endian mode, same as for page descriptors.
SPEED	*[7:6]*	VME transaction speed, same as for page descriptors.
AM	*[5:0]*	VME address modifier.

5.7.11. ***DESC_LEN (BAR2 + 0x0028) (Read-Only)***

Number of bytes to transfer for last DMA descriptor read.

5.7.12. ***DESC_VMEADDR_LO (BAR2 + 0x002C) (Read-Only)***

VME start address for last DMA descriptor read, bits 31:0

5.7.13. ***DESC_VMEADDR_HI (BAR2 + 0x0030) (Read-Only)***

DESC_VMEADDR bits 63:32

5.7.14. ***DESC_BUSADDR_LO (BAR2 + 0x0034) (Read-Only)***

PCIe start address for last DMA descriptor read, bits 31:0

5.7.15. ***DESC_BUSADDR_HI (BAR2 + 0x0038) (Unsigned Read-Only)***

DESC_BUSADDR bits 63:32

5.7.16. ***DESC_NEXTADDR_LO (BAR2 + 0x003C) (Read-Only)***

PCIe next descriptor address for last DMA descriptor read, bits 31:0

5.7.17. ***DESC_NEXTADDR_HI (BAR2 + 0x0040) (Read-Only)***

DESC_NEXTADDR bits 63:32

5.7.18. ***DESC_UNUSED (BAR2 + 0x0044) (Read-Only)***

Unused word of last DMA descriptor read.

5.7.19. ***DESC_CHECKSUM (BAR2 + 0x0048) (Read-Only)***

Checksum word of last DMA descriptor read.

6. PCI Express

6.1. PCIe Overview

For maximum performance, the V120 can communicate with the host PC over 4 lane, PCI Express Gen 1. This provides 8 Gbps in each direction between the PC and V120.

PCI Express is logically a different transport layer over top of PCI. The PCIe device (in this case the V120) appears with its registers mapped into the host address space.

6.2. PCIe Host Interfaces and Cabling

To connect the V120 over a PCIe link, a host adapter board and cable are required. 4-lane host adapter cards are available from Highland Technology as the J120 and J121. The J120 transparently translates the host PCIe to a single 4-lane cable for connection to a single V120. The J121 uses an active bridge to connect two 4-lane cables for 2 V120s in a single host slot. Highland Technology also sells 4-lane PCIe cables in 2, 4, and 7 meter lengths, as the J4-2, J4-4, and J4-7 respectivelyFor longer runs than 7 meters, fiber optic translators such as the Samtec PCIES can be used. Highland does not directly sell these adapters, but has verified the Samtec optical cable as working for runs up to 300 meters. The J121 specifically is known to notwork with the PCIEO optical cable, as it does not provide cable power.

The V120 can also be connected to 8-lane host adapters by using a 4-lane cable with an 8-lane end, such as the Molex 74546-4082.

7. Endian Control

7.1. The Challenge

The term “endianness” is used to discuss the way in which data is translated between hardware registers, which are inherently atomic and correctly ordered, and memory buses, which are not.

VMEbus is byte addressed and "big endian". The "first" byte (the lowest address) of a multi-byte value is the 8 most significant bits. When a 32-bit value is stored in two consecutive 16-bit VME registers, the register at the lower address holds bits 31:16, and the higher register holds bits 15:0.

The PCI bus, and therefore as a consequence, PCIe, are byte addressed and "little endian". The first byte (the lowest address) of a multi-byte value is the 8 least significant bits. When a 32-bit PCIe register is accessed with two 16-bit reads, the read of the lower address holds bits 15:0, and the higher address holds bits 31:16.

The following table demonstrates reading from the same 32-bit register on both little-endian PCIe systems and big-endian VME systems.

PCIe Read	Bytes	Data	VME Read	Bytes	Data
32-bit	0-3	0x12345678	32-bit	0-3	0x12345678
16-bit	0-1	0x5678	16-bit	0-1	0x1234
	2-3	0x1234		2-3	0x5678
8-bit	0	0x78	8-bit	0	0x12
	1	0x56		1	0x34
	2	0x34		2	0x56
	3	0x12		3	0x78

This presents a difficulty; when bridging to VME from PCIe the question of which bytes go into which locations is dictated by what the actual size of the data being transferred is. To further complicate matters, while the PCIe bus works with a base unit of 32-bit transfers, VMEbus allows for base units of either 16 or 32-bits, with data lines 31-16 considered optional.

7.2. The Solution

We note here that there can be no single, automatic solution to endian conflicts, as any given data transfer involves assumptions on both ends which are unknown at the other end. The V120 includes endian control bits in each page register so that specific data transfers can be customized based on what the user knows about data types and intent, as well as a transaction split bit to determine whether the upper 16 data lines can be used.

The size of a programmed VME data transfer is determined by the request made by the host processor when initiating the operation. For the common case of an Intel x86 processor, the transfer size is a part of the instruction opcode. In Intel syntax assembly, this would be the difference between the three statements:

```
movzx  eax,BYTE PTR [rax]
movzx  eax,WORD PTR [rax]
mov    eax,DWORD PTR [rax]
```

Or more readably in C (using C99 integer types for the sake of explicit sizing):

```
uint8_t byte = *(uint8_t *)data;      // a.k.a. unsigned char
uint16_t word = *(uint16_t *)data;    // a.k.a. unsigned short
uint32_t dword = *(uint32_t *)data;   // a.k.a. unsigned long
```

Using the default V120 endianness mode of Access Dependent, the transfer size is assumed to be the inherent size of the data being transferred, and thus determines the translation between VME and PCIe spaces. This sounds complicated, but is the access mode most likely to “just work”. This access mode preserves the meaning of the data, or at least as well as it can. The downside of this mode is that, because of the access dependence, the translations will come out differently when moving small amounts of data at a time than when trying to use block copy instructions.

The other three modes are Byte, Word, and Dword modes. In contrast to the Access Dependent mode, the three remaining modes will move the same data to the same memory locations regardless of the PCI transaction type. The goal of these transaction modes is to treat the data at any given location the same way no matter what. So Byte mode moves data with the same bytes at the same memory locations on both sides of the bus. Word mode keeps words, and Dword mode keeps dwords.

Access Dependent mode performs a Byte transfer when moving a byte at a time, a Word transfer on words, and a Dword transfer on dwords.

Each V120 page descriptor has a 2-bit "E" (endian) field and the single SP (split) bit.

The two encoded E bits define four data transfer modes:

E	Code	Meaning
0	Access	like-sized transfers of all types are correct.
1	Byte	bytes are copied in memory order in all transfers
2	Word	16-bit words are copied in memory order in all transfers
3	Dword	32-bit longwords are copied in memory order in all transfers

To address the VME bus size issue, each page descriptor also has the SP bit. If SP is set, addressed VME modules are assumed to have no P2 connector. A 32-bit PCI Express access to such a module is emulated with two 16-bit VME cycles, with the (lower) even address first, and the (higher) odd address second.

7.3. Clever Endianness Tricks

The V120 per-page endianness and split transaction controls can simplify access to a wide range of VME hardware. Take for example a Highland Technology V470. This model reports the temperature of RTD reference A as a pair of registers using conventional VME big-endian ordering, the most significant word (RAHI) is located at byte offset 0x58, and the least significant word (RALO) at byte offset 0x5A. Additionally, this module supports only D16 accesses.

By setting the page accessing the V470 to use the default Direct endian mode (0) and setting the Split Transaction bit, 32-bit accesses will be transformed into two 16-bit accesses, with the most significant word at the numerically lower memory address, i.e. traditional big-endian. Thus, given `volatile uint32_t *rtda = base + 0x58;` reading from the pointer will correctly read the RTD value.

Now take, on the other hand, a module that reverses these two words, placing the least significant word at the numerically lowest memory address. This is often done in older hardware as an attempt to simply access for x86 users. In this case, RALO would be at byte offset 0x58 and RAHI and byte offset 0x5A.

By setting that page to use Word endian mode (2), and setting Split Transaction, the VME bus will still perform two sequential accesses when asked for a 32-bit read, but the first access to address 0x58 will convey the least significant word (i.e. the lower address in an x86) and the second the most significant. 16-bit accesses to other registers will continue to operate as expected.

It should be noted that, in Word mode, 8-bit accesses will occur at the requested byte address XOR 1. For instance, given a 16-bit register at offset 0x60 with flags in bits 7-0 and unused bits from 15-8, the correct pointer to those bits would be `volatile uint8_t *flags = base + 0x60;` rather than the actual VME memory address of 0x61. This confusion can be avoided by avoiding byte reads and writes to VME, as they convey no technical advantages, and instead working with the least significant byte of `volatile uint16_t *flags = base + 0x60;` .

The use of the proper endianness mode and transaction splitting brings advantages in both code clarity and performance. Transaction splitting allows a transaction requiring two 16-bit VME transfers to be conveyed in a single PCIe packet, greatly reducing overhead. In addition, the host CPU needs to perform fewer operations. The operation penalty becomes even more apparent when endianness swapping is required as well.

Without splitting / swapping

```
    uint32_t res = (V470->rahi << 16);
    res |= V470->ralo;
    uint32_t old_ibuf = V208->ibuf_lo;
    old_ibuf |= (V208->ibuf_hi << 16);
    V208->ibuf_lo = (new_ibuf & 0xFFFF);
    V208->ibuf_hi = (new_ibuf >> 16);
```

With splitting / swapping

```
    uint32_t res = V470->rtda;
    uint32_t old_ibuf = V208->ibuf;
    V208->ibuf = new_ibuf;
```

7.4. *Endian Translation Details*

7.4.1. *Mode 0 – Access Dependent mode*

In Access Dependent mode, reads and writes to VME space will occur as a transaction of the requested transaction size, based at the requested base address. Thus bytes will move single bytes, words will move 2 bytes, and dwords will move 4 bytes.

Big Endian (VME)　　　Little Endian (PCI)

Transaction (Address 0)	**Source (VME)**	**VME Backplane**	**PCI Bus**	**Destination (PCI)**
Byte Read	12 34 56 78	12__	______12	12 __ __ __
Word Read	12 34 56 78	1234	____1234	34 12 __ __
Dword Read (D32, SP=0)	12 34 56 78	12345678	12345678	78 56 34 12
Dword Read (D16, SP=1)	12 34 56 78	1234 5678	12345678	78 56 34 12
	Source (PCI)	**PCI Bus**	**VME Backplane**	**Destination (VME)**
Byte Write	78 56 34 12	______78	78__	78 __ __ __
Word Write	78 56 34 12	____5678	5678	56 78 __ __
Dword Write (D32, SP=0)	78 56 34 12	12345678	12345678	12 34 56 78
Dword Write (D16, SP=1)	78 56 34 12	12345678	1234 5678	12 34 __ __ 12 34 56 78

7.4.2. *Mode 1 – Byte mode*

In Byte mode, each byte will wind up at the same memory location in both PCI and VME spaces. This allows data elements like strings and binary objects to be transferred.

Big Endian (VME)　　　Little Endian (PCI)

Transaction (Address 0)	**Source (VME)**	**VME Backplane**	**PCI Bus**	**Destination (PCI)**
Byte Read	12 34 56 78	12__	______12	12 __ __ __
Word Read	12 34 56 78	1234	____3412	12 34 __ __
Dword Read (D32, SP=0)	12 34 56 78	12345678	78563412	12 34 56 78
Dword Read (D16, SP=1)	12 34 56 78	1234 5678	78563412	12 34 56 78
	Source (PCI)	**PCI Bus**	**VME Backplane**	**Destination (VME)**
Byte Write	78 56 34 12	______78	78__	78 __ __ __
Word Write	78 56 34 12	____5678	7856	78 56 __ __
Dword Write (D32, SP=0)	78 56 34 12	12345678	78563412	78 56 34 12
Dword Write (D16, SP=1)	78 56 34 12	12345678	7856 3412	78 56 __ __ 78 56 34 12

7.4.3. *Mode 2 – Word mode*

In Word mode, each 16-bit word will wind up at the same memory location in both PCI and VME spaces. All memory accesses will be treated as if they are either parts or combinations of 16-bit words, that is to say, all VME byte addresses will equal the PCI address XOR 1.

Big Endian (VME)   Little Endian (PCI)

Transaction (Address 0)	**Source (VME)**	**VME Backplane**	**PCI Bus**	**Destination (PCI)**
Byte Read	12 34 56 78	__34	______34	34 __ __ __
Word Read	12 34 56 78	1234	____1234	34 12 __ __
Dword Read (D32, SP=0)	12 34 56 78	12345678	56781234	34 12 78 56
Dword Read (D16, SP=1)	12 34 56 78	1234 5678	56781234	34 12 78 56
	Source (PCI)	**PCI Bus**	**VME Backplane**	**Destination (VME)**
Byte Write	78 56 34 12	______78	__78	__ 78 __ __
Word Write	78 56 34 12	____5678	5678	56 78 __ __
Dword Write (D32, SP=0)	78 56 34 12	12345678	56781234	56 78 12 34
Dword Write (D16, SP=1)	78 56 34 12	12345678	5678 1234	56 78 __ __ 56 78 12 34

7.4.4. ***Mode 3 – Dword mode***

In Dword mode, each 32-bit dword will wind up at the same memory location in both PCI and VME spaces. All memory accesses will be treated as if they are either parts or combinations of 32-bit dwords, that is to say, all VME byte addresses will equal the PCI address XOR 3.

Big Endian (VME) Little Endian (PCI)

Transaction (Address 0)	**Source (VME)**	**VME Backplane**	**PCI Bus**	**Destination (PCI)**
Byte Read	12 34 56 78	__78	______78	78 __ __ __
Word Read	12 34 56 78	5678	____5678	78 56 __ __
Dword Read (D32, SP=0)	12 34 56 78	12345678	12345678	78 56 34 12
Dword Read (D16, SP=1)	12 34 56 78	1234 5678	12345678	78 56 34 12

	Source (PCI)	**PCI Bus**	**VME Backplane**	**Destination (VME)**
Byte Write	78 56 34 12	______78	__78	__ __ __ 78
Word Write	78 56 34 12	____5678	5678	__ __ 56 78
Dword Write (D32, SP=0)	78 56 34 12	12345678	12345678	12 34 56 78
Dword Write (D16, SP=1)	78 56 34 12	12345678	1234 5678	12 34 __ __ 12 34 56 78

8. USB/Ethernet Connections and Protocols

8.1. Ethernet Communications

8.1.1. IP Address

The V120 provides ASCII control commands over TCP/IP.

The firmware supports both static IP addressing and DHCP.

sw1	sw2	IP Address	
OFF	OFF	Static IP	Default is 192.168.254.183, reprogrammable with IP command
OFF	ON	192.168.254.(100+U)	U = UNIT switch, 0-15
ON	OFF	192.168.254.(1+U)	U = UNIT switch, 0-15
ON	ON	DHCP	Defaults to the static IP setting if no DHCP server can be found.

The selection code 0 static IP address can be changed by using the IP command over Ethernet or through the USB port. The default IP address as shipped is 192.168.254.183 with a subnet mask of 255.255.255.000.

Dipswitch codes 1 and 2 provide hard IP addresses whose last decimal digit are set by the UNIT rotary switch on each controller. This allows multiple V120s to be brought onto a network without a DHCP server, and without having to set individual IP addresses serially.

To use DHCP dynamic IP addressing, use dipswitch selection code 3. This will cause the V120 to query the network for a local DHCP server. DHCPDISCOVER queries begin at power-up, and continue to happen periodically until successful. If name resolution is successful, the V120 will then have a name on the network of “V120-nnnnn,” where nnnnn is the five-digit crate controller serial number. A V124 will have a name of “V124-nnnnn” instead.

Switch settings are read only at power-up.

8.1.2. TCP

Communication with a V120 consists of ASCII command strings sent and received on a TCP socket on port 2000. Most telnet clients can communicate with the V120 using the “raw” mode, or customer software can communicate simply by sending command strings , and can be invoked either through software or using most telnet clients.

8.2. USB Communications

The V120 enumerates as a USB serial port and uses the same ASCII communications protocol as Ethernet. USB and Ethernet operate concurrently. Most operating systems will recognize the FTDI FT230XS chip without additional drivers. Drivers are available from ***www.ftdichip.com.***

Serial port settings should be 115.2 kbaud, 8 data bits, no parity, 1 stop bit, with no handshaking.

8.3. Serial Command Protocol

A USB or TCP command to the V120 is a line of text beginning with a keyword command followed by optional arguments, terminated with either a carriage return, linefeed, or CR+LF pair. Input is case-insensitive and does not echo. Commas are ignored. In general, a command without an argument is a query. Any keyword can be truncated to its first two characters. Multiple commands may be sent in one line, separated by semicolons.

The V120 only transmits in immediate reply to a serial command. All replies are lines of text terminated by a CR+LF. Commands with no expected reply data and no errors evoke the prompt reply:

```
V120> <cr> <lf>
```

Or, to indicate an error:

```
Enn: message <cr> <lf>
V120> <cr> <lf>
```

Numeric values sent to the V120 are assumed decimal, but 0xnnnn is interpreted as hex.

Inquiry type commands evoke:

```
Requested_data <cr> <lf>
```

Returned numerical data is in hex, as in "0x1234ABCD"

8.4. System Commands

8.4.1. ***IDENT***

Returns a string of the form: “V120-1 VME Crate Controller Highland Technology Inc”

The orange USR LED will blink briefly when **IDENT** is executed.

8.4.2. ***STATUS***

Returns a status report (TBD).

8.4.3. ***HELP***

Returns a brief command summary.

8.4.4. ***RESET***

Reboot from the serial flash. uP code and the FPGA configuration are reloaded. The V120 will disappear from the PCIe space and the Ethernet connection will be broken during the reboot.

8.4.5. *POWER*

Returns a power supply report.

8.5. *Control Region Access Commands*

USB/Ethernet commands are provided for accessing the control region, BAR0 pages 0 through 7.

8.5.1. *CWRITE addr v1 [v2 […]]*

Write consecutive 32-bit values to the V120/V124 control registers starting at “addr”, which is 32-bit aligned. The values are assumed to be decimal unless preceded by "0x".

8.5.2. *CREAD addr [n]*

Read n number of consecutive 32-bit values from the V120/V124 control registers starting at “addr”, which is 32-bit aligned. Data is returned in hex.

```
V120> CR 0
0x0000FEEE
V120> CR 0 2
0x0000FEEE 0x00005668
```

8.6. *VME Region Access Commands*

Commands are provided to read and write the VME space. USB and Ethernet accesses do not use or modify the page descriptor registers.

8.6.1. *VMODE [A16/A24/A32/Mxx] [S0/S1/S2/S3]*

Set/query the VME address mode and speed.

The address modifier may be expressed as **A16 A24** or **A32** for the default supervisory access modes of 0x2D, 0x3D, and 0x0D respectively. Alternatively, arbitrary address modifiers may be requested as **Mxx**, where xx is the decimal value of the address modifier, 0-63. For example, **VMODE A24** is equivalent to **VMODE M61**, but to request non-privileged A24 data instead (0x39) would be **VMODE M57**

The speed setting is as described in section **4.3.1 Page Descriptors and VME Pages**, with 0 being the slowest and 3 being the fastest.

The default power-up condition is equivalent to **VMODE A16 S1**

8.6.2. *VWRITE BYTE/WORD/LONG addr v1 [v2 […]]*

Write consecutive values of BYTE, WORD, or LONG data size to VME space, starting at **addr** address, which is a 64-bit value that is aligned according to the data size selected. The values are decimal, or hex if preceded by "0x".

8.6.3. *VREAD BYTE/WORD/LONG addr [n]*

Read consecutive values of BYTE, WORD, or LONG data size from VME space, starting at **addr** address, which is a 64-bit value that is aligned according to the data size selected. Returned data is space-delimited hex.

If a Highland VME module were located at address 0xC000 in the A16 space:

```
V120> VMODE A16 S1
V120> VREAD WORD 0xC000
0xFEEE
```

Note that VME is big-endian, so no endian controls are needed for the read or write commands.

8.7. Ethernet Commands

Ethernet control and status serial commands can be performed over the Ethernet or USB ports.

8.7.1. *NETSTAT*

Get a brief summary of the Ethernet status.

```
V120> NETSTAT
HOSTNAME: V120-0000
MAC: 00:04:A3:02:EA:E8
LINK: 100
IP: 192.168.254.95 (DHCP)
SUBNET: 255.255.255.0
```

`IP:` is the actual/live IP address followed by either (DHCP) or (STATIC) to note whether the IP address was acquired statically or via DHCP.

8.7.2. *SUB [n.n.n.n]*

Set/query the Subnet mask. If DHCP is used, this value will be stored in memory but not used. Use the SAVE command to store this in non-volatile memory.

8.7.3. *IP [n.n.n.n]*

Set/query the reprogrammable hard IP address corresponding to Ethernet dipswitch selection code 0. The live IP address is instantly changed to the n.n.n.n value. Use the SAVE command to store this in non-volatile memory.

8.7.4. *SAVE*

Save the user IP address and Subnet mask into non-volatile memory.

8.7.5. *EXIT*

Exit gracefully from a TCP session. This has no effect on USB operation. It will close the TCP socket at port 2000 if it is open. Most telnet programs do not require this.

9. Startup Logic and Firmware Upgrade Procedures

The V120 is shipped with a factory code image, for both the microprocessor and FPGA code, stored in the on-board serial flash chip. Firmware can be updated by physical replacement of this plug-in chip. If users replace the flash chip, remove power and carefully remove the old chip, noting orientation and taking care to avoid bending pins. Plug in the new chip with the same orientation. Alignment dots are provided.

Users may also field-install code upgrades, over the PCIe, USB, or Ethernet links. If a valid upgrade is available, it will be loaded and run at power-up or restart time. An upgrade can be removed by a **FLASH ERASE** command, returning a unit to using factory code. The flash memory has 128 sectors. Sector 0...63 are the factory code, which can only be changed by physically replacing the flash chip. Sectors 64...128 can be field-loaded with an upgrade image.

If a unit is to be reflashed, Highland will provide a zip file with a name something like 22E120B_UPGRADE.ZIP. It will unzip into three files, 22E120B_UPGRADE.S28 (used for serial reflash over USB or Ethernet), 22E120B_UPGRADE.BIN (used for reflash over PCIe), and sha256sum.txt, which is a checksum file that can be used with the popular sha256sum utility to confirm validity of the two upgrade images.

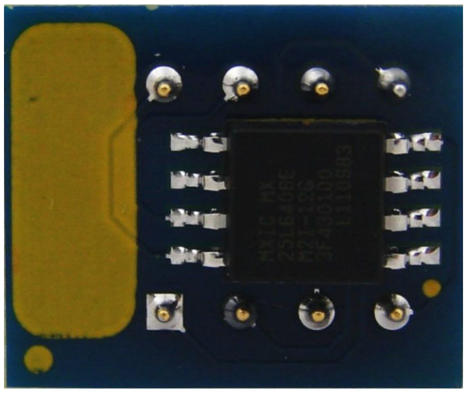
The V120 power-up or reset sequence is...

The FPGA is deconfigured and the red ERR LED is on solid. The orange USR LED is off, but will change state at each step below:

The microprocessor loads a baseline boot loader from the serial flash chip.

The bootstrap loader checks for the presence, and validity, of an upgrade code image. If this image is present and valid, it loads the upgrade image.

If there is no valid upgrade code, or the FPGA fails to configure from upgrade code, the default factory image is loaded. If there are any errors here, the boot loader hangs and makes long blinks on the red ERR LED. Otherwise, the factory code is run. The **STATUS** register summarizes the boot status.



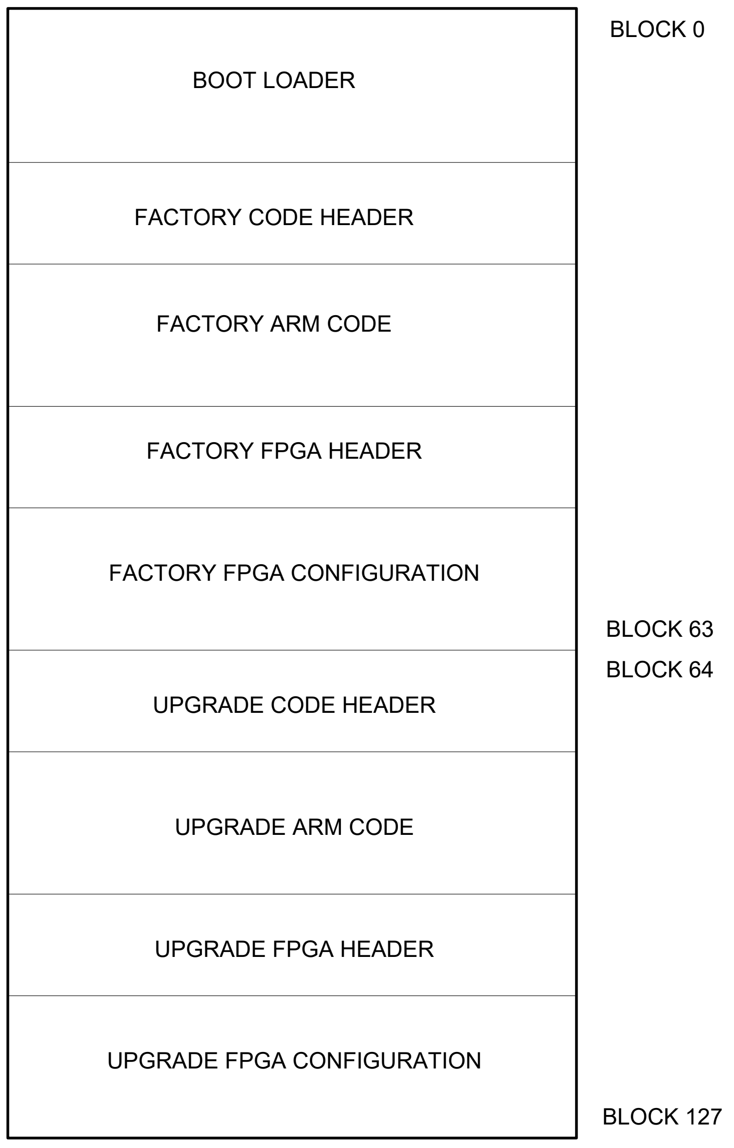
Plug-in serial flash chip

Serial Flash Memory Map

9.1. Serial Reflash Commands

A USB or Ethernet serial-command reflash session will look like the example below. The entire Highland-provided S28 text file is sent to the V120, one line at a time. The erase/reflash session might take about 10 minutes. The sequence is...

.



```
V120> FLASH UNLOCK
V120> FLASH ERASE
Erasing sector 64 of 127
Erasing sector 65 of 127
. . .
Erasing sector 127 of 127
V120> FLASH WRITE s28_line_1
. . .
V120> FLASH WRITE s28_line_n
V120> FLASH STATUS
<status report>
V120> RESET
```

The final RESET command reboots the controller and runs the new code.

9.1.1. *FLASH UNLOCK*

Unlock the serial flash. As a redundant measure the flash is locked at startup to prevent accidental use of the reflashing commands. Once unlocked, the flash will remain unlocked until reboot.

9.1.2. *FLASH WRITE srec*

Write a line of a Motorola-format S-Record to the “upgrade” portion of the flash. An invalid address in the S-Record, a failed checksum, or invalid S-Record format will result in an error. Use FLASH UNLOCK to enable this command.

9.1.3. *FLASH ERASE*

Erase the “upgrade” portion of the flash. Use FLASH UNLOCK to enable this command.

9.1.4. *FLASH STATUS*

This command returns a brief report identifying the factory and optional upgrade code images and their status. A typical report might look like

```
Highland V120-1A         SN 0007
Factory Code   22E120A   05-Jan-2013  OK
Factory FPGA   22C122A   05-Jan-2013  OK
Upgrade Code   22E120B   15-Mar-2013  OK    (or None or FAIL)
Upgrade FPGA   22C122B   15-Mar-2013  OK    (or None or FAIL)
Running  UPGRADE         FPGA  OK
```

9.2. PCIe Reflash Procedures

The flash memory ARM code and FPGA configuration can be reloaded using PCIe register operations, using data from the provided 22Exxxx_UPGRADE.BIN file. The associated operations use the **MACRO** register, macro parameters, and the **BUFFER** region.

9.2.1. *Flash Report Macro*

Writing 0x00000081 to the **MACRO** register returns a summary of the flash memory content. The text of the serial **FLASH STATUS** report will be deposited into the **BUFFER** region as ascii text, with a final null character.

9.2.2. *Reset Macro*

Macro 0x00000082 reboots the controller, as if it had a power down/up cycle.

9.2.3. *Erase Flash Macro*

Macro 0x00000083 erases the user-upgrade region of the flash memory. If the module is subsequently power cycled or reset, the original factory code will be run. The MACRO register will not clear until the entire region has been erased. During this time, the MP1 register can be probed for the number of the 64-kilobyte sector that is currently being erased. The sectors to be erased are numbered 64 to 127.

9.2.4. *Write Flash Macro*

Macro 0x00000084 writes to the user-upgrade region of the flash memory.

Highland will furnish a file named something like 22E120B_UPGRADE.BIN for field program upgrade. Before installing the upgrade, execute the **FLASH UNLOCK** and then the **FLASH ERASE** macros.

Once the upgrade region is erased, the upgrade install procedure is...

Set integer Z to (filesize/1024)

For K = 0 to Z-1

Copy the next (Kth) block of 1024 bytes (or fewer if the end-of-file is reached) from the .BIN file to the **BUFFER** space. Do this as 256 longword (32 bit) writes.

Write integer K to the **MP0** register

Write macro code 0x00000084 to the **MACRO** register

Wait for the low byte of **MACRO** to self-clear

Next K

Now reboot by writing the RESET code 0x00000082 to the **MACRO** register. The module will disappear from the VMEbus for several seconds.

Use the Flash Report Macro to verify that the upgrade is installed and working.

9.2.5. ***Flash Unlock***

Macro 0x00000086 enables write accesses to the flash memory. This must be executed before any flash programming operations.

10. Software and Drivers

Linux software support for the V120/V124 consists of three layers, the driver, the library, and the application software. The driver can be interfaced directly, but this can be complicated. The library provides simple wrapper functions around the driver, and is recommended for programmers writing their own application. Finally, the application software uses the library to perform basic tasks from the command line. This is the easiest way to get started, as well as serving as example code for the use of the library.

All Highland provided Linux software is provided under the BSD license, allowing incorporation in part or in full into proprietary applications. The driver is also dual-licensed under the GPL; end users modifying the driver may extend either or both licenses to their derived work.

The latest version of the V120 Linux device driver, libraries, and command-line tools can be downloaded from: https://github.com/highland-technology-inc

10.1. Device Driver

The base layer is the driver. For each connected V120/V124, the driver will create three files under `/dev`, specifically `/dev/v120_cX`, `/dev/v120_vX`, and `/dev/v120_qX` where X is the decimal value of the crate ID selected on the switch on the module, 0-15.

The first device `/dev/v120_cX` represents the BAR0 control region; and holds the page descriptor registers and the module control registers.

The second device `/dev/v120_vX`, is the mapped BAR1 VME space.

The third device `/dev/v120_qX`, is the interrupt endpoint.

10.1.1. ***Memory-mapped IO***

To interact with either the control or VME devices, first open a file descriptor to the device using the `open()` function. Then pass the file descriptor to the `mmap()` function to get a pointer to it. The driver is fully reentrant; multiple processes may access either device simultaneously.

10.1.2. ***Interrupts***

VME IRQs are translated into the blocking/non-blocking state of the `/dev/v120_qX` char device. This is compatible with either blocking `read` calls or `select/poll/epoll` techniques followed by a `read` that will not block. Note that `read` calls will return 0 bytes, the information is that an interrupt is available, not what that interrupt is. Regardless, the read must be performed to accept the driver's notification of a new interrupt; this clears an internal flag causing the file to block again until the next time interrupts are asserted.

Once an interrupt is asserted, the user software must clear **all** interrupts before another interrupt will emit from the endpoint. This is due to the nature of VME's level-sensitive and overlapping interrupts; only the user software can tell whether IRQ3 is still asserted because the interrupting module has yet to be serviced or because a second module is also asserting IRQ3. `v120irqd` provides a reliable mechanism for dealing with this.

10.1.3. *Streaming DMA*

Note: DMA is not yet implemented; this section is prospective and subject to change.

Streaming transfers are performed between user-allocated memory buffers and contiguous regions of memory in the V120. First, the streaming transfer must be configured by making an `ioctl()` call of type `V120_CONFIGURE_DMA`, with an argument for `DMA_READ`, `DMA_WRITE`, or `DMA_BIDIR`. This will provide the driver with information such as VME address modifier, endianness translation, etc. After this, a `read()` or `write()` call will perform a typical blocking operation. Alternatively, `aio_read()` and `aio_write()` calls can be used to perform a non-blocking operation with a callback.

The information provided by the ioctl will be held by the driver on a per-instance basis. The read and write calls will provide mutex locking that ensure the atomicity for the actual transfer. This means that multiple processes, each with their own handle for the crate, can safely perform streaming transfers with no collision concerns. If, however, multiple threads all hold the same file handle, keeping the ioctl programmed values from being changed before the associated DMA transfer has occurred is a concern that must be dealt with by the programmer.

10.1.4. *Permanent DMA*

Note: DMA is not yet implemented; this section is prospective and subject to change.

Preallocated transfers require that the buffer memory at the application level be requested through the driver, rather than by conventional `malloc()` calls. This allows the driver to provide a memory buffer that is locked to their initial location in physical memory; it cannot be either paged to disk or moved in memory. This buffers are requested through an `ioctl()` call of type `V120_ALLOC_PERMANENT`, with argument size (number of bytes in the buffer). Once a preallocated buffer has been created, DMA descriptor chains 0-15 can be assigned to it with a `V120_BUILD_PERMANENT ioctl()`. Eventually the DMA chain can be executed by making an `ioctl()` call of type `V120_EXEC_PERMANENT`, with arguments `bufno` (0-15) and `blocking` (true, false).

10.2. Library

The library provides a set of wrapper functions that greatly simplify working with the V120 driver. The functions are built into libv120.so, and user programs can be linked against it by using `#include <v120.h>` and compiling with the `-lV120` flag.

A high level interface is provided for working with VME regions. Typically, each region will represent a single VME module, or a single disjoint part of a module. Different regions can have different access control settings, such as transaction splitting, speed, and endianness, and the high-level interface will properly sort things out to ensure that each region is accessed correctly.

Using the high level interface, a programmer gets a handle to a given v120 using the `v120_open()` function, attaches a list of regions to that handle using `v120_add_vme_region()`, and then calls `v120_allocate_vme()`. For each region in the list, this generates a user space pointer to the first address of the requested region, allowing memory-mapped IO to be performed against it.

The library also performs a low-level interface, allowing individual page descriptors to be set and pointers to individual pages or page ranges to be retrieved by number.

See the documentation in v120.h for more information.

10.3. Application

The v120 application is a monolithic program for working with the V120/V124 from Linux command prompt. This software allows access to both the V120 internal registers and the VME space.

Commands are provided for reading and writing VME registers, querying the V120, resetting the V120, performing a VME SYSRESET, upgrading the V120 firmware, and more. Some examples include:

```
v120 report power
v120 flash-upgrade 22C120_upgradeB.bin
v120 write –a16 0xC000 0xDEAD 0xBEEF 0 0 0xFEED
v120 read –a24 –dwidth=l 0x8000 128
```

11. Versions

V120-1:	VME PCI express crate controller
V120-21:	VME PCI express crate controller with conformal coating
V124-1:	VXI PCI express crate controller
V124-21:	VXI PCI Express Crate Controller with conformal coating

12. Customization

Consult factory for information about additional custom versions.

13. Hardware and Firmware Revision History

13.1. V120 Hardware Revision History

Revision C	June 2018 Improved manufacturability. Functionality equivalent to Rev. B
Revision B	May 2014 Added ferrite beads to VME lines to prevent signal reflections on long backplanes.
Revision A	March 2013 Initial PCB release.

13.2. V124 Hardware Revision History

Revision B	June 2013 Added ferrite beads to VME lines to prevent signal reflections on long backplanes. Changed VXI CLK10 drive circuitry to reduce drive impedance.
Revision A	October 2012 Initial PCB release.

13.3. Firmware Revision History

22E120 Revision H	December 2020 Fixes a bug that may incorrectly set the PE bit in the STATUS register.
22E120 Revision G	October 2018 Updated code to accommodate new Ethernet PHY.
22E120 Revision F	January 2017 Added bus timer support when acting as a system controller for external VME masters. Reduced bus timeouts. Hardware revisions A-B require Highland to apply ECO1054 to support this functionality.

22E120 Revision E	November 2016 Added DMA, front panel monitor, and external clock support.
22E120 Revision D	June 2015 Added VME interrupt support.
22E120 Revision C	December 2013 Fixed a PCIe reset problem that could under some circumstances hang the V120 when the host PC was reset.
22E120 Revision B	October 2013 Added arbitration and support for VXI sideband signals.
22E120 Revision A	February 2013 Initial release.

14. Accessories

Host-end PCIe cable drivers, cables, and switches are commercially available in a variety of formats, and can be purchased either from Highland or through third-parties. Cabled PCI Express is usable over up to 7 meters of cable, and switch boxes can extend range and/or fan out one cable to multiple VME crates. Fiber optic PCIe extenders are available to extend the link up to 300m.

J4-2:	2-meter 4x PCI express cable
J4-3:	3-meter 4x PCI express cable
J4-4:	4-meter 4x PCI express cable
J4-5:	5-meter 4x PCI express cable
J4-7:	7-meter 4x PCI express cable
J120-1:	PC cable driver board - single 4-lane, for one crate
J121-1:	PC cable driver board - dual 4-lane, for two crates