

NAME

vme – VME interface

SUPPORTED CONTROLLER CARDS

spec supports the following VME controllers. Note, SBS (formerly Bit 3) is now GE Fanuc.

SBS (Bit 3) Model 403 ISA – Available on *Linux* and on HP-700 platforms with E/ISA slots. Only A16 access supported. No kernel drivers involved.

SBS (Bit 3) Model 616/617/618/620 PCI with SBS Driver – Available on *Linux* with the SBS model 1003 v1.0 driver, on Solaris platforms with the SBS model 945 driver and on HP platforms with the SBS model 934 driver. If the SBS controller is to be the master on the VME crate, be sure to set the system controller jumpers on the SBS VME module.

SBS (Bit 3) Model 616/617/618 PCI with no Driver – spec provides built-in user level support on *Linux* platforms, which may be useful on platforms with kernel versions not supported by the SBS driver. This built-in support doesn't include support for DMA transfers, though. If the SBS controller is to be the master on the VME crate, be sure to set the system controller jumpers on the SBS VME module.

SBS (Bit 3) Model 487-1 E/ISA – Requires SBS (Bit 3) Model 933 support software. DMA transfers supported. Note, you must use the HP *eisa_config* utility to change the default board settings to include "Memory Mapping Enabled" and "Privilege Checks Disabled".

SBS (Bit 3) Model 466-1 SBus – Requires SBS (Bit 3) Model 944 support software.

SBS (Bit 3) Model 467-1 SBus – Same as Model 466-1 above, but DMA transfers supported.

Struck Model SIS1100/3100 PCI – Available on *Linux* using either spec's built-in driverless support or using Struck's kernel driver. The built-in support doesn't include support for DMA transfers.

Struck Model SIS3150 USB – Available on *Linux* and Mac OS X platforms using spec's built-in support.

BUILT-IN FUNCTIONS

The type of data access and/or VME address modifier for the functions can be selected with the optional argument *opts* as follows (if more than one option is needed, make a comma-separated list in the single string argument):

"D8" – byte access.

"D16" – short-word access.

"D32" – long-word access, but only available with `vme_get32()` and `vme_put32()`.

"A16" – use A16 (amod=0x2D) addressing.

"A24" – use A24 (amod=0x3D) addressing, on adapters that support it.

"A32" – use A32 (amod=0x0D) addressing, on adapters that support it.

"DPRT" – use dual-port memory access (amod=0x1D), on adapters that support it.

"amod=0xxx" – specify the hexadecimal value for the address modifier.

The `vme_move()` function recognizes the following additional strings in its fourth argument:

"no_dma" – Don't use the DMA interface for this transfer, even if it is available.

"no_inc" – Rather than increment the VME address for each item transferred, write each item to the same VME address.

The default mode for the A16 access functions `vme_get()` and `vme_put()` is D8. The default mode for the A32 access functions `vme_get32()` and `vme_put32()` is D32. The defaults for all functions can be overridden by options in the *opts* string.

Not all VME adapters supported by `spec` support A32 access.

`vme_get(addr [, opts])` – Returns the data at `addr` in the 64K A16 address space.

`vme_put(addr, data [, opts])` – Writes `data` to `addr` in the 64K A16 address space.

`vme_get32(addr [, opts])` – Returns the data at `addr` in the A32 address space.

`vme_put32(addr, data [, opts])` – Writes `data` to `addr` in the A32 address space.

`vme_move(from, to [, cnt [, opts]])` – Copies data between a `spec` data array and VME address space. One of the `from` and `to` arguments must be the name of a `spec` data array while the other must be a VME address. If the optional argument `cnt` is present, it designates how many data items (not bytes) to copy. If missing or zero, the number of elements in the array is copied. A32 addressing is used by default. The default data size is determined by the data size of the array. Both can be overridden by `opts`.

MULTIPLE VME CONTROLLERS

`spec` allows up to four VME controllers to be configured by the same instance of `spec`. In the configuration editor, use the `^F` and `^B` command on the VME controller line of the **Interfaces** screen to configure each controller. To include a unit number in a VME address for a device, enter the address using the notation `unit:address`. An absent `unit:` prefix implies VME controller unit zero.

In the `config` file, the unit number associated with a VME controller is specified with `@vme_0`, `@vme_1`, etc.

For the `vme_get()`, `vme_get32()`, `vme_put()`, `vme_put32()` and `vme_move()` functions, the VME unit numbers are encoded using the same "unit:address" syntax as above. For the function calls, such an address argument must be passed as a string by using quotes to make a constant string, or by passing a string-valued variable.

SEE ALSO

www.gefanuc.com

www.struck.de