VME Device Drivers

Driver registration

As with other subsystems within the Linux kernel, VME device drivers register with the VME subsystem, typically called from the devices init routine. This is achieved via a call to :c:func:`vme_register_driver`.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 7);** *backlink*
>
> Unknown interpreted text role "c:func".

A pointer to a structure of type :c:type:`struct vme_driver <vme_driver>` must be provided to the registration function. Along with the maximum number of devices your driver is able to support.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 11);** *backlink*
>
> Unknown interpreted text role "c:type".

At the minimum, the '.name', '.match' and '.probe' elements of :c:type:`struct vme_driver <vme_driver>` should be correctly set. The '.name' element is a pointer to a string holding the device driver's name.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 15);** *backlink*
>
> Unknown interpreted text role "c:type".

The '.match' function allows control over which VME devices should be registered with the driver. The match function should return 1 if a device should be probed and 0 otherwise. This example match function (from vme_user.c) limits the number of devices probed to one:

```
#define USER_BUS_MAX    1
...
static int vme_user_match(struct vme_dev *vdev)
{
        if (vdev->id.num >= USER_BUS_MAX)
                return 0;
        return 1;
}
```

The '.probe' element should contain a pointer to the probe routine. The probe routine is passed a :c:type:`struct vme_dev <vme_dev>` pointer as an argument.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 35);** *backlink*
>
> Unknown interpreted text role "c:type".

Here, the 'num' field refers to the sequential device ID for this specific driver. The bridge number (or bus number) can be accessed using dev->bridge->num.

A function is also provided to unregister the driver from the VME core called :c:func:`vme_unregister_driver` and should usually be called from the device driver's exit routine.

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 43);** *backlink*
>
> Unknown interpreted text role "c:func".

Resource management

Once a driver has registered with the VME core the provided match routine will be called the number of times specified during the registration. If a match succeeds, a non-zero value should be returned. A zero return value indicates failure. For all successful matches, the probe routine of the corresponding driver is called. The probe routine is passed a pointer to the devices device structure. This pointer should be saved, it will be required for requesting VME resources.

The driver can request ownership of one or more master windows (:c:func:`vme_master_request`), slave windows (:c:func:`vme_slave_request`) and/or dma channels (:c:func:`vme_dma_request`). Rather than allowing the device driver to request a specific window or DMA channel (which may be used by a different driver) the API allows a resource to be assigned based on the required attributes of the driver in question. For slave windows these attributes are split into the VME address spaces that need to be accessed in 'aspace' and VME bus cycle types required in 'cycle'. Master windows add a further set of attributes in 'width' specifying the required data transfer widths. These attributes are defined as bitmasks and as such any combination of the attributes can be requested for a single window, the core will assign a window that meets the requirements, returning a pointer of type vme_resource that should be used to identify the allocated resource when it is used. For DMA controllers, the request function requires the potential direction of any transfers to be provided in the route attributes. This is typically VME-to-MEM and/or MEM-to-VME, though some hardware can support VME-to-VME and MEM-to-MEM transfers as well as test pattern generation. If an unallocated window fitting the requirements can not be found a NULL pointer will be returned.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 59); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 59); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 59); ***backlink***

Unknown interpreted text role "c:func".

Functions are also provided to free window allocations once they are no longer required. These functions (:c:func:`vme_master_free`, :c:func:`vme_slave_free` and :c:func:`vme_dma_free`) should be passed the pointer to the resource provided during resource allocation.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 78); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 78); ***backlink***

Unknown interpreted text role "c:func".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master] [Documentation] [driver-api]vme.rst`, line 78); ***backlink***

Unknown interpreted text role "c:func".

Master windows

Master windows provide access from the local processor[s] out onto the VME bus. The number of windows available and the available access modes is dependent on the underlying chipset. A window must be configured before it can be used.

Master window configuration

Once a master window has been assigned :c:func:`vme_master_set` can be used to configure it and :c:func:`vme_master_get` to retrieve the current settings. The address spaces, transfer widths and cycle types are the same as described under resource management, however some of the options are mutually exclusive. For example, only one address space may be specified.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 95); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 95); *backlink*

Unknown interpreted text role "c:func".

Master window access

The function :c:func:`vme_master_read` can be used to read from and :c:func:`vme_master_write` used to write to configured master windows.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 105); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 105); *backlink*

Unknown interpreted text role "c:func".

In addition to simple reads and writes, :c:func:`vme_master_rmw` is provided to do a read-modify-write transaction. Parts of a VME window can also be mapped into user space memory using :c:func:`vme_master_mmap`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 108); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 108); *backlink*

Unknown interpreted text role "c:func".

Slave windows

Slave windows provide devices on the VME bus access into mapped portions of the local memory. The number of windows available and the access modes that can be used is dependent on the underlying chipset. A window must be configured before it can be used.

Slave window configuration

Once a slave window has been assigned :c:func:`vme_slave_set` can be used to configure it and :c:func:`vme_slave_get` to retrieve the current settings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 125); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 125); *backlink*

Unknown interpreted text role "c:func".

The address spaces, transfer widths and cycle types are the same as described under resource management, however some of the options are mutually exclusive. For example, only one address space may be specified.

Slave window buffer allocation

Functions are provided to allow the user to allocate (:c:func:`vme_alloc_consistent`) and free (:c:func:`vme_free_consistent`) contiguous buffers which will be accessible by the VME bridge. These functions do not have to be used, other methods can be used to allocate a buffer, though care must be taken to ensure that they are contiguous and accessible by the VME bridge.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 136); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 136); *backlink*

Unknown interpreted text role "c:func".

Slave window access

Slave windows map local memory onto the VME bus, the standard methods for accessing memory should be used.

DMA channels

The VME DMA transfer provides the ability to run link-list DMA transfers. The API introduces the concept of DMA lists. Each DMA list is a link-list which can be passed to a DMA controller. Multiple lists can be created, extended, executed, reused and destroyed.

List Management

The function :c:func:`vme_new_dma_list` is provided to create and :c:func:`vme_dma_list_free` to destroy DMA lists. Execution of a list will not automatically destroy the list, thus enabling a list to be reused for repetitive tasks.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 163); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 163); *backlink*

Unknown interpreted text role "c:func".

List Population

An item can be added to a list using :c:func:`vme_dma_list_add` (the source and destination attributes need to be created before calling this function, this is covered under "Transfer Attributes").

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 172); *backlink*

Unknown interpreted text role "c:func".

Note

The detailed attributes of the transfers source and destination are not checked until an entry is added to a DMA list, the request for a DMA channel purely checks the directions in which the controller is expected to transfer data. As a result it is possible for this call to return an error, for example if the source or destination is in an unsupported VME address space.

Transfer Attributes

The attributes for the source and destination are handled separately from adding an item to a list. This is due to the diverse attributes required for each type of source and destination. There are functions to create attributes for PCI, VME and pattern sources and destinations (where appropriate):

- PCI source or destination: :c:func:`vme_dma_pci_attribute`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 193); *backlink*

  Unknown interpreted text role "c:func".

- VME source or destination: :c:func:`vme_dma_vme_attribute`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 194); *backlink*

  Unknown interpreted text role "c:func".

- Pattern source: :c:func:`vme_dma_pattern_attribute`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 195); *backlink*

  Unknown interpreted text role "c:func".

The function :c:func:`vme_dma_free_attribute` should be used to free an attribute.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 197); *backlink*

Unknown interpreted text role "c:func".

List Execution

The function :c:func:`vme_dma_list_exec` queues a list for execution and will return once the list has been executed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 204); *backlink*

Unknown interpreted text role "c:func".

Interrupts

The VME API provides functions to attach and detach callbacks to specific VME level and status ID combinations and for the generation of VME interrupts with specific VME level and status IDs.

Attaching Interrupt Handlers

The function :c:func:`vme_irq_request` can be used to attach and :c:func:`vme_irq_free` to free a specific VME level and status ID combination. Any given combination can only be assigned a single callback function. A void pointer parameter is provided, the value of which is passed to the callback function, the use of this pointer is user undefined. The callback parameters are as follows. Care must be taken in writing a callback function, callback functions run in interrupt context:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 219); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 219); *backlink*

Unknown interpreted text role "c:func".

```
void callback(int level, int statid, void *priv);
```

Interrupt Generation

The function :c:func:`vme_irq_generate` can be used to generate a VME interrupt at a given VME level and VME status ID.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 235); *backlink*

Unknown interpreted text role "c:func".

Location monitors

The VME API provides the following functionality to configure the location monitor.

Location Monitor Management

The function :c:func:`vme_lm_request` is provided to request the use of a block of location monitors and :c:func:`vme_lm_free` to free them after they are no longer required. Each block may provide a number of location monitors, monitoring adjacent locations. The function :c:func:`vme_lm_count` can be used to determine how many locations are provided.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 249); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 249); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 249); *backlink*

Unknown interpreted text role "c:func".

Location Monitor Configuration

Once a bank of location monitors has been allocated, the function :c:func:`vme_lm_set` is provided to configure the location and mode of the location monitor. The function :c:func:`vme_lm_get` can be used to retrieve existing settings.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 259); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 259); *backlink*

Unknown interpreted text role "c:func".

Location Monitor Use

The function :c:func:`vme_lm_attach` enables a callback to be attached and :c:func:`vme_lm_detach` allows on to be detached from each location monitor location. Each location monitor can monitor a number of adjacent locations. The callback function is declared as follows.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 268); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 268); *backlink*

Unknown interpreted text role "c:func".

```
void callback(void *data);
```

Slot Detection

The function :c:func:`vme_slot_num` returns the slot ID of the provided bridge.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 281); *backlink*

Unknown interpreted text role "c:func".

Bus Detection

The function :c:func:`vme_bus_num` returns the bus ID of the provided bridge.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 287); *backlink*

Unknown interpreted text role "c:func".

VME API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 293)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: include/linux/vme.h
   :internal:
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\driver-api\[linux-master][Documentation][driver-api]vme.rst, line 296)

Unknown directive type "kernel-doc".

```
.. kernel-doc:: drivers/vme/vme.c
   :export:
```