96.2.8 YY

Usage Guide of UNIX VME Library for General Purpose VME I/O Device Driver Version 1.0

Y.Yasu(KEK On-line Group)

Contents

Introduction 1
How to Compile and Link 1
General I/O 2
vme_mapopen 2
vme_mapclose 2
vme_mapprobe 3
Interrupt Handling 3
vme_intopen 3
vme_intclose 4
vme_intenable 4
vme_intdisable 4
vme_intwait 4
vme_intnowait 5
vme_intclear 5
vme_intcheck 5
Return Status of Interrupt Handling 6
List of Available Computers 7

Introduction

The mapping function and interrupt handling function for VMEbus are described. The routines are provided for doing test of VME module without making the special device driver for the VME module. DMA function is not supported in the routines.

The method of the mapping to VMEbus and the interrupt handling are usually different in each computer system. The routines absorbs the difference and thus you can make system-independent program by using the routines.

The computers available are shown in the section "List of Available Computers".

How to compile and link

It is simple to compile and link a user's program using the VME library. The following examples show how to compile and link the program.

Define the environmental variables.
First, VME_ROOT means the root directory for the VME library. For example;
% setenv VME_ROOT /usr/local/vme/gendev

Second, CC means the C compiler. The parameter depends on the Operating system. For HP-RT, the parameter should be "ccrt".

% setenv CC cc

Finally, compile and link the program.

```
%  $CC example.c -o example -I$VME_ROOT/include \
     -L$VME_ROOT/lib -lvmelib
```

General I/O

vme_mapopen

Name **:**

vme_mapopen for mapping the VME address space.

Syntax :

<C>

```
#include "vmelib.h"
virtualaddr = vme_mapopen( mode, vmeaddr, size )
int mode;
volatile caddr_t vmeaddr, virtualaddr;
size_t  size;
```

Arguments :

mode	select address/data space. VME_A16D16MODE VME_A16D32MODE VME_A24D16MODE VME_A24D32MODE VME_A32D16MODE VME_A32D32MODE
vmeaddr	starting VME physical address of VME module to be mapped
size	size of the VME address to be mapped

Description :

This routine maps the VME address space to the process address space. The return value is a mapped virtual address to VMEbus. VME physical address to be available is defined at installation of VME device drivers. See the description of available VME address space.

Return Value :

Upon successful completion, the mapped address is returned.
When the call fails, the value -1 is returned.

vme_mapclose

Name **:**

vme_mapclose for unmapping the VME address space.

Syntax :

<C>

```
#include "vmelib.h"
vme_mapclose( addr, mode )
caddr_t  addr;
int mode;
```

Arguments :

addr	the mapped address
mode	selected address/data space at calling vme_mapopen

Description :

This routine unmaps the VME address space.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_mapprobe

Name **:**

vme_mapprobe for checking whether the VME address can be read or not.

Syntax :

<C>

```
#include "vmelib.h"
vme_mapprobe( addr, size, mode )
volatile caddr_t  addr;
size_t  size;
int  mode;
```

Arguments :

addr the mapped address
size size of the mapped address
mode select address/data space.
VME_A16D16MODE
VME_A16D32MODE
VME_A24D16MODE
VME_A24D32MODE
VME_A32D16MODE
VME_A32D32MODE

Description :

This routine checks whether the VME address space can be read or not. It is useful to check the read access of the VME address space. The addresses to be checked should be contiguous. If the routine probes the non-existing memory, the process calling the routine will abort.

Interrupt Handling

Those routines are used for handling interrupts from VMEbus. The routines don't assume specific hardware equipment, but require well-defined vector number and number of IRQ level. By default, the routines assume the number from F0 to F9 in hex. as vector number and the number 3 as IRQ level because those values should be defined at building UNIX kernel.

vme_intopen

Name **:**

vme_intopen for opening the specified interrupt vector.

Syntax :

<C>

```
#include "vmelib.h"
vme_intopen( vector, intrno )
int vector, *intrno;
```

Arguments :

vector vector number to be opened. By default, F0 - F9 in hex. are valid
intrno ID related to the vector number. By default, 0 - 9 will be returned.

Description :

This routine opens the specified interrupt vector and gets the ID. The ID is used by other routines for identifying the vector number.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intclose

Name **:**
vme_intclose for closing the specified interrupt vector.
Syntax :
<C>
```
#include "vmelib.h"
vme_intclose( intrno )
int intrno;
```
Argument :
intrno interrupt number corresponding to the vector number
Description :
This routine closes the interrupt vector corresponding to intrno.
Return Value :
Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intenable

Name **:**
vme_intenable for enabling the interrupt.
Syntax :
<C>
```
#include "vmelib.h"
vme_intenable( intrno )
int intrno;
```
Argument :
intrno interrupt number corresponding to the vector number
Description :
This routine enables the interrupt .
Return Value :
Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intdisable

Name **:**
vme_intdisable for disabling the interrupt.
Syntax :
<C>
```
#include "vmelib.h"
vme_intdisable( intrno )
int intrno;
```
Argument :
intrno interrupt number corresponding to the vector number
Description :
This routine disables the interrupt .
Return Value :
Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intwait

Name **:**
vme_intwait for waiting the interrupt.
Syntax :
<C>
```
#include "vmelib.h"
vme_intwait( intrno, interrupt_count, timeout )
int intrno, *interrupt_count, timeout;
```

Arguments :

intrno	interrupt number corresponding to the vector number
interrupt_count	the count of interrupt already occurred is returned.
timeout	time-out time in seconds

Description :

This routine waits for the interrupt . If an interrupt occurs, interrupt counter will be incremented. Then, the routine is called, the interrupt counter is reset. while the value of the counter is returned. The behavior of the routine is described in the "Return Status of Interrupt Handling" section.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intnowait

Name **:**

vme_intnowait for checking and reporting the interrupt.

Syntax :

<C>

```
#include "vmelib.h"
vme_intnowait( intrno, interrupt_count )
int intrno, *interrupt_count;
```

Arguments :

intrno	interrupt number corresponding to the vector number
interrupt_count	the count of interrupt already occurred is returned.

Description :

This routine checks whether the interrupt comes in or not. and reports it. If an interrupt occurs, interrupt counter will be incremented. When the routine is called, the value of the interrupt counter is returned and then the counter is reset. The routine doesn't wait for any interrupt.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intclear

Name **:**

vme_intclear for clearing the interrupt.

Syntax :

<C>

```
#include "vmelib.h"
vme_intclear( intrno )
int intrno;
```

Argument :

intrno interrupt number corresponding to the vector number

Description :

This routine clears the interrupt counter.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

vme_intcheck

Name **:**

vme_intcheck for checking and reporting the interrupt.

Syntax :

<C>

```
#include "vmelib.h"
vme_intcheck( intrno, interrupt_count )
```

int intrno, *interrupt_count;

Arguments :

intrno	interrupt number corresponding to the vector number
interrupt_count	the count of interrupt already occurred is returned.

Description :

This routine checks whether the interrupt comes in or not. and reports it. The interrupt counter is not reset.

Return Value :

Upon successful completion, value 0 is returned.
Otherwise, the call fails. The value of the global variable "errno" is returned.

Return Status of Interrupt Handling Routines

We consider the return value of vme_intwait routine. We can think 4 cases the routine returns the value. In one of them, the routine will return a value, zero when routine is called and then an interrupt occurs. In another case, the routine will return a value, zero when an interrupt already occurred and then the routine is called. In third case, the routine will return a value, ETIMEDOUT when the routine is called but an interrupt doesn't occur, that is, the time-out occurs. In last case, the routine will return a value, EINTR when the routine is called and then a signal occurs before an interrupt occurs.

List of Available Computers

The VME library is available on the following computers;

DECstation5000/Ultrix with DEC VMEbus adaptor(PMABV/T6000)
HP9000/74x(VME board computer)/HP-RT

"o" means "available";

Computers	General I/O	Interrupt Handling
DECstation /Ultrix]	O	O
HP74x/ HP-RT	O	O