PS/CO-Note 92-22

15 October 1992

Equipment Modules using the VME Bus

J.Cuperus, W.Heinze, Cl.H.Sicard, U.Reich

Introduction

Most of our equipment modules (EM) use the CAMAC bus. New equipment now is often controlled from the VME bus. A problem arises when, for the same EM, there is some equipment controlled by CAMAC modules and some by VME modules. The EM may be very different and it is useful to define them separately. On the other hand, the operator wants to see a common EM and is not interested in the particularities of the bus.

EM Naming Convention

Whenever there are two different versions of the EM, we will define two EMs: 'EMNAME' for the old CAMAC version and 'EMNAME-V' for the new VME version. The operator will see both under the name 'EMNAME'. Later, when all the CAMAC equipment has disappeared, we will again have a single implementation of the EM, known everywhere under the name 'EMNAME'.

In the case of power converters, for example, the EM will be defined as 'POW' for CAMAC and as 'POW-V' for VME. The operators will access both versions under the name 'POW'.

EM Definition

EMNAME and EMNAME-V are defined separately with the genmod facility. They can have a different superclass, if useful, but ***must*** have the same CLASSNO (3 for POW). Both versions must share essential properties but may implement them with different procos. They may differ in auxiliary properties, related to the bus hardware.

There is a new field in the EM definition form: BUS. This field has the value 'C' for the CAMAC version and 'V' for the VME version. It is automatically initialised to 'C' for all existing EM definitions.

RO Class and Instance Variables

In ORACLE tables EQUIPMENT and INSTVAL, all equipment is listed under its complete EM name: EMNAME or EMNAME-V.

Read-only class and instance variables are filled in, as before, with the genmod facility. The classname is EMNAME or EMNAME-V. The variable names and the values may be different for both versions.

PROCO Naming Convention

For procos which are not general purpose, we follow the old naming convention with names of the form RnnXXXXX for read procos and WnnXXXXX for write procos, where nn is the CLASSNO and X stands for any character (e.g. R03CCVX and W03CCVC for POW). We make no distinction between versions as the same proco may well serve both versions in some cases.

PROCO Libraries

The genmod facility can generate templates for all special purpose procos (with the class number in them) used by the EM version. The most practical way for using this template is as the second file in the RAND editor, from which you can copy sections in your working file. The template contains only the procos of a single version but the proco file should contain the procos of both versions (suitably documented), so as to prevent duplication of certain common procos.

Bus Addresses

Bus addresses are important, not only for the proper functioning of the EM but also for maintenance and general system documentation. Therefore we need to store these addresses in standard read-only instance variables: ADDRESS1 to ADDRESS9.

For the CAMAC bus this is straightforward because there is a standard driver for all modules, which accepts crate, station, and register numbers as parameters.

For the VME bus, the format for the addresses is: **device_address /subaddress**, where

device_address	a number (0..65535) characterizing the address of the device on the VME bus via which the equipment is connected,
subaddress	subaddress or channel number in range 0..65535.

The data base transmits the addresses to the property code writer in the integer data columns ADDRESSi with following encoding:

bit 31..16 (MS word):	device_address,
bit 15..0 (LS word):	subaddress.

This definition is deliberately very general to cover all possible cases of accessing VME modules, especially accessing them via a driver or via direct memory mapping. The general philosophy is that the addressing should be most transparent for the user. The meaning of both numbers depends on the kind of VME modules (identified by class variables MODUL1 to MODUL9). To illustrate this, we give a few examples:

- For MIL1553 connected equipment, ADDRESS1 could contain: 1/43. The device address would be the bus controller number (1) and the subaddress the RTI number (43).

- An example for direct access could be an ICV 196 card. In that case ADDRESS1 could contain: 80/3, where 80 is the MS byte of the base address of the module (500000 hex) and 3 the group of I/O registers connected to the equipment.

The exact meaning of both numbers must be described in the equipment's documentation. The correspondence of the address demanded by a driver and the real VME base address of the module is given in the descriptor file of the driver.

If only the device_address is given, it will be encoded in the MS 16 bits of ADDRESSi, the LS 16 bits being zero. E. g., 3 would give 00030000 hex which is equivalent to 3/0.

Remarks:

- Most modules need only one VME address and this should be in ADDRESS1. If more addresses are needed, the driver needs the implicit knowledge of which addresses belong to what hardware features. A convention could be: if a module needs 2 addresses (like the MPV908), the first should be the one in the shorter address space and the second the one in the longer address space. Another implicit knowledge of the driver is the data width and addressing length.
- Many modules do not need a channel number. Normally, this number would signify something like the channel of an ADC converter. However, for 1553 controlled equipment, the channel number is the address of the specific equipment on the bus 1553.
- If one limits the device_address for direct addressing to a number 0..255 (the MS byte of the address), the minimum address space corresponding to any module ("granularity") is fixed. It is 256 bytes for short addressing, 64 Kbyte for standard addressing, and 16 Mbyte for extended addressing. This seems to be sufficient, leaving 256 different addressing possibilities for each of the 3 classes of modules.
- There is one specific remark to make for the STRUCK STR750 flash ADC system. The modules have standard address space but 32 bits data width. The LynxOS on our processor board MVME147S does not allow drivers to operate on standard addressing with 32 bits data width (however, a direct user can open memory space corresponding to this access modality). The driver must convert standard addressing to extended addressing, the modules accept this by a jumper setting corresponding to extended addressing. However, in that DSC, no extended addressing of another module is permitted since it would overlap with the STR750 modules.