MBS ESONE Server for Wiener VC32-CC32 CAMAC Controller

N. Kurz

September 21, 2004

GSI, Gesellschaft für Schwerionenforschung mbH
Planckstraße 1, D-64291 Darmstadt
Germany
Tel. (0 6159) 71–0
Contents

1  MBS ESONE Server for VC32-CC32 ................................. 3
   1.1  General Remarks ........................................... 3
   1.2  VC32-CC32 Overview ........................................ 3
   1.3  ESONE Utilization with VC32-CC32 .......................... 4
   1.4  VC32-CC32 in MBS User Readout Functions ................... 5
       1.4.1  Setup Parameters ..................................... 5
       1.4.2  Hints for User Readout Function Development .......... 5
Chapter 1

MBS ESONE Server for VC32-CC32

1.1 General Remarks

This note can be retrieved as pdf file from the web page http://daq.gsi.de/document.htm

In case of problems or questions please mail to n.kurz@gsi.de or call Tel: 2979 at GSI.

The Wiener VC32-CC32 CAMAC controller can be utilized in MBS user readout functions since about 3 years. Now it is also supported in the MBS ESONE CAMAC server (mesone_serv) for the current MBS production version 4.3 and the development version 4.4.

This note doesn't describe the functionality of the VC32-CC32 system. Please read the manual supplied by the company WIENER for detailed information. Instead a prescription will be given how to set up a system with VC32-CC32 controllers for the MBS ESONE server and for MBS data acquisition systems.

1.2 VC32-CC32 Overview

The VC32-CC32 CAMAC controller consists of a VME interface module (VC32) and a CAMAC controller (CC32), which are connected via a SCSI like cable of about 5m length at maximum. Only one CC32 can be connected to a VC32, but many VC32, and therefore many CC32, can be operated from a single VME crate. The readout of many VC32-CC32 in one VME crate is always sequential.

CAMAC access is achieved through VME access in VME A24 addressing mode. Apart from standard CAMAC it allows also Fast CAMAC Level 1 and broadcast writes. Both features are not provided by the various other CAMAC controllers supported within the MBS.

The VME A24 base address of the module can be set with jumpers on the VC32. Each VC32 needs a unique VME address window of size 0x8000 (32KB). If more than one VC32 will be used in a single VME crate it is required to set the VME base address of all VC32 with VME address spacing of 0x10000, i.e; first VC32: 0x550000 (factory setting), second VC32: 0x560000, third: 0x570000, ...
1.3 ESONE Utilization with VC32-CC32

Within the addressing scheme of the VC32-CC32 there is in principle nothing like a crate number which can be set on other CAMAC controllers. Instead the VME A24 base address defines the CAMAC crate to be addressed. On the other hand for CAMAC accesses in the ESONE scheme at least the crate nr. (C), the station nr. (N), the sub-address (A) and the function (F) has to be specified to compose a CAMAC access pointer (CNAF).

To achieve this the VME A24 base address has to be used implicitly as crate number. REM\_ESONE\_BASE is one of two MBS setup parameters to be specified in the MBS setup.usf file for the esone server (m_esone_serv). It stands for remote esone base address, where the CAMAC access shall be done in a remote crate. The second setup parameter is CONTROLLER\_ID, which can be specified for each CAMAC crate to be accessed. It describes the type of CAMAC controller in a specific CAMAC crate.

For the VC32-CC32 the ESONE server assumes the following bijective VME A24 address ↔ crate number mapping: Adjacent CAMAC crate numbers differ by 0x10000 in the VME A24 base address.

**Example:**

<table>
<thead>
<tr>
<th>VC32 A24 VME address</th>
<th>Crate Number</th>
<th>RIO2 Address</th>
<th>RIO3 Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x550000</td>
<td>↔ 1</td>
<td>0xee550000</td>
<td>0x4f550000</td>
</tr>
<tr>
<td>0x560000</td>
<td>↔ 2</td>
<td>0xee560000</td>
<td>0x4f560000</td>
</tr>
<tr>
<td>0x570000</td>
<td>↔ 3</td>
<td>0xee570000</td>
<td>0x4f570000</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>0x630000</td>
<td>↔ 15</td>
<td>0xee630000</td>
<td>0x4f630000</td>
</tr>
</tbody>
</table>

**REM\_ESONE\_BASE:** In the above example REM\_ESONE\_BASE has to be set to 0xee540000 for a RIO2 as VME controller and to 0x4f540000 for a RIO3 as VME controller (on a RIO2 0xee000000 maps to VME A24 address 0x0 and on RIO3 0x4f000000 maps to VME A24 address 0x0). The MBS ESONE server calculates the base address for each VC32 by REM\_ESONE\_BASE + 0x10000 * CrateNumber.

**CONTROLLER\_ID:** has to be set for each VC32 controller (each crate number) used in the prescribed scheme to 11. This hides all technical details, like where to get X and Q response, for the user.

**VME Address Sliders:** The VME address sliders of the VC32 have to be set to 0x550000 to be CAMAC crate 1, 0x560000 to be crate 2, and so on in the example above.

With REM\_ESONE\_BASE, CONTROLLER\_ID and the VME base addresses sliders set correctly on the VC32 the user can now issue the m\_esone\_serv CAMAC CNAF command in the usual manner, to work interactively with CAMAC. The remote CAMAC client library, functions, programs can be used unchanged as before since all VC32-CC32 technical details are handled by the m\_esone\_serv MBS program.
1.4 VC32-CC32 in **MBS** User Readout Functions

If the VC32-CC32 is used in a **MBS** data acquisition 5 sets of parameters have to be set correctly in the **MBS** setup.usf file and a user readout functions has to be developed.

A runnable **MBS** setup and user readout function can be found in the template directories `/mbs/v43/exa/rio3_vc32cc32` or `/mbs/v44/exa/rio3_vc32cc32`.

1.4.1 Setup Parameters

**REM_CAM_BASE**: In the REM_CAM_BASE array the VME A24 base addresses of the VC32 in use have to be specified as a function of the crate number. In principle any base address is valid if it correctly reflects the VME hardware setting of the VC32. It is required to use the same setting as described in the previous section to get the ESONE functionality as well. As usual the parameter values for RIO2 and RIO3 are different although pointing to the same VME address. The values recommended can be found in the table in the previous section in the columns named RIO2 Address and RIO3 Address.

**REM_CAM_OFF**: REM_CAM_OFF must be set to 0.

**REM_CAM_LEN**: REM_CAM_LEN must be set to 0x10000 for all VC32 (CAMAC crates) in use.

**CONTROLLER_ID**: CONTROLLER_ID must be set to 11 for all VC32 (CAMAC crates) in use.

**RD_FLG**: RD_FLAG must be set to 1 for all crates to be readout in the user readout function on an external trigger. For each crate with RD_FLAG set to 1 the user readout function is called and a sub-event is produced.

1.4.2 Hints for User Readout Function Development

Only a few important items will be addressed here. For a comprehensive view please see the **MBS** manual and the template f_user.c function in the directories given above.

**CNAF Coding**: The coding of the CAMAC CNAF (crate nr. (C), station nr. (N), sub-address (A) and function (F)) follows the same principles, but is different compared to the coding of the CBV and GTBC controllers developed at GSI. Therefore the template f_user.c contains a macro named CNAF, which shall be used to code pointers for CAMAC access. The CNAF macro for VC32-CC32 controller is:

```
#define CNAF(c,n,a,f,offset) \ (long *) (0x0*c + (n<<10) + ((f & 0xf)<<2) + (a<<6) + (long)offset)
```

**CAMAC access pointer construction**: The **MBS** delivers in the pointer array pl_rem_cam[] as a function of the crate number the virtual base pointer which corresponds to the hardware address specified in the REM_CAM_BASE setup array. With the help of the CNAF macro and
pl\_rem\_cam[] it is easy to get access pointers for all CAMAC actions.

Example C=2,N=7,A=1,F=16: pl\_camac = CNAF(2,7,16,pl\_rem\_cam[2])

**CAMAC X and Q Checking:** CAMAC X of a CAMAC action just executed can be found as data bit nr. 2 (counting bits from 0) after a pseudo CAMAC read for N=0, A=0, F=0. CAMAC Q is bit nr. 3 in the same data word. See VC32-CC32 manual.

**Mixing CAMAC Read and Write:** CAMAC read (F=0,2) and write (F=16) calls to an identical register use different addresses due to different CAMAC functions F. The processor regards these as independent and interchanges sometimes read and write calls to gain performance. Since the exact sequence of CAMAC actions is most of the time important (for example checking X and Q after a CAMAC write shall never be exchanged) the user must call the inline assembler

asm volatile ("eieio");

for all PowerPC processors (RIO2, RIO3) and

asm volatile ("nop");

for all 68K (E7) processors between CAMAC actions which shall not be exchanged.
Contents