The Wiener VME-USB controller for data acquisition: A status report

Kris Hagel and Brian Roeder

This year, we have developed and commissioned a new data acquisition system for the Cyclotron Institute based on the Wiener VME-USB control module, model VM-USB [1]. The VM-USB replaces the SBS controller and CES model CBD-8210 trigger module that were previously used to control the data acquisition system. This upgrade was necessary because the CES trigger module is no longer manufactured. The VM-USB was chosen because of its flexibility, autonomous data acquisition capabilities, and USB 2.0 computer interface. For most of the data acquisition used in the lab, these features will allow the development of systems with less acquisition deadtime. However, migration to the new system will require users to move away from the data acquisition with CAMAC modules as a similar VME-CAMAC interface has not yet been implemented in our system.

The VM-USB controller has been integrated into the existing data acquisition software at the Cyclotron Institute. Separate protocols in the "Transport Manager" program allow the user to choose at startup either the VM-USB or the previous SBS controller for backwards compatibility. It can be used currently with an assortment of VME electronics modules, including the Mesytec 32-channel VME ADCs, CAEN 128-channel TDC V1190, CAEN 16-channel dual-range QDC V965, SIS 3820 32-channel VME scaler and other VME modules.

The user will notice a few changes in the frontend and backend DAQ programming for the VM-USB when compared with the SBS controller. First, to take advantage of the data transfer rates of the USB 2.0 interface with the frontend control computer, the VM-USB uses a set of user-defined command stacks to carry out the readout of the data event-by-event. These command stacks are set up by the user in the "Frontend::Init()" method such that the modules are read out event-by-event in the order that they are defined in the Init() method. Once the command stack and readout order is defined, the VM-USB is triggered to execute the command stack (i.e. read out the data) by a NIM trigger pulse going into the front panel LEMO input "I1". During the data digitization and readout process, an inhibit NIM logic signal is output from the "O1" LEMO connector on the VM-USB module. The data read out in that event are stored in an onboard data buffer on the VM-USB that can hold up to 28 kB of data. When the data buffer is full, the VM-USB dumps the stored information, which may contain many individual events, to the control computer where it can be analyzed as online data and/or recorded to a file. Converting, reading out, and transferring data in this way have been observed to improve the deadtime of the DAQ system by about ~3x for a typical experimental setup (e.g. 50 μs for the VM-USB versus 150 μs for the same setup with the SBS controller). The remaining deadtime of the DAQ arises mostly from the conversion time of the DAQ modules.

In the TAMU DAQ software, the readout commands for a given VME module are added to the command stack in the "Frontend::Init()" method by using a C++ class constructor similar to the following (for the Mesytec ADC):

*CycMesytecMadc32Module*Adc = new CycMesytecMadc32Module(**GetVmeUsbStack(0)**, VMEaddress)*

where the readout commands are added to the event-by-event readout stack “0” with the function “GetVmeUsbStack(0)”. If a given VME module is to be read out event-by-event, its commands should be added to stack “0”. Once created, the module can be set up with the same initialization commands that were available when the SBS controller was used. In general, a readout delay should be also be added to the stack with the function “SetReadoutStartDelay(x)” (where “x” is the number of microseconds to wait) such that the readout of the VME modules does not begin before the digitization is completed. The amount of time to wait depends on the number of channels and/or number of modules to be read out. By contrast, a module that is initialized in the “Frontend::Init()” method, but should not be read event-by-event, should be defined as being added to stack “-1”. An example of this type of module is the CAEN V812B VME CFD, which is declared in the “Frontend::Init()” method as follows:

CycCaenV812BModule Cfd = new CycCaenV812BModule(**GetVmeUsbStack(-1)**,VMEaddress).*

The last type of module to be read out would be a scaler such as the SIS 3820 VME scaler. While this module can also be read out event-by-event by adding it to command stack “0”, typically a timed readout of this module is desirable, such as, every 10 seconds. In this case, the module should be added to stack “1”. In addition, the command “SetScalerReadFrequency(x)” (where “x” is the number of seconds) is invoked to set the time in seconds between scaler readouts, as before in the SBS DAQ system. Up to 8 command stacks can be programmed on the VM-USB, but to date only stacks “0” and “1” are implemented. The other stacks operate on interrupt vectors and will be implemented as needs arise.

Once the command stack “0” is defined, no further functions need to be added to the “Frontend” class for the “Transport Manager”. While the functions “Frontend::Event()”, “Frontend::Begin()”, “Frontend::End()” and “Frontend::ReadScalers()” still exist in the Frontend class, VME commands defined inside these methods are ignored because the readout of the modules is controlled by commands in the “stack”. For this reason, is not possible to observe the raw data being read out from the VME modules in the Frontend part of the program as was available with the SBS VME controller. If the user wishes to observe the raw digitized data from the VME modules, one must unpack the data in the backend analysis part of the DAQ code and display the raw data from the data buffer there.

For the analysis part of the TAMU DAQ system, the data buffers from the VM-USB are unpacked online or from the data file by the “Analysis Manager” program in the same way as they were from the SBS controller. To differentiate data coming from SBS controller type data files and VM-USB type data files, the VM-USB event-by-event data buffers are recorded with EventType=6 and EventSubType=1, and the VM-USB scaler data buffers are recorded with EventType=6 and EventSubType=2. However, the user should be careful to unpack the data as defined by the readout data word order for a given module. For example, the data word order for the Mesytec VME ADC begins with a “header” word, followed by “data” words containing digitized data from channels that were above the ADC threshold and then finally a “trailer” finishes the data word from the ADC for that event. Further information about the data word unpacking is often available in the module manual. This approach

differs from the SBS VME controller because the user could, in principle, choose only to record "important" data words and discard the rest.

The TAMU DAQ system with the VM-USB controller was tested in an experiment in June 2010. In that experiment, ^{37}K was produced with MARS with the p(^{38}Ar,^{37}K)2n at 29 MeV/u. To compare the two systems, the ^{37}K production rate was measured in two separate runs: one with the SBS controller and an existing DAQ control computer, and one with the VM-USB controller and a new DAQ control computer with a USB 2.0 input. All other parts of the readout electronics were the same for the two runs. In this simple test, the same ^{37}K production rate of ~500 eV/nC was measured for both runs. This indicated that the two DAQ systems gave the same results for comparable measurements. Following this test run, the VM-USB has been successfully used as the DAQ VME controller in several experiments.

In conclusion, the Wiener VM-USB control module has been integrated into the TAMU DAQ system and the TAMU DAQ software. It has been used to read out most of the existing VME DAQ modules at the Cyclotron Institute. The improvements in acquisition readout time and low cost make the new control module a suitable replacement for the old control system that utilized the SBS VME control module.

[1] For more information about the Wiener VM-USB controller, see the manual at:
http://www.**wiener**-d.com/Support/XX**USB**/Manual_**VM-USB**_3_3.pdf.