VME in CMS

C. Schwick

22nd October 2005

Table of contents

Introduction

These pages contain information on the use of VME in CMS. In particular they server the following purposes:

- Maintain CMS wide design rules for VME interface designers. These make it possible to create a homogenous software environment to access VME modules in the CMS experiment.
- Maintain information on the PCI-to-VME bridges used in CMS. This includes development status information, lists of bugs or shortcomings in either sofware or hardware components, and some performance measurements.
- The pages contain a set of drivers for the PCI-to-VME bridges supported by the CMS online group. The drivers are tested or if necessary modified so that they fit into the CMS online environment. They are compatible with the CMS Hardware Access Library (HAL) (http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL) and are contained in the CMS online software.

VME64x in CMS

CMS has decided to use VME64x modules. Of the VME64x specification those features should be implemented which allow to perform a plug and play configuration of VME crates. For the configuration procedure software is provided by the CMS DAQ group. It is contained in the Hardware Access Library (HAL) (http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL).
The Guidelines for the VME interface design are summarized in a CMS internal note: CMS IN 2004/005

VMEBridges

The CMS online sofware currently supports two PCI-to-VME bridges:

- The CAEN optical bridge. It consists of following modules:
  - A2718 : The VME Controller.
  - A2818 : The PCI card.
- The CAEN USB bridge. This module is software compatible with the optical version. (ref. No. A1718).
- The SBS optical bridge (Model 620 or 618).

Why a PC-to-VME interface and not a Single board linux computer?

- It is cheaper to upgrade a PC than VME-board computer.
- All CMS online software is developed on Linux PCs. We will profit from recent linux kernel developments and all available linux tools. It is not so clear how a VME linux computer will be able to follow this rapid development. Experience shows that on the VME single board computers in the end you always have pretty old OSs, compilers, ... You also have to consider that the online software uses external packages which are usually developed for standard PCs. Of course it is possible that all this software will also work on such a VME-board PC but probably some of the packages have to be ported by us which can be complicated and time consuming.
- The big Linux community will not care a lot if there is a problem with some VME-board computer. So the company producing the board has to do all work which is special to that computer. If after 5 years the hardware is obsolete, who knows how long they will continue to port up-to-date kernels to their old systems. It will of course depend on some economical considerations for the company, and not on the request of a client like us...

CAEN VME Bridges

Introduction

CAEN offers two different versions of their VME bridge. One with an optical link, and a second with a USB interface:

- The CAEN optical bridge. It consists of following modules:
  - A2718 : The VME Controller.
  - A2818 : The PCI card.
- The CAEN USB bridge (ref. No. A1718). This module is software compatible with the optical version, but needs a different driver.

Link to the CAEN pages: http://www.caen.it/nuclear/product.php?mod=V2718 (http://www.caen.it/nuclear/product.php?mod=V2718)

Some technical information:

- The VME controllers of the optical version can be chained (i.e. one PCI interface can control several crates which are connected by and optical link in a ring topology.)
- The maximal length of the fiber is specified by the producer of the optical link components to be 275m for a fiber of type 62.5/125um. (This is the fiber type which comes with the module from CAEN). When you use the fiber type 50/125um the link producer claims that the maximal link length is 550m. *Attention:* Both maximal lengthes have not been tested by anybody so far.
- The PCI board A2818 supports 3V and 5V slots PC slots. *Attention: There is a jumper on the card which must be set correctly!*
- The software consists of a drivers (specific for the optical or the USB version) and of a user library which is identical for both versions. This means that the user code which works with one module automatically also works with the other version. An exception to this might be software which need a handling of interrupts. Since the USB technology is not adequate for real time applications, interrupt handling in the USB version of the VME bridge is reduced to a simple polling mechanism: The user has to poll the controller if on the VME side an interrupt has occured.
- The optical version supports full interrupt handling which means that VME interrupts are routed through the bridge and generate an interrupt on the PCI bus. How in detail this interrupt is handled by the software is described in the section "Status".

3.1.1 Technical Info

Power consumption

The following values have been measured by CAEN on the V2718:

Supply voltage	current	remark
5V	1A	ongoing data transfers
5V	0.8A	no VME activity
-12V	150mA	All Lemo outputs configured for NIM and driving
-12V	40mA	Lemo outputs inactive or configured for TTL levels
12V	0mA	Connected but not used.

The 9U solution

- The majority of the VMEControllers in CMS will be housed in 9U crates. CAEN will provide us with a solution to put the Controller in a 9U crate. A prototype of the adapter has been tested by CMS and is found to work satisfactory.
- It will NOT be possible to adapt 6U modules to the 9U version.

3.1.2 Status

The optical and the USB versions of the module are available and functional. They have been tested with the HAL library which contains full support of the modules.

- Reading large blocks with block-transfer AMs (> 8MB) causes an Error in the interface. This will be solved at the level of the VME library delivered by CAEN.
- Interrupt handling is currently implemented in the following way: The user library provides a call which goes into a wait state until an interrupt occurs. With a bitmask the user can specify which interrupt it wants to wait for. Optionally a timeout for the wait call can be given. This implementation means that the multithreading has to be handles either at the HAL level or at the user level (the software which uses the HAL). Intterrupts have been tested by the Wisconcent group. No problem seems to exists if used with firmware revisions v1718vub_Rev0.14.rbf and higher.
- The CAEN module is "FAST" in the sense that during a read process the slave really must have the data stable before giving the DTACK. During block transfers the address cycles arrive earlier than with the SBS (but VME compliant) If you realize that your module "does not work" with the CAEN but did work ok with the SBS, please check the timing on the VME backplane with a scope.
- The CAEN module (6U version only) does not drive AS and DS0/1 high. This leads to slow rising edged of these signals. This is according to VME spec, but not optimal of course. For the 6U modules which CMS will use this will probably not change. Please check if this is a problem for you, and if so, please find out why. The 9U version of the module will have these lines actively driven.

Software (for the USB Bridge V1718 and the optical bridge V2718)

The software is the original unmodified driver from CAEN. In the tar file which can be downloaded here there is just a script which does the installation of the driver in the directory which is expected by the HAL library.

Latest version of the software:

For SLC4 Linux: daq-caenvme-2.11.2-1.cmsos8.slc4.src.rpm
(http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/daq-caenvme-2.11.2-1.cmsos8.slc4.src.rpm)

This software version needs at least hal version 3-17. This is the driver which crashes less ofter than all previous drivers. Not clear if it is bug-free though...

Limits: At maximum 8MB block transfers can be made.

Installation instructions:

- As superuser run

  ```
  rpmbuild --rebuild daq-caenvme-{version}.src.rpm
  ```

  You will obtain binary RPMs in the directory

```
/usr/src/redhat/RPMS/i386
```

- You obtain a binary rpm daq-caenvme-{version-release}.i386.rpm containing the user library and the firmware upgrade utilitey, a development rpm daq-caenvme-devel-{version-release}.i386.rpm with the include files and a kernel-module-daq-caenvme-{version-release}.i386.rpm which contains the driver. (Ignore the debug rpm). Install the rpms on the machine where you need the software.
  Attention: The binary rpms of the drivers will only install on machines which run the same kernel version as the machine where you have built the rpms (i.e. where you have executed the step above).

The library is installed in /opt/xdaq/lib. The firmware upgrade utility is installed in /opt/xdaq/bin/CAENVMEUpgrade The include files are installed in /opt/xdaq/include. In addition a service named "caenvme" is created in /etc/init.d. The service is activated with the chkconfig utility so that the driver will be loaded automatically at every reboot. If you do not want to reboot your machine after the first installation just type as superuser:

```
/sbin/service caenvme start
```

The software documentation you find on the CAEN www side.

Previous versions:

- daq-caenvme-2.10.3-7.src.rpm has a problem in the proc filesystem and could crash. In addition the driver loads even if there are problems in the initialization. Later it will crash.
- daq-caenvme-2.10.2-7.src.rpm is not counting up the driver use count. You can unload the driver while it is used. This in general crashes the computer.
- daq-caenvme-slc4-2.5.8-4.src.rpm (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/daq-caenvme-slc4-2.5.8-4.src.rpm) contains still a bug in the CAENVME library/driver which causes the computer to crash.
- xdaq-caenvme-SLC4-2.5-4.src.rpm (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/xdaq-caenvme-SLC4-2.5-4.src.rpm) contains a bug in the CAENVME library/driver which causes the computer to crash.
- CAENVME_v2-5_CMS-03.tgz (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVME_v2-5_CMS-03.tgz) does not contain the firmware upgrade software.
- v2-4_CMS-02 (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVME_v2-4_CMS-02.tgz)
- v2-3d_CMS-02 (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVME_v2-3d_CMS-02.tgz) This version in addition cures the problem of PC crashes when used in multi tasking and multi processor environments. This cure has been implemented with semaphores.
- This version has a problem on multi processor machines. It might crash your machine (like all previous versions): v2-2_CMS-01 (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVME_v2-2_CMS-01.tgz)
- v2-1_CMS-02 (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVME_v2-1_CMS-02.tgz) This version has no support for byte swapping.
- CAENVMEUSB_v-1-2_CMS-01 (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/CAENVMEUSB_v-1-2_CMS-01.tgz) This version has only support for the USB module v1718.

- v-1-2_CMS-01
  This version has only support for the USB module v1718. This version has a problem for long block transfers.

3.1.4 Firmware

Here is the firmware currently used in CMS for the CAEN VME bridges. Note that the firmware for the VME Module are the same for the V1718 and the V2718. Also the Revision History (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/Revision_History.txt) of CAEN can be downloaded.

A mini utility in order to read out the firmware versions from your optical bridge module can be downloaded here: caen-firmware-version.tgz (http://cmsdoc.cern.ch/~cschwick/software/distribution/caen-firmware-version.tgz). Look into the code and change it to fit your needs.

Optical Bridge V2718

- **v1718vub_rev2.02.rbf**
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/v1718vub_rev2.02.rbf) is the firmware for the main FPGA on the VME board (it is the same firmware as for the USB version of the bridge).

- **a2818pcb_rev0.8.rbf**
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/a2818pcb_rev0.8.rbf) is the firmware for the PCI card of the optical link.

- **a2719ci_rev0.5.rbf**
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/a2719ci_rev0.5.rbf) is the firmware for the optical add-on card on the VME board.

USB Bridge V1718

- **v1718vub_rev2.02.rbf**
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/CAEN/v1718vub_rev2.02.rbf)

Important notes

A new version of the firmware is not necessarily compatible with an old version of the software.

When you upgrade the firmware of your board(s) be sure to do the upgrade with the software (= driver + library) which you have currently installed. If the firmware upgrade requires a software upgrade the software upgrade MUST BE done AFTER the firmware upgrade.

The firmware upgrade utility you can find in the installation directory of the cms online executives:

```
/opt/xdaq/bin/CAENVMEUpgrade
```

Modules with old Controller-firmware

If your module is equiped with a very old firmware (for example rev. 05 for the V1718vub firmware)and in addition for some reason you standard firmware has been corrupted during a firmware update, you could encounter problems to recover due to a small firmware bug. The following recipee allows you to recover the standard firmware of the module:

- install a software on your computer which is compatible with the backup-firmware version of your controller. (Versions 1.1 and 1.2 should do)
- Power the module up while having the jumper on the ’backup’ position.
- While the module is powered change the jumper to the ’standard’ position.
- Now do your firmware upgrade of the standard version.
- Power cycle the module and verify that the new standard firmware is working correctly. If this is the case you might also want to upgrade the backup firmware of your module.

Performance Measurements

The VME Hardware was a optical VME bridge from CAEN (V2718_Kit) and a RAMix139 VME memory (32Mb). The PC was a PIII 800 MHz PC running Linux with a 2.4.x kernel. The measurements have been carried out with

- CAEN library libCAENVME.so.2.1
- firmware for V2718 : v1718vub_rev0.11.rbf
- firmware for A2818 : a2818pcb_rev0.5.rbf
- firmware for A2719 : a2719ci_rev0.3.rbf

All measurements used the **HAL library** (http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL/index.html)(version ver-03-05-test-04).

A measurement with the usb version of the CAEN Bridge (Model V2718_Kit) cane be looked at **here**. (The measurement has been obtained with the firmware v1718vub_rev0.12.rbf for the V1718 board. The rest of the measurement conditions have been identical to those for the optical bridge.)

Performance for single accesses

This test was made running the "PerformanceTester" application contained in the HAL examples. It has been run one time with the dummy BusAdapter and one time with the BusAdapter for the VME Bridge. This allows to subtract the overhead due to the HAL library itself.

Results using the VMEDummyBusAdapter

```
================================================================================
ATTENTION ==> the measured time values do not measure the process time but
              the total time elapsed during the measurement (laboratory-time)
              So, in order to get meaningfull measurements,  be sure not to
              have other active processes on your machine.
================================================================================


(TimeoutException) Timeout during polling the item "rwFlagMask"
      1000ms have passed wheras the timeout has been set to 1000ms
      3680000 polls have been carried out
      (HardwareDevice::pollItem)

read                      r/w-item with full mask : 2.57833us
```

```
read           r/w-item with full mask and offset : 2.95839us
write                     r/w-item with full mask : 2.99179us
write          r/w-item with full mask and offset : 3.42508us
read                       r-item with full mask : 2.53569us
read             r-item with full mask and offset : 2.91862us
write                      w-item with full mask : 2.66218us
write             w-item with full mask and offset: 3.07601us
unmaskedRead              r/w-item with full mask : 2.66056us
unmaskedRead   r/w-item with full mask and offset : 2.90977us
unmaskedWrite             r/w-item with full mask : 2.65765us
unmaskedWrite  r/w-item with full mask and offset : 2.92356us
setBit                    r/w-item with 1bit mask : 3.26963us
setBit         r/w-item with 1bit mask and offset : 3.71627us
resetBit                  r/w-item with 1bit mask : 3.26833us
resetBit       r/w-item with 1bit mask and offset : 3.69591us
setBit                      w-item with 1bit mask : 2.95803us
setBit           w-item with 1bit mask and offset : 3.35657us
resetBit                    w-item with 1bit mask : 2.85544us
resetBit         w-item with 1bit mask and offset : 3.37863us
================================================================================
```

Results using the CAEN2718LinuxPCIBusAdapter

Remark: The items which need significant more time involve two VME accesses. (This is documented in the HAL documentation). In order to obtain the overhead due to the driver and the hardware the measured values of the values above measured with the DummyBusadapter must be subtracted from the values below.

```
================================================================================
ATTENTION ==> the measured time values do not measure the process time but
              the total time elapsed during the measurement (laboratory-time)
              So, in order to get meaningfull measurements,  be sure not to
              have other active processes on your machine.
================================================================================


(TimeoutException) Timeout during polling the item "rwFlagMask"
      1009ms have passed wheras the timeout has been set to 1000ms
      100000 polls have been carried out
      (HardwareDevice::pollItem)

read                      r/w-item with full mask : 14.2928us
read           r/w-item with full mask and offset : 14.7548us
write                     r/w-item with full mask : 24.1194us
write          r/w-item with full mask and offset : 24.6097us
read                       r-item with full mask : 14.1733us
read             r-item with full mask and offset : 14.7672us
write                      w-item with full mask : 13.5404us
write             w-item with full mask and offset: 14.0083us
unmaskedRead              r/w-item with full mask : 14.1884us
unmaskedRead   r/w-item with full mask and offset : 14.8598us
unmaskedWrite             r/w-item with full mask : 13.7226us
unmaskedWrite  r/w-item with full mask and offset : 14.3353us
setBit                    r/w-item with 1bit mask : 24.3967us
setBit         r/w-item with 1bit mask and offset : 24.9829us
resetBit                  r/w-item with 1bit mask : 24.3639us
resetBit       r/w-item with 1bit mask and offset : 24.9634us
setBit                      w-item with 1bit mask : 14.0308us
```

```
setBit           w-item with 1bit mask and offset : 14.5417us
resetBit                    w-item with 1bit mask : 13.821us
resetBit         w-item with 1bit mask and offset : 14.4us
================================================================================
```

Performance for Block Transfers

Measurement conditions:

- The measurements exclusively measure the block transfer performance of the VME bridge.
- Data blocks have been transferred for different sizes from the host via the VME Bridge to the memory module and vice versa. The data has been generated with a random generator.
- The size of the data blocks has been varied from 256 bytes to 32Mb. Altogether 12 blocksizes where chosen. The data read back was always verified against the data previously written. No data error has been observed.
- The measurement at each data point was repeated 100 times. The mean values of the resulting transfer rates are summarized in the colored plot below.
- Additional plots have been produced, in order to show the distribution of the data throughput for the 100 measurements taken for at each block-size.

Results

Read and write performance as a function of the block size.

Read / Write Block Transfer of optical CAEN bridge with HAL

Distribution of read throughput for fixed block sizes (100 Measurements).

Read Block Transfer performance distribution for fixed blocksize of optical CAEN bridge with HAL

Read Block Transfer performance distribution for fixed blocksize of optical CAEN bridge with HAL

Distribution of write throughput for fixed block sizes (100 Measurements).

Write Block Transfer performance distribution for fixed blocksize of optical CAEN bridge with HAL

Write Block Transfer performance distribution for fixed blocksize of optical CAEN bridge with HAL

SBS PCI-to-VME bridge

Models 618 and 620

Info about the SBS interface modules

The SBS controler is supported by the CMS online software, in particular by the HAL (http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL/index.html), the Hardware Access Library used in CMS and XDAQ. Model 618 has some loopback diagnostics feature in order to test the optical link which is missing in model 620. For the rest models 618 and 620 are identical.

The SBS interface in CMS

The HAL uses single word and block transfer accesses of the SBS module and uses the user-level linux library which comes with the module. Interrupt support is currently being implemented. It uses the SBS library support for interrupt handling (via callbacks to a user routine).

CMS wants to build/use VME64x modules which allow for plug and play crate configuration. With the SBS module it is possible configure a VME64x crate with VME64x modules which implement the VME64x-plug-and-play specification. (This means: it is possible to generate accesses with AM=2F.) The software is included in the HAL library. It has been tested with the first two VME64x VME modules of ECAL.

You should know that the SBS modules do not support D64, A64 or A40 accesses. You cannot expect terrific performance with it (see the measurements: 25-27 Mb/sec with a RAMix memory; SBS claims to have achieve 35Mb/sec (???!!!)). Another disadvantage might be, that you cannot chain the modules. If you want to have some minimal performance in the local data aquisition you may be do not want to do this anyway. (You can plug in principle many adapters in the same PC.

Be aware that you need an old conventional PCI 32 bit slot with 5V in order to be able to plug the module in the PC (there exists now modern servers which only have 3.3V PCI-64 bit connectors). The latest versions of the bridge (Model 620-3) also support 3.3V PCI slots. This is important for you if you also want to operate for example a Fedkit in the same PC: the GIII needs a 64 bit slot with 3.3V. But there are many motherboards which have a mixture of different slots.

With the current driver of SBS (v2p3p0) there is an issue if you want to use the user level interrupt callback mechanism in order to do plug and play according to the VME64 (NOT VME64x) specification. The specification requires that after power up you handle the interrupts of all boards one after the other: You must not acknowledge an interrupt of a board before having set the relevant registers for the module configuration. Unfortunately the SBS driver works in an incompatible for this procedure: The interrupts are all acknowledged and the calls of the user level interrupt handling routine are queued. When the handler is called the first time, the interrupts of all modules are already acknowledged.

The SBS bridge is in many repects better suited for CMS than the National MXI bridge. The latter are made for use with VXI which we do not use in CMS. In addition the NI/MXI module has the following disadvantages: The software is only partly public, nobody ever succeeded to generate a large block transfer, software updates seem to cost a lot of money, the LVDS cables are extremly unhandy, it seems to be impossible to generate the 0x2f address modifier, people have reported difficulties to use it in XDAQ-applications.

Drivers for the SBS VME bridge

This Section deals only with the drivers for SLC4 kernels (2.6.x). Many issues of the driver of the 2.4.x kernel might still be existent. The driver has not been tested. It is only given here for people who have SBS modules in the labs and would like to try to continue to use them. Please also refer to the old documentation and the described problems therein.

SBS has been bought by another company and since then the source code of the driver is only distributed with a non disclosure agreement. Therefore only binary RPMs can be distributed here. The binaries are available for kernel 2.6.9-55.0.6.EL.cernsmp here:
btp-2.6.9-55.EL.cernsmp-3.1.C.i586.rpm
btp-devel-2.6.9-55.EL.cernsmp-3.1.C.i586.rpm
btp-debuginfo-2.6.9-55.EL.cernsmp-3.1.C.i586.rpm

SBS BusAdapter and the HAL

For your conveniance you can donwload the RPM of the HAL BusAdapter here:

daq-sbslinuxbusadapter-4.0.2-4.i386.rpm
daq-sbslinuxbusadapter-devel-4.0.2-4.i386.rpm

Drivers for the SBS VME bridge

SBS has officially released a driver version v2p3p0 which has been desinged for linux kernels 2.4.x. (Can be obtained from http://www.sbs.com (http://www.sbs.com/)).

This driver compiles and installs without problems on a CERN-Linux-Redhat-7.3.4 machine **which has the kernel sources installed**. (A small bug in the mkbtp script has been removed for the CMS version below). It also compiles and installs on CERN Linux SLC302 and SLC303 machines. **HOWEVER**there is an issue for SLC303 machines (kernel 2.4.21-20.EL.cern and the corresponding smp version). On these kernels BusErrors get losts (the software continues as if the access with the BusError was successfull. In case of a read access arbitrary data is delivered to the user.) The reason for this behaviour and a work around has been found by **Evgueni Vlassov**:

- The driver of SBS is not running correctly on SMP (multi-processor) machines. Since the newer kernel versions have been released various kernel tasks are distributed on the different processors of the system (on older kernel versions all kernel tasks including the interrupt service routines where running on the same processor). Due to this (and a bug in the SBS driver) the synchronization of a Interrupt service routine and some other driver code gets lost. As a consequence the Bus Errors get lost.
- A possible workaround for this problem (which only seems to appear on multi-processor machines) is to either load the single processor version of the kernel (in this case you do not exploit more than one processor of your machine) or to downgrade to an old kernel version (It seems that kernel version 2.4.20-30.7.cernsmp works.) We have not tried yet to downgrade a SLC303/304 system to such an old kernel version and verified that all drivers used in the DAQ environment compile and run correctly. It must be tried out.

Thanks a lot to Evgueni who spent quite some time to find out the details of this problem. Please contact Evgueni for further details.

A feature of the original SBS driver was that certain functions (those of the so called nano bus API) where not correctly declared as extern "C" if compiled with C++. This excluded the use of these functions in the HAL since the linker did not find the functions due to incorrect name mangling. As a consequence it was not possible to use the bt_reset() function in order to reset the VME bus. This has been corrected in the latest CMS-version fo the driver.

Latest version for CMS:

v2p3p0_CMS-01
(http://cmsdoc.cern.ch/~cschwick/software/distribution/SBS/SBS_v2p3p0_CMS-01.tgz)

This version is based on the SBS version v2p3p0. It compiles without problems on CERN Linux 7.3.4. The btpapi.h had to be modified so that the bt_reset() function can be called from C++ programs. It works with HAL versions ver-03-04 and higher. Earlier versions have to adjust the path to the SBS driver by hand (in Makefile.in) since the v2p3p0 subdirectory is not recognized by those versions in the configure script.

Installation instructions:

- Install the PCI-card of the SBS interface in the PC. (If you do not do this the driver compiles and will be loaded, but no device files will be created)
- Download the latest version of the patched driver into a directory of your choice. (See link above.)
- Unpack the tgz file:

```
gtar -xzvf SBS_{version}.tgz
```

-

```
cd SBS
```

- become superuser and type

```
./installSBS
```

The software documentations you find the the doc subdirectory. The driver is installed in the directory /usr/local/SBS. If this directory is already present, you are asked to remove or rename it. To load the driver after every boot you have to execute

```
/usr/local/SBS/1003/v2.0/sys/mkbtp
```

Previous versions:

- v2-0_CMS-02
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/SBS/SBS_v2-0_CMS-02.tgz): Does compile on CERN-Linux-Redhat-7.3.3 and does contain the fix for the nano bus API. You can use this version on CERN Linux 7.3.3 with HAL version later than ver-03-03.
- v2-0_CMS-01
  (http://cmsdoc.cern.ch/~cschwick/software/distribution/SBS/SBS_v2-0_CMS-01.tgz): Does compile on CERN-Linux-Redhat-7.3.3 but does not contain the fix for the nano bus API. Therefore this version will not be usable with HAL SBS-BusAdapters which implement the resetBus() method. (HAL Versions LATER than ver-03-03.)

Performance Measurements

The VME Hardware was a optical VME bridge from SBS (Model620) and a RAMix139 VME memory (32Mb). The PC was a PIII 800 MHz PC running Linux with a 2.4.x kernel. The measurements have been carried out with

- SBS driver and user library version v2p3p0

All measurements used the **HAL library** (http://cmsdoc.cern.ch/~cschwick/software/documentation/HAL/index.html)(version ver-03-05-test-04).

Performance for single accesses

This test was made running the "PerformanceTester" application contained in the HAL examples. It has been run one time with the dummy BusAdapter and one time with the BusAdapter for the VME Bridge. This allows to subtract the overhead due to the HAL library itself.

Results using the VMEDummyBusAdapter

```
================================================================================
ATTENTION ==> the measured time values do not measure the process time but
              the total time elapsed during the measurement (laboratory-time)
              So, in order to get meaningfull measurements,  be sure not to
              have other active processes on your machine.
================================================================================


(TimeoutException) Timeout during polling the item "rwFlagMask"
      1000ms have passed wheras the timeout has been set to 1000ms
      3680000 polls have been carried out
      (HardwareDevice::pollItem)

read                        r/w-item with full mask : 2.57833us
read           r/w-item with full mask and offset : 2.95839us
write                       r/w-item with full mask : 2.99179us
write          r/w-item with full mask and offset : 3.42508us
read                          r-item with full mask : 2.53569us
read             r-item with full mask and offset : 2.91862us
write                         w-item with full mask : 2.66218us
write             w-item with full mask and offset: 3.07601us
unmaskedRead                r/w-item with full mask : 2.66056us
unmaskedRead   r/w-item with full mask and offset : 2.90977us
unmaskedWrite               r/w-item with full mask : 2.65765us
unmaskedWrite  r/w-item with full mask and offset : 2.92356us
setBit                      r/w-item with 1bit mask : 3.26963us
setBit         r/w-item with 1bit mask and offset : 3.71627us
resetBit                    r/w-item with 1bit mask : 3.26833us
resetBit       r/w-item with 1bit mask and offset : 3.69591us
setBit                        w-item with 1bit mask : 2.95803us
setBit           w-item with 1bit mask and offset : 3.35657us
resetBit                      w-item with 1bit mask : 2.85544us
resetBit         w-item with 1bit mask and offset : 3.37863us
================================================================================
```

Results using the SBS620LinuxPCIBusAdapter

Remarks:

- The items which need significant more time involve two VME accesses. (This is documented in the HAL documentation).
- In principle the SBS allows to map the VME Devices of the crate into memory. In this case single accesses can be performed much faster. However this technique has significant disadvantages for the software design. It is much harder to write a VMEBusAdapter for the HAL using memory mapping which works for an arbitrary crate configuration. Since there was no urgent need to improve the performance for single word VME accesses, such a busadapter has not been written.

```
================================================================================
ATTENTION ==> the measured time values do not measure the process time but
              the total time elapsed during the measurement (laboratory-time)
              So, in order to get meaningfull measurements,  be sure not to
              have other active processes on your machine.
================================================================================



read                      r/w-item with full mask : 18.6139us
read           r/w-item with full mask and offset : 19.0851us
write                     r/w-item with full mask : 33.4937us
write          r/w-item with full mask and offset : 34.2225us
read                        r-item with full mask : 18.7048us
read             r-item with full mask and offset : 19.2273us
write                       w-item with full mask : 18.5893us
write             w-item with full mask and offset: 18.9426us
unmaskedRead              r/w-item with full mask : 19.1327us
unmaskedRead   r/w-item with full mask and offset : 19.5662us
unmaskedWrite             r/w-item with full mask : 18.4104us
unmaskedWrite  r/w-item with full mask and offset : 19.0731us
setBit                    r/w-item with 1bit mask : 34.2165us
setBit         r/w-item with 1bit mask and offset : 34.5748us
resetBit                  r/w-item with 1bit mask : 33.7547us
resetBit       r/w-item with 1bit mask and offset : 34.6076us
setBit                      w-item with 1bit mask : 18.9606us
setBit           w-item with 1bit mask and offset : 19.556us
resetBit                    w-item with 1bit mask : 18.6855us
resetBit         w-item with 1bit mask and offset : 19.2885us
================================================================================
```

Performance for Block Transfers

Measurement conditions:

- The measurements exclusively measure the block transfer performance of the VME bridge.
- Data blocks have been transferred for different sizes from the host via the VME bridge to the memory module and vice versa. The data has been generated with a random generator.
- The size of the data blocks has been varied from 256 bytes to 32Mb. Altogether 12 blocksizes where chosen. The data read back was always verified against the data previously written. No data error has been observed.
- The measurement at each data point was repeated 100 times. The mean values of the resulting transfer rates are summarized in the colored plot below.
- Additional plots have been produced, in order to show the distribution of the data throughput for the 100 measurements taken for at each block-size.

Results

Read and write performance as a function of the block size.

Read / Write Block Transfer of optical SBS bridge with HAL

Distribution of read throughput for fixed block sizes (100 Measurements).

Read Block Transfer performance distribution for fixed blocksize of optical SBS bridge with HAL

Read Block Transfer performance distribution for fixed blocksize of optical SBS bridge with HAL

Distribution of write throughput for fixed block sizes (100 Measurements).

Write Block Transfer performance distribution for fixed blocksize of optical SBS bridge with HAL

Write Block Transfer performance distribution for fixed blocksize of optical SBS bridge with HAL