μCap DAQ Technical Report and User Manual

Tom Banks and Fred Gray
University of California, Berkeley

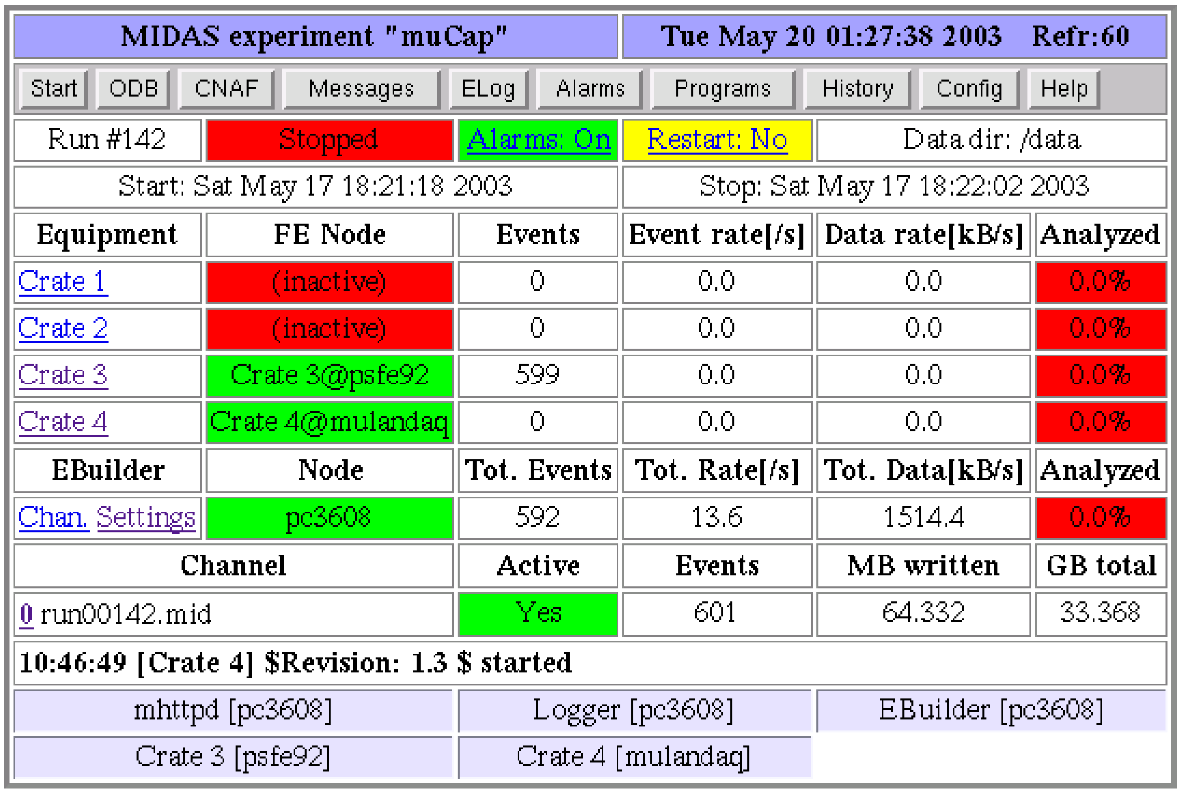
May 21, 2003

1 DAQ hardware and software

The μCap DAQ system consists of the following elements:

- Crate 1, a 9U VME crate containing
  - CPU: Motorola MVME2600 (hostname psfe90; 200 MHz PowerPC 604e)
  - SBS Technologies PMC-Gigabit-ST3 gigabit Ethernet card
  - Active electronics: 3 TDC400 modules
- Crate 2, a 9U VME crate containing
  - CPU: Motorola MVME5100 (hostname psfe91; 500 MHz PowerPC 7410) [1]
  - SBS Technologies PMC-Gigabit-ST3 gigabit Ethernet card
  - Active electronics: 4 TDC400 modules
- Crate 3, a 6U VME crate containing
  - CPU: Motorola MVME2300 (hostname psfe92; 333 MHz PowerPC 604r) [2]
  - Active electronics: 2 CAEN V767 multi-hit TDCs, SIS36xx FIFO for ePC compressors, 4 Struck DL401 flash ADCs, 1 Struck DL403 clock generator
- Crate 4, a 9U VME crate containing
  - SIS VME-PCI interface
  - CPU: commodity PC (hostname mulandaq, dual 1 GHz Pentium 3; National Semiconductor DP83820 gigabit Ethernet)
  - Active electronics: 6 $g-2$ two-phase waveform digitizers, 1 $g-2$ multi-hit TDC

[1] Borrowed from Martin Grossman as a temporary replacement for a second MVME2600.
[2] Borrowed semi-permanently from Martin Grossman.



MIDAS experiment "muCap"			Tue May 20 01:27:38 2003 Refr:60		
Run #142	Stopped	Alarms: On	Restart: No	Data dir: /data	
Start: Sat May 17 18:21:18 2003			Stop: Sat May 17 18:22:02 2003		
Equipment	**FE Node**	**Events**	**Event rate[/s]**	**Data rate[kB/s]**	**Analyzed**
Crate 1	(inactive)	0	0.0	0.0	0.0%
Crate 2	(inactive)	0	0.0	0.0	0.0%
Crate 3	Crate 3@psfe92	599	0.0	0.0	0.0%
Crate 4	Crate 4@mulandaq	0	0.0	0.0	0.0%
EBuilder	**Node**	**Tot. Events**	**Tot. Rate[/s]**	**Tot. Data[kB/s]**	**Analyzed**
Chan. Settings	pc3608	592	13.6	1514.4	0.0%
Channel		**Active**	**Events**	**MB written**	**GB total**
0 run00142.mid		Yes	601	64.332	33.368
10:46:49 [Crate 4] $Revision: 1.3 $ started					
mhttpd [pc3608]		Logger [pc3608]		EBuilder [pc3608]	
Crate 3 [psfe92]		Crate 4 [mulandaq]			

Figure 1: Top-level page of the μCap MIDAS interface.

- Control PC
  - CPU: commodity PC (hostname pc3608, 1.7 GHz Pentium 4; Intel PRO/1000 T Server Adapter gigabit Ethernet)

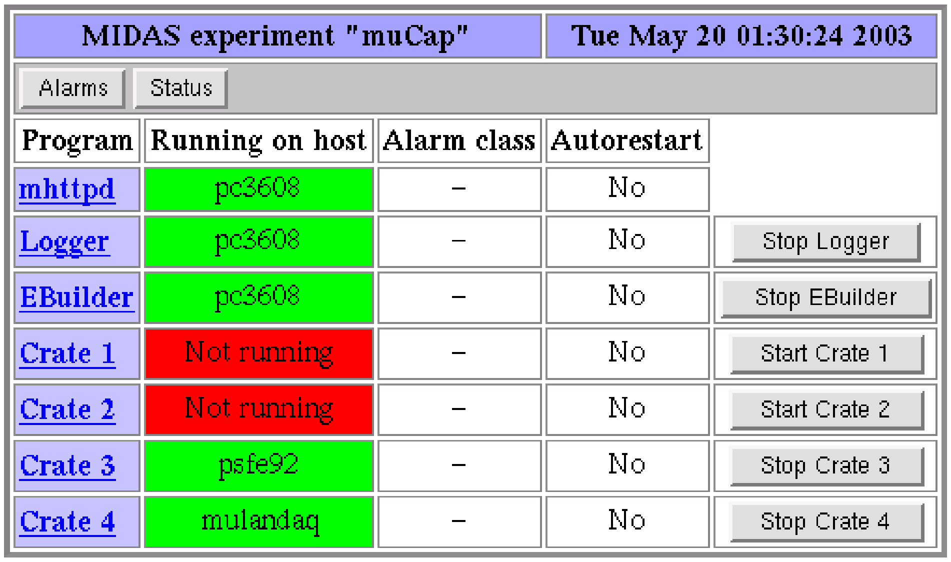
The computers are connected through a D-Link DGS-1008T gigabit Ethernet switch. The control PC and crates 1, 2, and 4 are equipped with gigabit Ethernet cards, while the rate in crate 3 is expected to be low enough that gigabit speed is not required.

2 Quick start guide

When pc3608 is booted, the `mserver`, `mhttpd`, and `mlogger` MIDAS processes are started automatically. Consequently, it is possible to connect a Web browser to

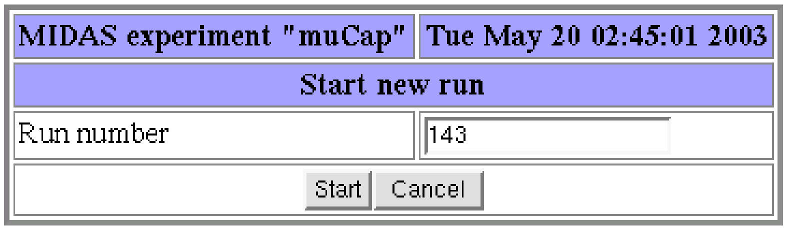
```
http://pc3608.psi.ch:8081/
```

with no further setup. The resulting top-level page appears in Figure 1. The first stage in starting up the system is to verify that the system is in the “Stopped” state, as indicated next to the run number 142 in Figure 1. Otherwise, this text would indicate “Running” on a green background. If



MIDAS experiment "muCap"			Tue May 20 01:30:24 2003	
Program	Running on host	Alarm class	Autorestart	
mhttpd	pc3608	–	No	
Logger	pc3608	–	No	Stop Logger
EBuilder	pc3608	–	No	Stop EBuilder
Crate 1	Not running	–	No	Start Crate 1
Crate 2	Not running	–	No	Start Crate 2
Crate 3	psfe92	–	No	Stop Crate 3
Crate 4	mulandaq	–	No	Stop Crate 4

Figure 2: Page obtained by selecting “Programs” from the top-level page of the MIDAS interface.



MIDAS experiment "muCap"	Tue May 20 02:45:01 2003
Start new run	
Run number	143

Figure 3: Page obtained by selecting “Start” from the top-level page of the MIDAS interface.

necessary, the run should be stopped by selecting the “Stop” button, which appears in place of the “Start” button when relevant.

The next step in starting up the system is to select the “Programs” button. This button leads to a page that lists all MIDAS processes, indicating whether or not they are running. If a process is running, the associated hostname is displayed on a green background. Otherwise, the words ”Not running” are displayed on a red background. A button is provided to start or to stop each process. In normal operation, all of the processes listed must be running. To return to the top-level page, one may select the “Status” button.

Once all processes have been started, a run may be started by selecting the “Start” button on the top-level page. This action will lead to a page requesting the new run number, as shown in Figure 3. The run number should be verified or updated, and the “Start” button should be selected on this page as well. Unless an event limit or file size limit has been selected in the logger settings, the run will continue until it is explicitly stopped by selecting the “Stop” button on the top-level page.

3 Software structure

Each crate runs a MIDAS front-end program. Crates 1 and 2 run exactly the same program; crates 3 and 4 each have their own unique program. These program transmit their partial events over the network to the MIDAS event builder running on pc3608. Events arrive asynchronously from the crates and are matched up according to their serial numbers by the event builder. The event builder contains user hooks to which online compression can be attached.

The four crates synchronize with each other through the MIDAS remote procedure call (RPC) mechanism. RPC services allow one program to initiate calls to one of a designated set of functions within a different program, even if the two programs are running on different machines. The procedure call takes the form of a network message sent over a TCP stream. The MIDAS library function `cm_yield`, which is called periodically by all relevant processes, checks for any messages that may have been received and calls the associated handler functions with the arguments provided. Optionally, the caller may wait for the recipient function to exit and return a value to the caller.

Crate synchronization involves two RPC calls:

- `rpc_ready_for_cycle(INT crate_number, INT cycle_number)`:
  Direction: From other crates to crate 3.
  This RPC indicates that crate `crate_number` is ready to start another event involving RAM number `cycle_number`.

- `rpc_end_of_cycle(INT cycle_number, INT event_number)`:
  Direction: From crate 3 to other crates.
  This RPC indicates that the data-taking phase of the most recent event, number `event_number` involving RAM number `cycle_number`, has ended and that its readout should begin.

Both of these RPCs are used in `TCP_FAST` mode, which does not wait for a value to be returned. In this mode, RPCs seem to have a latency of a few hundred microseconds.

This handshaking method is essentially a “pre-approval” process. Each crate sends its `rpc_ready_for_cycle` message as soon as it has finished its readout of the previous event, without waiting for the end of the current event. In general, crate 3 should receive the RPCs permitting the beginning of the next RAM2 cycle before the RAM1 cycle is finished. The WFDs (crate 4) are treated as a special case. If the ODB key `/Equipment/Crate 4/Settings/Synchronous` is set to “n,” then the WFDs (crate 4) are treated as a special case. They do not have a double-buffer or FIFO mechanism, so they would otherwise introduce significant deadtime. If this ODB property is set to “n,” then crate 3 does not wait for an `rpc_ready_for_cycle` from crate 4. If it has been received by the time it is otherwise ready to start the next cycle, then the WFDs are included; otherwise, they are not. When crate 4 is not included, the WFD start signal is not generated, and no `rpc_end_of_cycle` message is sent to crate 4 at the end of the cycle. Crate 4 later sends the appropriate number of zero-length events in addition to its next real event to catch up and remain synchronized.

Many useful configuration options are accessed through the online database (ODB). The ODB is a hierarchical system like a filesystem directory structure that maps keys to values. The Web interface, reached by selecting the ODB button on the top-level status page, allows the user to navigate though the hierarchy of subdirectories to reach a key of interest and to view and modify

	MVME2600 (200 MHz PowerPC 604e)		MVME5100 (500 MHz PowerPC 7410)	
MTU	BSD sockets	Zero-copy	BSD sockets	Zero-copy
1500	15.8	30.3	65.8	94.1
9000	18.7	36.2	80.4	119.5
16000	18.8	37.2	78.9	92.8

Table 1: Throughput (10^6 bytes/s) of the gigabit link using a benchmark program to transmit random data.

its value. For example, the key `/Equipment/Crate` n`/Settings/Enabled` enables or disables crate n in the readout. Similar keys exist for each module to provide finer-grained control of what subset of the system is in use. The VME base addresses of each module are also set through the ODB, as are parameters such as pedestals and thresholds. The contents of the ODB are saved in each raw data file, providing a permanent history of the configuration of the DAQ system.

4 Gigabit Ethernet performance

A number of throughput tests were performed with the gigabit Ethernet system. The theoretical maximum throughput of TCP/IP over gigabit Ethernet using standard 1500 byte frames is 117.7×10^6 bytes/s. It seems to be possible to almost reach this rate with sufficiently fast computers on each end, but a substantial amount of CPU time is required to do so. Notably, the older MVME2300 and MVME2600 CPUs are not capable of even getting close.

To measure potential throughput, a benchmark program supplied by John Heffner of the Pittsburgh Supercomputing Center was used. It provides two modes, one which transmits data using the ordinary BSD socket library and one which uses the `sendfile` system call. The latter mode is called “zero copy” because it allows the Linux network protocol stack to avoid touching the data being transmitted; in this mode, all copy and checksum operations are offloaded to the network card rather than being performed by the main CPU. Table 1 shows the throughput achieved as a function of the maximum transmission unit (MTU) for two different CPU boards. No inexpensive gigabit switch, including ours, supports MTU settings larger than 1500 bytes (“jumbo frames”), so these tests were performed with a direct cable connection from the PowerPC board to pc3608. The TCP window size was set to 256 KB on both sizes.

One important discovery was that all PCI latency register on the MVME2300 and MVME2600 series boards are set to 0 by the Motorola firmware. This register indicates how long each device is allowed to monopolize the PCI bus with burst transfers; a value of 0 disables burst transfers altogether. Not surprisingly, performance was very poor with this setting, typically around 10×10^6 bytes/s. The values shown in Table 1 were obtained with a PCI latency timer setting of 128, which is the default for the MVME5100 series boards. A small additional improvement beyond these results can be obtained with a latency timer value of 255, which is the maximum.

5 PowerPC Linux setup

As part of the recent MIDAS development, the crate CPUs were set up to run under Linux rather than vxWorks. A jumper on each CPU board (in a different location on each model; see the Motorola documentation) selects between two different flash memory modules for the board's bootstrap program. One of the modules contains the vxWorks firmware, and the other contains the original Motorola PPCBug firmware, which is used to load Linux. The Linux kernel is downloaded using the TFTP protocol from the `/tftpboot` directory on pc3608. A kernel was custom-built for each board using the source distribution obtained via rsync from the `linuxppc_2_4_devel` tree as described at

`http://www.penguinppc.org/dev/kernel.shtml`.

The kernel is then configured to mount its root filesystem via NFS (Network Filesystem) from `pc3608:/data/ppc/export/`*hostname*. This directory contains a copy of the Debian GNU/Linux 3.0 distribution for the PowerPC. Unfortunately, this NFS hostname and path are set in each kernel image through the kernel configuration file, so it is necessary to rebuild the kernel to change them.

On the other hand, if it ever becomes necessary to change the PowerPC configuration, which is used for TFTP, there are a few essential and non-obvious PPCbug commands to be used on the serial line interface. The network settings are accessed through the `niot` ("network I/O teach") command, while all other relevant parameters are set with `env`. Finally, "factory fresh" boards are shipped with the real-time clock disabled; for some reason, this also inhibits the network interface. Consequently, the clock on a new PowerPC board must be set in PPCBug with the command `set`*MMDDYYhhmm*, where the argument represents the current date and time.

6 PVIC mode

Previous incarnations of the μCap DAQ system have used the PVIC link, a proprietary PCI bus interconnect system procured from Creative Electronic Systems (CES) SA. Although the current configuration of the system does not use the PVIC bus in any way, a mode of operation exists and has been tested where it provides the primary data transfer mechanism. However, it was not reliable enough to actually be suitable for production running; at higher transfer rates, the PVIC bus sporadically locks up or drops nodes from the chain. The PVIC hardware has been returned to CES so that known fixes to the transceivers can be applied.

If it does not prove possible to transfer data with sufficient bandwidth from crates 1 and 2 using gigabit Ethernet, the PVIC will likely have to be re-introduced. In this event, a hybrid scheme will probably be used: the PVIC link will connect only crates 1 and 2 to the control PC. By keeping the PVIC chain shorter and the number of nodes smaller, the stability of the system is likely to be somewhat better.

In PVIC mode, a PVIC event builder program is run on pc3608, in place of or (in hybrid mode) in addition to the ordinary MIDAS event builder program. It is based on the master program from the pre-MIDAS DAQ system, but it has been converted to the form of a MIDAS front-end program so that it can inject data into the MIDAS buffer system. It coordinates with the crate front-end programs to transfer data from a reserved area (with a `mem=` kernel boot option) near the top of the

RAM of the crate PC to a similar area on pc3608. The data in this area is formatted as a MIDAS event. Two additional RPC calls are employed:

- `rpc_event_ready(INT crate_number, INT event_number, INT pci_address, INT size)`:
  Direction: From crates to PVIC event builder.
  This RPC indicates that crate `crate_number` has finished reading out event `event_number`, and that the data begins at its address `pci_address` and extends for `size` bytes.

- `rpc_buffer_free(INT pci_address)`:
  Direction: From PVIC event builder to crates.
  This RPC indicates that the PVIC event builder has copied the data from the buffer beginning at `pci_address` and that the crate is free to reuse this region of memory.

7 VME address maps

The VME base addresses of various modules are set through the ODB in the tree under `/Equipment/Crate` n`/Settings`. These settings must, in general, match the hardware settings on the boards. The current address maps are shown in Table 2.

8 Data format

The high-level data format is described by appendix A of the MIDAS manual,

`http://midas.triumf.ca/doc/Midasformat.html`.

It is recommended, but not strictly necessary, to use the MIDAS libraries to access the data within these files. In all cases, the data have been swapped if necessary to little-endian byte order, corresponding to the Intel architecture. 32-bit bank structures are used, since the number of bytes in a bank frequently exceeds 64K. Not all defined banks are necessarily present in any given event, depending on various ODB settings and on whether or not crate 4 participated. The following types of banks are currently defined:

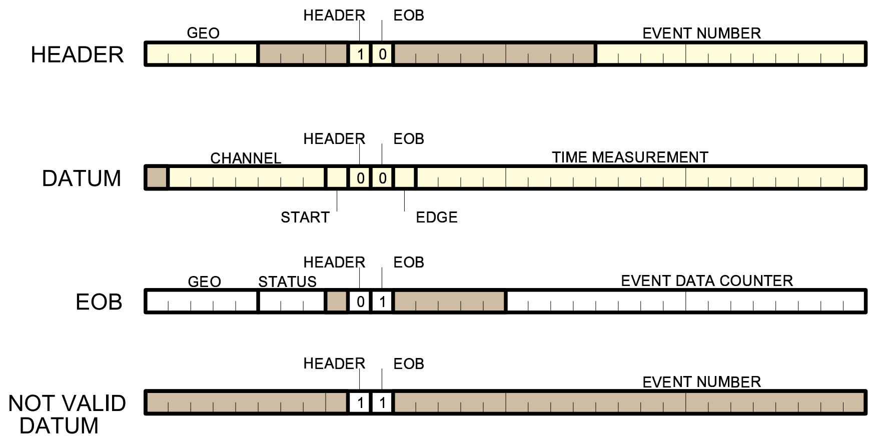
- Raw TDC400
  Bank name: TDCn
  Data structure: a sequence of 64 bit words formatted as follows:

  | Time (16 bits) | Input bitmap (48 bits) |

- CAEN TDC
  Bank name: CAEn
  Data structure: a sequence of 32 bit words formatted as shown below (Reproduced from p. 27 of the CAEN V767 User's Manual.):

Module	Base addr.	Addr. mode	User/supervisor
	Crate 1		
TDC400 1	0x08000800	A32	U/S
TDC400 2	0x08001000	A32	U/S
TDC400 3	0x08001800	A32	U/S
	Crate 2		
TDC400 4	0x08002000	A32	U/S
TDC400 5	0x08002800	A32	U/S
TDC400 6	0x08003000	A32	U/S
TDC400 7	0x08003800	A32	U/S
	Crate 3		
CAEN TDC 1	0x08000000	A32	U/S
CAEN TDC 2	0x08010000	A32	U/S
CAEN TDC 3	0x08020000	A32	U/S
SIS FIFO	0x08030000	A32	U/S
VMIC I/O	0x0800	A16	S
DL403 clock	0xA000	A16	U
DL401 ADC 1	0xB000	A16	U
DL401 ADC 1	0x08010000 †	A32	U
DL401 ADC 2	0xB100	A16	U
DL401 ADC 2	0x08200000 †	A32	U
DL401 ADC 3	0xB200	A16	U
DL401 ADC 3	0x08300000 †	A32	U
DL401 ADC 4	0xB300	A16	U
DL401 ADC 4	0x08400000 †	A32	U
	Crate 4		
WFD 1	0xE0100000	A32	U/S
WFD 2	0xE0200000	A32	U/S
WFD 3	0xE0300000	A32	U/S
WFD 4	0xE0500000	A32	U/S
WFD 5	0xE0400000	A32	U/S
WFD 6	0xE0700000	A32	U/S
MTDC	0xE8000000	A32	U/S

Table 2: Base addresses for μCap DAQ electronics. († A32 base addresses for the DL401 ADCs are configured by the DAQ software rather than in the hardware.)



- Compressor FIFO
  Bank name: CMPn
  Data structure: a sequence of 32 bit words formatted as described on pp. 6-7 of R. Prieels, "Manual for COMET,":

  |BAxx|xxxx|tttt|tttt|tttt|tttt|tttt|tttt|

  "where the values of B and A are the MSB specifying the group, xxxxxx indicates the FPGA number and the 24 bits t give the time in units of 30 ns."

- Struck DL401 flash ADC
  Bank name: ADCn
  Data structure: a sequence of 32 bit words consisting of interleaved ADC values from the four channels on the module, formatted as

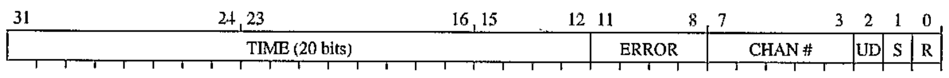
  | ADC 1 (8 bits) | ADC 2 (8 bits) | ADC 3 (8 bits) | ADC 4 (8 bits) |

- raw $g-2$ WFD
  Bank name: WF$n\{$T,B$\}$, where "T" and "B" refer to the top and bottom halves of the module.
  Data structure: Divided into the following sub-banks: phase 0 ADC, phase 0 TDC, phase 1 ADC, phase 1 TDC. Each sub-bank then has the following structure:

  – 32 bits: VME base address of this portion of the module
  – 32 bits: Starting address of data within module
  – 32 bits: Number of bytes of data needed
  – 32 bits: Number of bytes of data actually read
  – Actual data– The 8-bit ADC samples within a phase are arranged in reverse time order. The 32-bit TDC words are also arranged in reverse time order; they have the following structure:

| Time (16 bits) | Disc. 0 (4 bits) | Disc. 1 (4 bits) | Disc. 2 (4 bits) | Disc. 3 (4 bits) |

- Fitted $g-2$ WFD
  Bank name: FE$n\{$T,B$\}$
  Data structure: 16 bytes for each fitted WFD event, formatted as:
  - 32 bits: raw event number
  - 32 bits: time
  - 16 bits: area
  - 8 bits: height
  - 8 bits: width
  - 8 bits: pedestal
  - 8 bits: discriminator
  - 8 bits: quality
  - 8 bits: 0xff
- $g-2$ MTDC
  Bank name: MTDC (only one is allowed)
  Data structure:
  - 32 bits: VME base address of MTDC
  - 32 bits: Number of bytes of MTDC data to follow
  - 32 bit words in the following format (reproduced from p. 6 of E. Hazen and G. Varner, $g-2$ note 281):



R	Reserved: not currently used
S	Synchronization error
UD	Up/down bit: 0 - leading edge; 1 - trailing edge
CHAN #	1..12 ECL Data inputs 13..24 NIM Data inputs 25 ENABLE 26 DISABLE 27 RESET 28 20-bit time (mili-second) roll-over word
ERROR	bit 8 - 5-row FIFO overflow (OVFL1) bit 9 - 50-row FIFO overflow (OVFL2) bit 10 - External FIFO overflow (FFOVFL) bit 11 - parity error (PTYERR)
TIME	bits 12..21 - 10-bit ASIC time bits 22..31 - 10-bit Extended time

9 Source code

CVS is used for version control. The CVS repository is at

```
mucap@kaon.physics.berkeley.edu:/home/mucap/cvsroot,
```

and the module name is `daq`. This manual is even included in the CVS repository.

10 Electronic logbook

An electronic logbook based on Stefan Ritt's ELOG package is running at the URL

```
http://pc3608.psi.ch:8080.
```

The PSI firewall blocks Internet access to all but a very few ports. Consequently, to use this interface remotely, it is necessary to set up an ssh tunnel by running the command

```
ssh -L 8080:pc3608.psi.ch:8080 data@pc3608.psi.ch
```

and logging in with the usual password. The logbook may then be accessed from a browser running on your local machine as `http://localhost:8081`. A similar tunnel may be established, replacing 8080 by 8081, to access the MIDAS control interface remotely.