ATLAS

ATLAS Muon MDT

MROD Module: Test Procedures

Document Version: 1.0

Document ID: ATLAS-TDAQ-2005-XXX

Document Date: 26 October 2005

Document Status:

Abstract

Describes the order and procedure of tests of the MROD (MDT Read-Out Driver) module after production, prior to acceptance for use in the ATLAS experiment.

Institutes and Authors:

NIKHEF Amsterdam: H. Boterenbrood

Table 1 Document Change Record

Title:	MROD Module: Post-Production Tests		
ID:	ATLAS-TDAQ-2005-XXX		
Version	**Issue**	**Date**	**Comment**
1	0	26-Oct-2005	First version (draft).

Table of Contents

1 Introduction

The main task of the MDT Read-Out Driver module or *MROD* in the ATLAS experiment, is to receive the data streams from five to eight MDT chambers, which together form a "tower". The MROD builds event fragments from the incoming data and sends these via a so-called Read-Out Link (ROL, this is an S-Link connection) to a Read-Out Buffer (ROB, located on a ROBIN card), from where the data can be retrieved by the second-level trigger and by the Event Builder. In addition, the MROD detects and reports errors and inconsistencies in the incoming data streams (and where possible initiates corrective action). Ideally it also collects statistics and it allows to "spy" on the data. Moreover, zero suppression and data reduction schemes are implemented by the MROD.

1.1 Purpose of the document

The purpose of this document is to provide information about the post-production test procedures for the MROD module.

1.2 References

[1] "MROD-X-In Programmer's Manual", P.Jansweijer, NIKHEF, internal note, http://www.nikhef.nl/~peterj/MROD-X/MROD_X_In.pdf

[2] "MROD-X-Out Programmer's Manual", P.Jansweijer, NIKHEF, internal note, http://www.nikhef.nl/~peterj/MROD-X/MROD_X_Out.pdf

[3] TTC Interface Module (TIM) documentation, http://www.hep.ucl.ac.uk/atlas/sct/tim

[4] "ATLAS MDT ROD Production Readiness Review: Design Overview", http://www.nikhef.nl/~i73/mrodprr/mrod_prr_design.pdf

[5] "MROD Module: MROD-X Design, changes with respect to the MROD-1 design", http://www.nikhef.nl/%7Epeterj/MROD-X/MROD_X_Design.pdf

2 MROD Module Parts to Test

The following list sums up the various parts or functions of an MROD module that are to be included in the tests (not necessarily in order of importance):

- *SHARC* DSPs
- Configuration registers (FPGA function)
- GOL input links
- Buffer memories
- *SHARC* serial links
- RocketIO serial links
- Eventbuilding (FPGA function)
- TIM interface
- SLINK output link
- Interrupts to the *SHARC*s
- VME-interface
- ID chip
- Temperature sensors
- LEDs

There are in total 4, 5 or 6 ***SHARC* DSPs** (programmable Digital Signal Processors) on the MROD module, depending on the MROD type (6 or 8 input channels) and configuration.

One SHARC is located on each of the 3 (6-channel type) or 4 (8-channel type) socalled *MRODin* parts of the MROD, and one SHARC (or two) is located on the socalled *MRODout* part of the module.

Booted with software, the SHARCs handle the module's configuration, monitoring and –depending on the MROD's mode of operation– the event data flow.

Event data processing (on the MRODin) and –depending on the mode of operation– event data flow is handled by the hardware; the SHARCs configure and monitor the FPGA functionality through a series of **configuration registers** in each of the onboard FPGAs.

Event data from the MDT muon chambers is received from the CSMs (*Chamber Service Modules*) on up to 8 fiber optic **GOL input links**, which are integrated on the MROD module. Event data is processed and then stored in **buffer memories** (one for each of the 8 GOL inputs) before being transported via either **SHARC serial links** from SHARC to SHARC, or FPGA-to-FPGA **RocketIO serial links** to the output part of the MROD. There the data from the various inputs are merged (**eventbuilding**) and formatted (header and trailer data words are added), and –if required– trigger information originating from the **TIM** (TTC Interface Module) **interface** is added. Subsequently the data is send out on the **SLINK output link**, which goes to the ROS (ReadOut System).

Error situations and other (exceptional) events occurring on the MROD module while in operation are routed to **SHARC interrupt** inputs; the software running on the SHARCs responds appropriately to any interrupt source, for both modes of MROD operation.

Optionally all or part of the event data may be routed to the local system via the **VME-interface** for purposes of stand-alone data-taking and/or data monitoring, again in both modes of MROD operation. The VME-interface forms the interface between the crate controller (the local host system) and the SHARCs, providing overall control and communication with the SHARCs, and thus with the MROD. The crate controller interfaces to the general ATLAS DAQ and Run Control system.

Each MROD module can be identified by an onboard **ID chip** containing a unique identifier (64-bit number).

The temperature of the FPGAs on the MROD can by monitored through **temperature sensors** in each FPGA (i.e. simply a diode in the FPGA silicon), connected to a temperature sensor device which in its turn is connected to the FPGA. Each device can be controlled by the SHARCs via one of the FPGA's registers.

And finally there are a number of onboard **LEDs** which can be used to provide visual feedback for SHARC operations (mainly used for software debugging, because they won't be visible in a fully equipped MROD VME-crate).

Below the main components of the MROD are shown in a couple of block diagrams.

Figure 1 illustrates the situation in which the MROD data flow is controlled by the SHARCs (in other words, directly by software); this mode of operation is in fact a backup solution.

The preferred mode of operation has evolved out of what is now the backup solution, for reasons of performance, and is illustrated in Figure 2.

In the latter mode of operation, hardware controls the MROD data flow and the tasks of the onboard SHARCs have been reduced to configuration and monitoring (which are still important tasks!). Selecting between either of the 2 modes is only a matter of configuration of the MROD by the SHARC software; the hardware can run in either mode.

Figure 3 shows the layout of the communication (ring) network between the onboard SHARCs. The network is implemented by means of proprietary SHARC serial linkports. The network is the means by which the SHARCs are booted, can receive commands and send their monitoring information and error messages to the outside world. SHARCs A and B are directly accessible from the VME-bus.

Note that SHARC B will not be mounted on most of the MRODs. SHARC F is only present on 8-channel MRODs. The SHARC communication network layout is such that this is possible without breaking the communication ring.

More details about the MROD design and hardware can be found in [4] and [5].

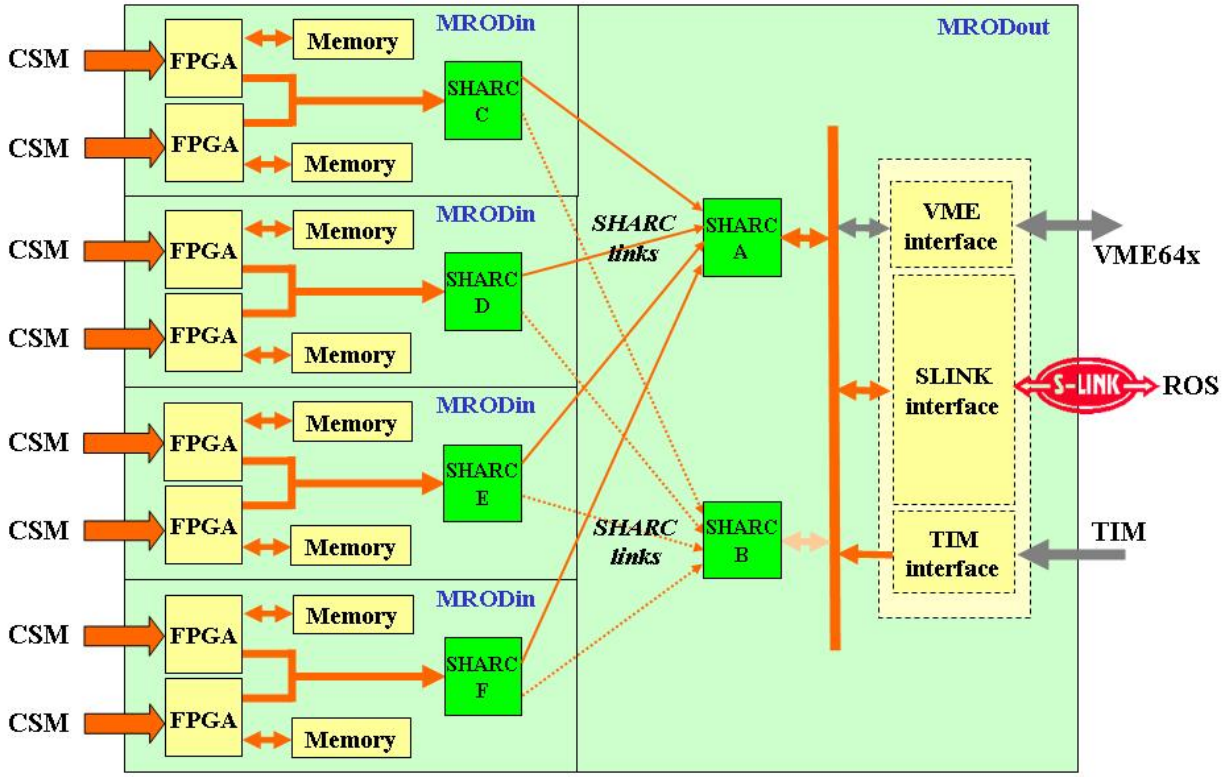

Figure 1. Simplified block diagram of the MROD-module, when used in the mode where the internal data flow is handled and controlled by the SHARCs directly. The main event data flow paths are shown in orange.

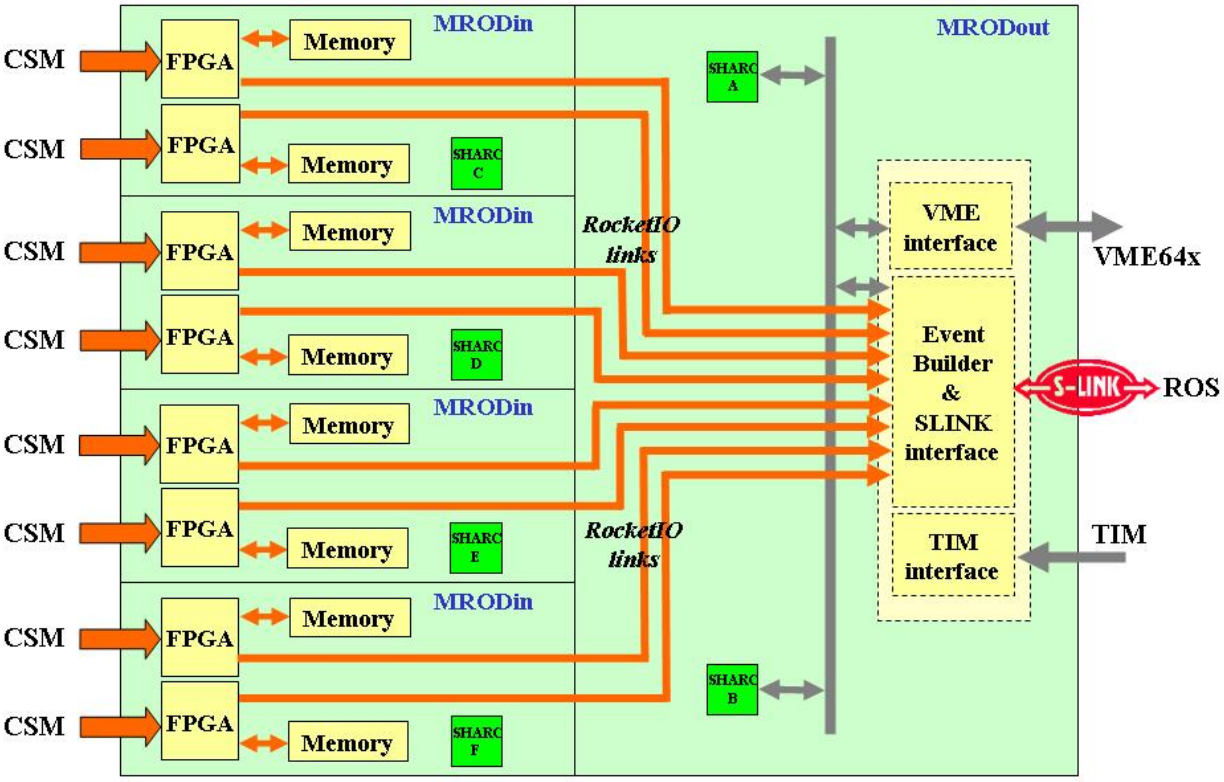

Figure 2. Simplified block diagram of the MROD-module, when used in the mode where the internal data flow is handled by hardware and the SHARCs perform only configuration and monitoring tasks. The main event data flow paths are shown in orange.

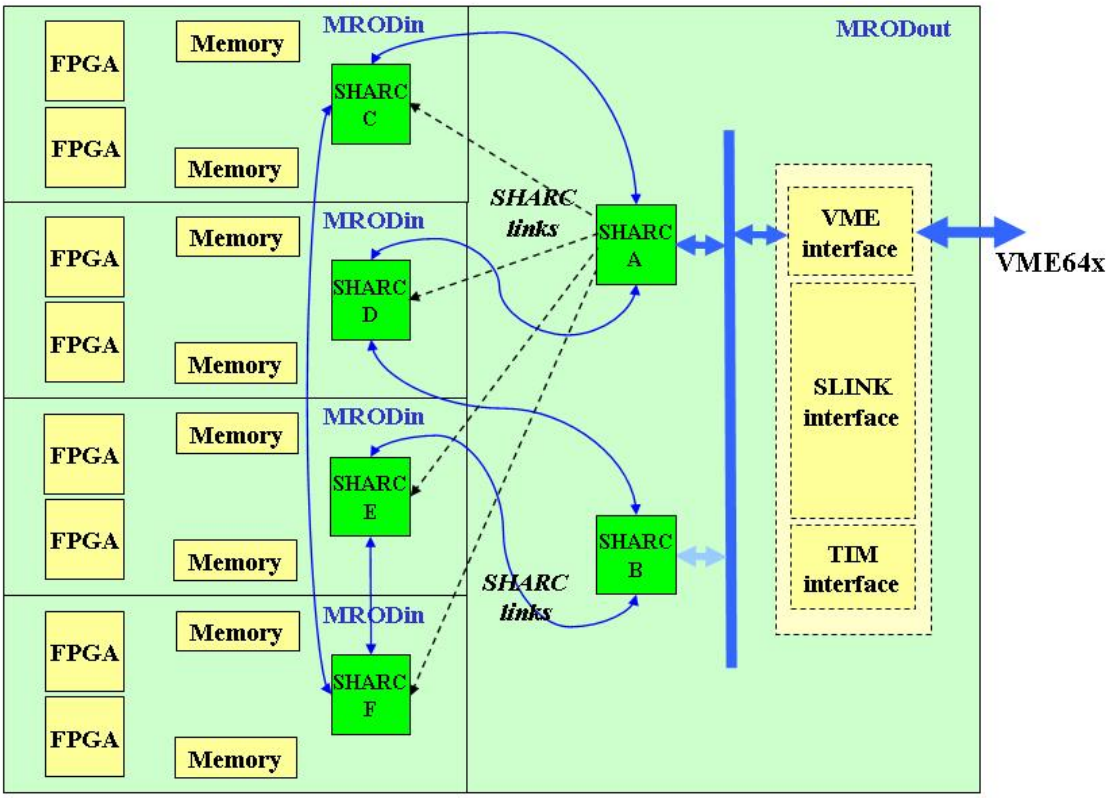

Figure 3. Block diagram of the MROD-module, showing only the communication network (in blue) between the onboard SHARC DSPs and the outside world (VME). SHARCs B and F are optional; although not shown in this diagram, there are alternative links present to deal with the absence of these SHARCs and maintain a path of communication [1]. The dashed lines (in black) denote links through which the MRODin SHARCs are booted.

[1] To be more precise: a link between SHARCs D and E to deal with an absence of SHARC B, and a link between SHARCs C and E to deal with an absence of SHARC F

3 MROD Post-Production Testing

3.1 Introduction

After production the first thing that needs to be done, is that the firmware for the various FPGAs on the MROD module is programmed into the module. Once this is done the module should be functional and testing can commence.

Testing consists of 2 parts:

- functionality test
- burn-in test

The main purpose of the functionality tests on the MROD after production is to check whether all MROD 'parts' (as listed in the section 2) are present and functioning correctly. For this purpose a series of test programs to be run on the SHARCs, have been developed.

A script to automate a sequence of these tests has been written. It starts the various test programs that display their progress and write out logfiles. Where possible test programs are run on several SHARCs (of one MROD) simultaneously. The logfiles are automatically checked for errors after each individual test, the user is informed about any errors and the test sequence is paused only in case of errors. The user can then decide to continue the sequence or quit. If no errors were detected the sequence automatically proceeds to perform the next test.

The functionality test sequence, described below in section 3.3, tests all items described in section 2 automatically, *except* the SLINK output interface. The duration of a full sequence for an 8-input-channel MROD with 6 SHARCs is about 10 minutes.

A burn-in test is described in section 3.4.

3.2 Hardware Required

The hardware items required to run the test sequence on an MROD module are:

- ROD VME-crate, with TIM backplane
- crate controller (Concurrent Technologies)
- TTC Interface Module (TIM) [3]
- MROD Module(s) (the module(s) under test)
- 6 or 8 loopback fibers per MROD module (for MROD GOL inputs)
- (fiber to connect MROD SLINK output to a FILAR in a nearby PC).

The 'loopback' fibers are plugged into the MROD's GOL inputs (up to 8). The MROD BUSY-out is routed via the backplane to the TIM.

Figure 4 shows a picture of the functionality test setup. Several MRODs are placed in the crate at the same time, after which the functionality test sequences are run sequentially on each of them. Figure 5 shows the loopback fibers connected to the MROD inputs in more detail.

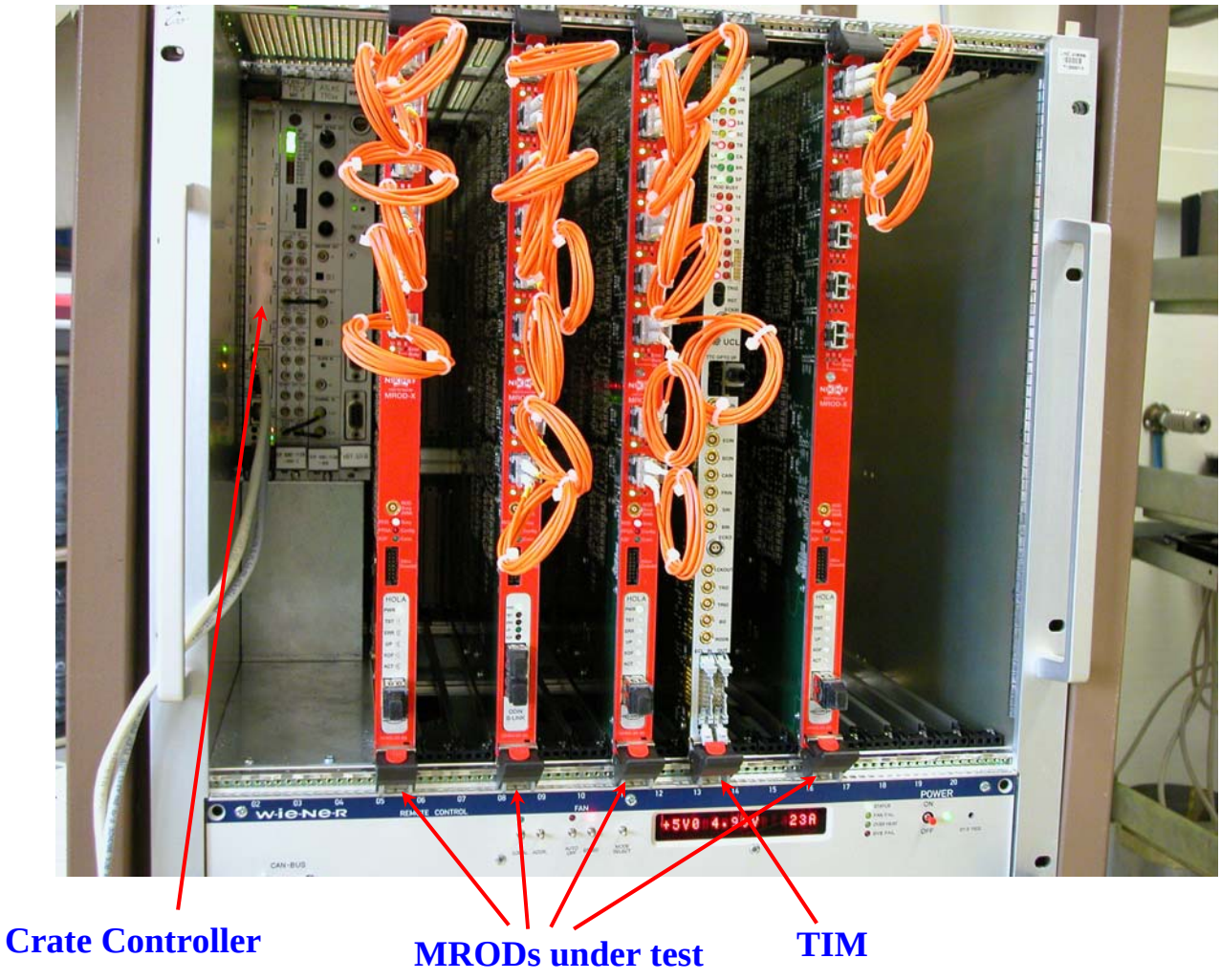

Figure 4. Typical VME-crate setup for the MROD functionality test.

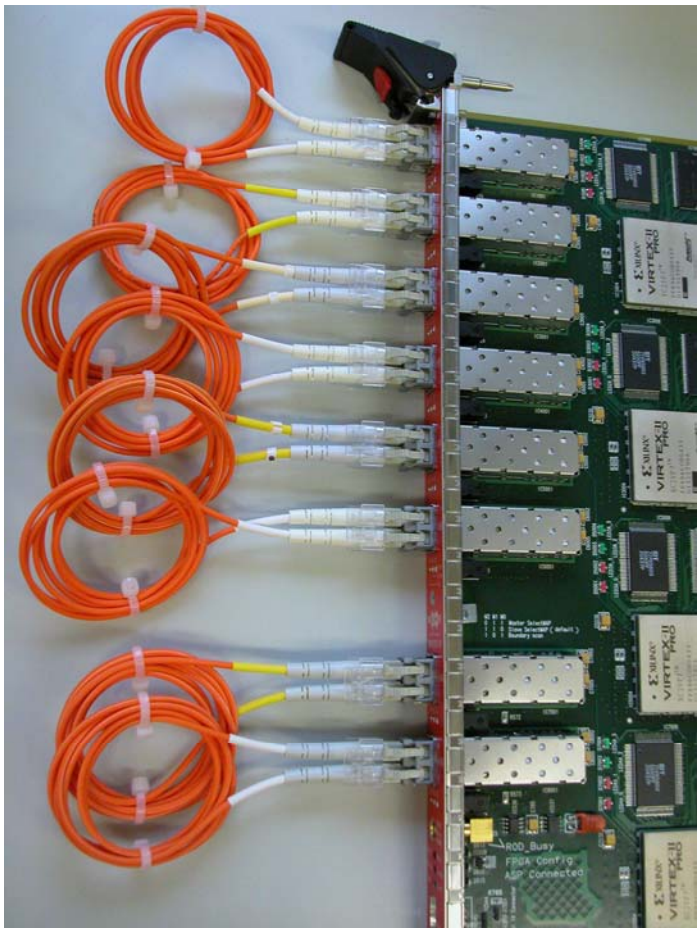

Figure 5. Loopback fibers connected to the MROD inputs.

3.3 Functionality Test

The full MROD functionality test sequence is started like this:

```
mrodtest <slot> <id>
```

in which **<slot>** is the VME-crate slot number where the MROD module to be tested is located, and **<id>** is the 'serial number' or 'identifier' that will be assigned to this MROD module. Probably a label with this serial number is to be applied to the front panel of the module, so that the module can later easily be identified.

Assuming an MROD serial number of 99, a directory called `MROD-99` is created in the directory from where **mrodtest** was called. All the test logfiles from the test programs to be run on this module will be stored in this directory.

The following test program scripts are now executed sequentially (test duration values based on an 8-input-channel MROD with 6 SHARCs) :

1. **`crcsrtest`**
   This test checks the MROD's socalled CR/CSR VME-space:
   - the socalled *BAR* register contains the VME-slot number (bits 7..3)
   - the ROM space contains a string that reads:
     "`MROD-X, Muon Drift Tube Readout Driver, NIKHEF`"
   - using the socalled *Bit Set* and *Bit Clear* registers the MRODOUT SHARCs may be reset, which is checked using known post-reset default contents of one particular SHARC register.

   Test duration: ca. 2 s.

2. **`idtest`**
   Some further information about the module is retrieved, via the SHARCs, i.e. the unique 64-bit number stored in the MROD module's ID-chip and the FPGA firmware versions of each of the MROD's FPGAs.

   Note that for this to succeed it is required that some of the SHARCs and FPGAs are present and basically functioning ! The ID-chip's content is checked for correctness by calculating its CRC and comparing with the CRC contained in the ID.

   The ID and version numbers are stored in small (text) files, in the `MROD-99` directory which was created, that have names derived from the numbers. Here is an example list of file names and a description of their content:
   - **`016800000A1B512C.txt`** : the MROD ID-chip serial number
   - **`05080300-MRODin.txt`** : MRODin FPGA version number
   - **`05080401-MRODout.txt`** : MRODout FPGA version number

   If the assigned MROD serial number and the ID are from now on linked together, it should be always possible to identify the MROD module, either visually by label, or remotely by ID.

   Test duration: ca. 2 s.

3. **`linktest`**
   This test exercises sequentially each (SHARC DSP propietary) link between two SHARCs on the MROD module, by sending blocks of data and checking the data on the receiving side; each link is tested in both directions (except for 4 links, because the second direction on this link would clash with the host server program interfacing to the program running on one of the two SHARCs involved). On a fully equipped MROD module there are in total 8+9=17 link interconnections between the 6 SHARCs.

   Test duration: ca. 3 min 15 s.

4. **`regtest`**
   In this test the SHARCs (MRODin as well as MRODout) check the default contents of each of the FPGA registers, then run a write/read test on each of the registers. The test also includes the functionality of the temperature sensor(s) associated with the FPGA(s) and a check for a reasonable temperature value ($30° \leq T \leq 55°$), as well as a check of the ID-chip contents (again), in case of the MRODout.

   Test duration: ca. 45 s.

5. **`memtest`**
   In this test each MRODin SHARC performs a number of write/read tests on the ZBT buffer memories of the MRODins (there are 2 such buffers per MRODin), including random pattern, walking-zero, walking-one and memory address overlay.

   Test duration: ca. 2 min 15 s.

6. **`iintrtest`**
   Using the internal test mode registers to upload simulated CSM data into the MRODin hardware, the (MRODin) SHARC in this test initiates the generation of one or more interrupts of each interrupt source to the SHARC. The occurrence of each individual interrupt is monitored and checked, including other interrupt-associated information, such as socalled 'interrupt overruns', event identifiers, TDC masks, etc. See MRODin documentation [1] for details.

   Test duration: ca. 25 s.

7. **`vmeintrtest`**
   In this test the MROD's capability to generate interrupts on the VME-bus is tested. A series of vectors on all levels (1 to 7) are generated, by each of the 2 MRODout SHARCs individually, and received and checked by a program running on the crate controller. It is assumed that further MROD VME-interface functionality is sufficiently tested, while running all other tests, which involve communication between SHARCs and the crate controller, via VME.

   Test duration: ca. 15 s.

8. **`rockettest`**
   This test covers the event processing and building of the MRODin and MRODout FPGAs, and the RocketIO links between the MRODin and MRODout parts of the MROD module. A small number of events (32) with a fixed size is generated, by using the internal test mode registers to upload simulated CSM data into the MRODin hardware. One of the SHARCs on the MRODout 'spies' on the events and checks the received data of the received events word-for-word. One MRODin input channel and RocketIO link is tested at a time. All 8 inputs are tested sequentially, and finally once all simultaneously.

   Test duration: ca. 45 s.

9. **`golttctest`**
   This test covers the GOL input link, the event processing and building of the MRODin and MRODout FPGAs, the RocketIO links between the MRODin and MRODout parts of the MROD module, and the connection of the MROD to the TIM backplane. With a trigger rate of 100 Hz a small number of events with a fixed size is generated using the GOL-test facility offered by the MRODin (simulated CSM data is generated on the GOL output and fed back in, using the 'loopback' fibers). One of the SHARCs on the MRODout 'spies' on the events and checks the received data of the first 100 events received word-for-word, including the TTC information received from the TIM via the backplane. One MRODin input channel GOL link and associated RocketIO link is tested at a time. All 8 inputs are tested sequentially, and finally once all simultaneously.

   Test duration: ca. 1 min 10 s.

10. **ointrtest**
    Using MROD items covered in the tests described above, interrupt sources to the SHARCs of the MRODout part of the MROD module are tested, including: SLINK down, SLINK FIFO half-full, RocketIO down, Event Counter Reset (ECR) and TTC FIFO full conditions (NB: SLINK LRL not tested). See MRODout documentation [2] for details.

    Test duration: ca. 30 s.

11. **ledtest**
    In this test all SHARCs run code to display a pattern on the LEDs they control. The user visually checks if all LEDs are functioning.

    Test duration: ca. 15 s

After the test sequence completes, a 'list directory contents' command (ls) will show the log files that have been created:

```
[n48@brenta Test] [35] ls MROD-99
0159000009C002CA.txt  golttc-Fb.log  linksend.log  rocket-all.log
05091300-MRODin.txt   intr-A.log     linktest.log  rocket-Ca.log
05100401-MRODout.txt  intr-B.log     mem-C.log     rocket-Cb.log
crcsr.log             intr-Ca.log    mem-D.log     rocket-Da.log
golttc-all.log        intr-Cb.log    mem-E.log     rocket-Db.log
golttc-Ca.log         intr-Da.log    mem-F.log     rocket-Ea.log
golttc-Cb.log         intr-Db.log    reg-A.log     rocket-Eb.log
golttc-Da.log         intr-Ea.log    reg-B.log     rocket-Fa.log
golttc-Db.log         intr-Eb.log    reg-C.log     rocket-Fb.log
golttc-Ea.log         intr-Fa.log    reg-D.log     vmeintr-A.log
golttc-Eb.log         intr-Fb.log    reg-E.log     vmeintr-B.log
golttc-Fa.log         led.log        reg-F.log
```

All log files will be kept for later reference.

Optionally **mrodtest** may be started like this:

mrodtest <slot> <id> *test*

where ***test*** is a string and should be one of *crcsr, version, link*, *reg*, *mem*, *iintr*, *vmeintr*, *rocket*, *golttc*, *ointr* or *led*. Now the **mrodtest** sequence skips all tests up to the test with this name, as described above, and proceeds to perform the rest of the test sequence from there.

Optionally a particular test may be run stand-alone, **regtest** for example:

progs/regtest.sh <slot> <id>

The MROD module may come in a number of different hardware configurations:

- With 8 input channels, i.e. four MRODin parts, each with a SHARC (known as C, D, E and F). By default **mrodtest** assumes this type of MROD when no extra options are given.
- With 6 input channels, i.e. MRODin part with SHARC F has been left out. To test this MROD type, call **mrodtest** with option **-F**.

- With only one SHARC in the MRODout part (instead of 2). The MRODout SHARCs are known as A and B. SHARC B is left out on this MROD type. To test this MROD type, call **`mrodtest`** with option **`-B`**.
- With 6 input channels and one MRODout SHARC (a combination of the previous 2 configurations). To test this MROD type, call **`mrodtest`** with option **`-FB`**. (*note: not yet implemented*).

The **`-F,-B`** or **`-FB`** option must follow the slot and id options, but may be placed before or after option ***`test`***.

Example: to test a 6-channel MROD module with serial number 99 in VME slot 16, starting with the FPGA register test, type:

```
mrodtest 16 99 -F reg
```

An additional test (which has not been included in the test sequence, for the time being, because of limited available hardware resources on-site) is available to test the functioning of the MROD S-LINK output interface, completing the test of all MROD components:

- **`slinktest`**
  This test is similar in setup to **`golttctest`**, except that only one test run is made, with all input channels included. In addition event data is now sent out on the SLINK output link which is connected to a FILAR module in a PC running a receiving application (**`slink_filar_dst`**, taken from the ATLAS *TDAQ/Dataflow* software distribution), which stores the events received into a file. After the required number of events have been received the event data file is checked (using the MROD event data integrity check program **`mrodchk`**) for correctness, i.e. no event format errors, and a minimum number of events present in the file.

3.4 Burn-In Test

In order to provide a minimum of burn-in testing, all MROD modules that have passed the functionality test sequence, described in the previous section, will be subjected to a duration test, in groups of several MRODs in one crate simulataneously, whereby each MROD runs a standalone high-rate test, resembling the **`golttctest`** described in the previous section. Events are generated by the built-in test facility and routed into the MROD inputs by ‘loopback’ fibers.

In this particular test however, events are triggered at the maximum rate that can be achieved by the combined system of MRODs, whereby the TIM in the crate serves to ‘OR’ the BUSY signals from each MROD, thus limiting the trigger rate to match the performance of the MROD system.

Data consists of events with a fixed size per MROD. One of the SHARCs on the MRODout ‘spies’ on the events and checks the received event data word-for-word, including the TTC information received from the TIM via the backplane. The SHARC doesn’t check every event, as this would limit the overall event rate considerably, but in this test it only checks every ‘n’ events (with ‘n’ chosen such that the MROD event rate is not limited by the SHARC data check operation, and the SHARC checks as much data as possible).

In addition the temperature of the MRODout FPGA is monitored and logged at regular intervals.

This test would typically run continuously for 12 hours minimum.

Figure 6 shows a picture of this setup, which is basically the same as for the functionality test, except that now we place the MRODs next to each other in the crate (and possibly several more at the same time than during the functionality test), to increase the stress on the MROD modules as much as possible, due to increased temperature and EMI from nearby modules.

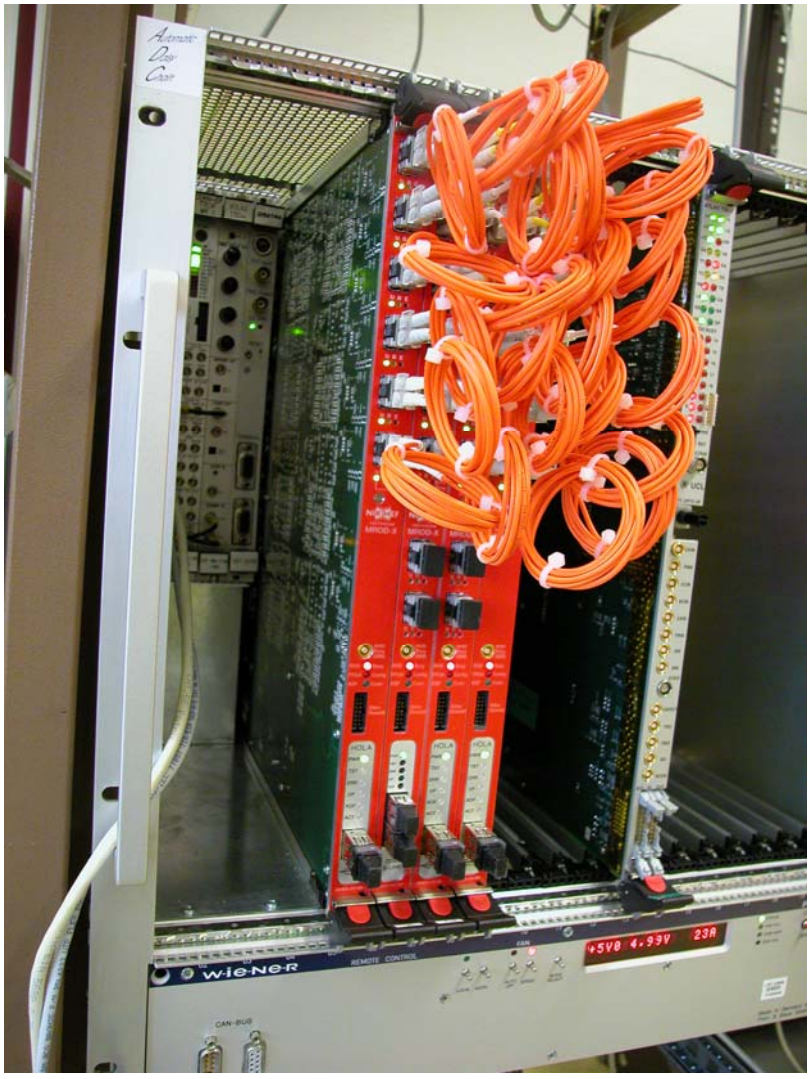

Figure 6. Typical VME-crate setup for an MROD burn-in test.

4 MROD Built-In Self Test

Although a real 'built-in' selftest is not implemented, it is possible to boot the SHARCs inbetween ATLAS data-taking periods with programs to do brief tests, or even make these tests part of the 'standard data-taking and -monitoring' SHARC code, so that they can be executed on request. These kinds of tests obviously need to be of short duration, and therefor cannot be as extensive as the post-production test sequence described earlier.

Possible tests could include e.g. the MRODin buffer memories, the RocketIO links, etc.