EDOC???

EUROGAM PROJECT

NSF DATA ACQUISITION SYSTEM

Diagnostic Procedures For EuroGam

Edition 1.0
January 1991

NSF Software Systems Group
UK Science and Engineering Research Council
Daresbury Laboratory

Diagnostic Procedures for EuroGam

David Brightly and Patrick Coleman-Smith

Edition 1.0
January 1991

1 Introduction

This paper is the result of discussion between John Alexander, Mike Bentley, Dave Brightly, Patrick Coleman-Smith, and Ian Lazarus. We started with the question "what do we, i.e., the support staff, do when an experimenter complains that the EuroGam system is refusing to acquire data?" Three main avenues of approach emerged:

Firstly, we should investigate the feasibility of a "system confidence test" of broad coverage. By this we mean an automatic procedure designed to demonstrate that the EuroGam system hardware and software can be put in a working state. This procedure would be at liberty to displace any or all of the user's current setup. It would exploit the Ge and BGO test pulse generators, the trigger unit driven "synchro-test" VXI line, and relatively lax timing windows to simulate a controlled set of physics events and check that they are correctly processed by all system components as far as data storage. It would require no physical movement of connectors, no user intervention, and be completed in perhaps 1–5 minutes. If the confidence test succeeded we would conclude that the system was in working order and that any problem lay with some aspect of the user's setup, detectors, target, beam etc. Were the confidence test to fail we would conclude that indeed some fault existed and would be obliged to adopt some methodical procedure for tracking it down.

Our second avenue of approach is to investigate the feasibility of quasi-automatic fault identification procedures. The object here is to isolate the fault within a single replaceable unit by means of a sequence of tests which extends the set of components known to be working until a fault is observed. Though we cannot guarantee to locate every exotic fault that might occur we believe that given the diagnostic mechanisms currently envisaged for EuroGam modules, together with some further ones which we have identified, we can provide a useful set of fault-finding tools. We would hope that these can be made sufficiently intelligible to experimenters that they can use them to diagnose problems occurring out of normal office hours without expert assistance. We expect that many of these tools will evolve from standalone module commissioning test programs.

Our third line of attack is to provide tools for checking a user's setup. The EuroGam hardware contains many user-settable parameters which critically affect data acquisition and we appreciate that a tyro user may well overlook or miss-define one or more of these. We therefore propose to investigate the possibility of tools for inspecting the user's setup (by interrogating the modules themselves), producing summaries, highlighting potential errors, and otherwise aiding the user in getting his setup right.

These ideas are examined in more detail in the subsequent sections.

2 System confidence testing

The object here is to demonstrate convincingly that the EuroGam data acquisition system is in working order, assuming that this *is* the case. In contrast to the tests described in the next section we make no preliminary checks of basic functions at all. Instead, we establish a viable setup configuration in the frontend crates and use the frontend cards' test modes to generate and process a predetermined sequence of pseudo-events. We can check for correct operation of the system by inspecting the stored EbyE data and the resulting singles and sorted histograms. These checks must be of the form:

- expect a peak of n counts within the region $[c \,..\, c']$ of spectrum s;
- expect events with sequence numbers $n_1 \,..\, n_2$ to consist of channels $a_1, \ldots, a_k$ with conversions in the ranges $[c_1 \,..\, {c_1}'], \ldots, [c_k \,..\, {c_k}']$.

Any discrepancies can be reported and can be of use to an expert in diagnosing the problem.

This scheme can be implemented by executing scripts in the user interface that invoke standard data acquisition functions to do the setting up (setting timing windows for example) and one or more special tools that check the resulting data and spectra. Remember that we are assuming that everything really is working properly so we can use any of the facilities of the system that we choose. Any error encountered while setting up (inability to communicate with a frontend crate, for instance) must terminate the confidence test prematurely. Again the error report associated with this failure is valuable diagnostic input to the expert who then has to sort out the problem.

The scheme will require software support in the Resource Manager controlling the Trigger Unit to cause the TU to pulse the "synchro-test" VXI line at a given frequency for a given number of times. Some sort of synchronisation with user interface scripts is also desirable so that different phases of the confidence test (setting different pulser levels, for example) can be properly sequenced. This might be achieved through the putative exception handling mechanism. Alternatively, simple timed delays and status polling of the frontend crates may be adequate, though this will extend the duration of the confidence test somewhat (and is inherently unattractive anyway).

It's worth noting what aspects the confidence test does not exercise:

- system behaviour under realistic conditions of pulse shape, height, and time distribution;
- tight timing windows that might be set using a scope;
- arbitrary user-defined trigger generation logic;
- arbitrary user-defined EbyE sorting operations;
- beam on target, detector operation, detector–frontend cabling.

Most of these aspects can only be realised with beam on target or with a radioactive source. Further confidence tests designed to run under these conditions are conceivable, but since one has little control over the system inputs in these circumstances one can do few strict numerical tests on the resulting outputs. We note that sources can produce events of multiplicity at most two.

3 Fault-finding diagnostic tools

This section examines the problem of identifying a fault once it is obvious that something is wrong somewhere. The idea is to perform a sequence of tests on isolated components until a malfunction appears. In contrast with the system confidence test we start by assuming that *nothing* is working properly. Only when a component has been shown to be working can we make use of it in a subsequent test. Ideally we would like to construct a test sequence that could be invoked from a user interface with no human intervention (to change cabling, for example). Hence we restrict ourselves to what can be achieved with builtin hardware test features and appropriate software to both set up the test and check the results. Procedures that require recabling, attaching probes, exchanging modules, etc., are the domain of the expert. We envisage that the software to perform these tests is permanently loaded in the appropriate crates and invocable by remote procedure call (RPC).

Our strategy is to gradually build confidence in parts of the system until a fault is found, using already tested components in exercising others. Since the EuroGam system is essentially a pipeline for EbyE data we start at the downstream end (data storage) and work upstream. The rest of this section discusses possible test procedures for each component in turn.

3.1 Basic crate operation

Before we can test individual modules we must be confident that each crate provides some minimal level of functionality. All crates (VXI and VME) have a processor card (invariably an MVME147 running VxWorks) which provides ethernet communications with the rest of the system. To avoid confusion we will refer to this as the "crate manager" (CM) processor. Of course, in the case of a VXI crate the CM processor is embedded in the slot 0 Resource Manager card.

CM processor If VxWorks boots to a state capable of responding to a null RPC we can assume the 147 card and ethernet communications are satisfactory.

VMEbus We will assume that each crate runs module initialisation and test code following the VxWorks boot. For VXI crates the standard requires that the presence of modules be sensed and the modules be properly initialised and brought into operation. This will certainly exercise the VMEbus and if this completes successfully we will assume that the VMEbus is operational. For VME-only crates we would expect to do some elementary module test and initialisation if only to be confident that the anticipated modules were indeed present. Some means of inquiring (by RPC) the success or failure of the initialisation code is needed. We will defer testing of the VMEbus interrupt system until it is required in a test for a specific module that can raise an interrupt.

VXIbus It's difficult to see what can be usefully done here without extra hardware. One possibility is to use the RM's capability of inspecting and asserting designated EuroGam VXI lines to make simple checks for continuity and held lines.

Power The power rail state for each crate can be checked by means of the environment monitoring subsystem. This should therefore be the first crate checked out!

3.2 Data Storage

Basic operations These can be checked out by simple file write/read/compare programs running in the user interface.

Redirection For verifying that upstream components of the EbyE pipeline are correctly forwarding data it's useful to be able to direct EbyE data to disc files rather than to tape. This eliminates the human intervention required to make sure tapes are mounted and is much quicker when accessing the small files diagnostic tests will generate.

Upstream EbyE interface Testing the EbyE data interface into the storage server requires the selftest feature in the sort unit described below.

3.3 Sort Unit (SU)

Histogram memory A conventional memory test

Slave processors A conventional processor test for each processor organised by the crate manager (CM) processor

Interface modules Appropriate standalone tests (e.g., of buffer memory) can be performed by the CM processor.

Selftest feature A means is required of injecting known test EbyE data at the frontend of the SU. One way of doing this is to download a sequence of one or more ersatz events into the SU and trigger the EbyE pipeline from the SU downstream to process these events a given number of times as if received from the Event Builder. This can be used to exercise the interfaces between the Sort Unit and Storage Unit. With a suitable sort program loaded in the SU checks can be made on its basic sorting capability by inspecting the histograms and EbyE data files resulting from processing the downloaded events.

Downstream EbyE interface This can be exercised using the SU's selftest feature or by providing an analogous test for the downstream interface alone, using downloaded ersatz events.

Snapshot feature A means of capturing a sample of the EbyE data entering the SU is valuable. This might comprise a circular buffer which always holds the last n events or, better, a buffer which captures the first n events after being explicitly armed and triggered. Users can exploit such a feature themselves to check the quality of their EbyE data in a more direct way than by sorting it since it can operate in parallel with writing tapes and saves having to inspect the tape files themselves.

Upstream EbyE interface A test for this requires the Event Builder selftest feature described below.

3.4 Event Builder (EB)

The EB is functionally similar to the Sort Unit and has the corresponding test requirements:

Slave processors As for SU.

Interface modules As for SU.

Selftest feature As for SU.

Downstream EbyE interface As for SU.

Snapshot feature As for SU. This feature of the EB is especially important as tests of much of the upstream hardware rely on inspecting EbyE data arriving at the EB. A separate implementation within a specific test program within the EB is desirable.

Upstream EbyE interface A test for this requires the ROCO selftest feature described below.

3.5 Histogrammer

Histogramming memory A conventional memory test.

Lookup table A data retention test.

Selftest feature The histogrammer supports a test mode in which a data word supplied over the VMEbus is processed as if captured from the DT32bus. A test exploiting this can be run by the histogrammer's CM processor.

DT32bus interface This can be exercised by means of the the DT32bus test outlined below and observing where histogramming memory is incremented.

3.6 DT32bus

Basic DT32bus operation can be tested by loading known data into a selected ROCO's FIFO and then enabling the Event Builder. The data should appear in the EB and can be inspected using the snapshot mechanism. The unselected ROCOs really need to be placed in a "bypass" mode to prevent them from placing data they may have in their FIFOs onto the bus. It's the nature of the DT32bus, with signals regenerated at each station, that fault diagnosis is inherently limited if we do not allow ourselves the liberty of recabling the bus and exchanging modules. For example, a failure to receive data at the Event Builder could be caused by a failure of any of the stations on the bus to propagate the bus signals correctly. This behaviour could also suggest a fault in the EB's upstream data interface. The Histogramming unit could be used to check independently for bus data cycles.

3.7 Readout Controller

Selftest feature The ROCO supports a test mode in which data may be loaded into its FIFO and subsequently transmitted over the DT32bus. This feature can be used to exercise FIFO emptying and DT32bus interfacing. The data must be checked when it arrives in the Event Builder, so a working EB has to be assumed.

VXI Readout Each ROCO drives a readout daisy-chain running through the set of Ge and BGO (and Trigger) cards in a single crate. Readout can be exercised by disabling all but one card, loading the FIFO of the selected card with data, and triggering the ROCO to start the readout process. The expected result is a subevent captured in the Event Builder. Each Ge and BGO card in the daisy-chain can be tried in turn starting with the card nearest the ROCO. Possible faults are an empty or corrupt subevent and timeout of the readout process detected by the ROCO. Some insight into the latter fault may be had by observing the behaviour of the "BLTACK" VXI line by means of the RM's FALA unit.

3.8 Trigger Unit (TU)

Correct operation of the Trigger Unit is required for testing individual Ge and BGO modules (see below).

Selftest feature The TU supports a test mode in which a sumbus input can be simulated by data transferred over the VMEbus. The event data thus generated can be read out over the VME bus and inspected by the RM processor.

VXI line outputs The VXI lines driven by the TU can (in principle!) be inspected by means of the Resource Manager FALA facility, and timing data extracted from the digitised waveforms.

Scalers Scalers within the TU provide evidence that user trigger inputs are being seen and processed.

3.9 Ge and BGO cards

Quite extensive exercising of an individual Ge or BGO card can be envisaged by disabling its crate's ROCO and using the card's onboard test pulse generator in conjunction with the Trigger Unit for pulsing the "synchro-test" VXI line and supplying suitable Fast Trigger and Validation signals (by setting the TU for single multiplicity events). The resulting data can be read from the card's FIFO over the VMEbus. Hardware Compton suppression can be tested by exercising Ge and BGO channels in pairs. More detailed diagnostic tests using the Resource Manager FALA facility to capture analogue waveforms within modules can be envisaged. However, having isolated a fault to within a single card, it makes more sense to exchange the card. Note that confidence in the Trigger Unit and its signal paths to the Ge and BGO cards can be gained by satisfactory operation of at least one Ge or BGO card in this test.

We would certainly expect test procedures for these cards to evolve from standalone commissioning programs, but we recognise that not all test procedures can be performed with cards in their production environment.

3.10 Other detector interfacing

Interfaces for other detectors and data sources are under consideration. These include a "general purpose interface" (GPI), a NIM–DT32 ADC interface and a FERA–DT32 interface. We would expect to see selftest features, analogous to those for Ge and BGO cards, in these cards as well. To exploit such features whatever format and mechanics the cards are designed in requires a local processor in the role of a networked "crate manager" which can invoke the tests and report the results.

One would prefer not to involve the Trigger Unit (almost certainly in another crate) in such tests, however. A test independent of other modules is always more desirable, provided the necessary timing signals can be generated or simulated effectively.

3.11 Other subsystems

Though not directly part of the EbyE pipeline, other subsystems can critically affect EbyE data flow:

High Voltage control *[Are there any selftest features in the LeCroy HV supplies—anyone?]*

Autofill system This system necessarily has feedback mechanisms (e.g., N_2 exhaust vent temperature sensors, bias shutdown sensors) which can be used to monitor proper operation.

4 Setup consistency checking

This section addresses the question "if the system passes the confidence test and the fault-finding diagnostics fail to find any faults, yet the user still cannot acquire data to his satisfaction, what then?" Under these circumstances we must assume that the EuroGam system is in working order but the user has done something within his competence (!) to prevent it taking data. We will examine ways of diagnosing his mistake.

Beam problems We know of no computer-accessible way of telling whether beam is reaching the target.

Detectors All Ge and BGO detectors have computer-controlled bias supplies. We envisage a tool which highlights those detectors without bias voltage within some acceptable working range. This must take into account the possibility that bias has been shut down on a "warm" detector.

Preamplifiers Detector preamplifiers are also interlocked with the detector bias supplies—when a preamplifier is powered off the interlock prevents bias being applied to the corresponding detector.

Cabling A cabling fault between detectors and VXI cards cannot be ruled out. This could be diagnosed by disabling all detectors but one in turn and observing which channel (if any) the data appears in.

Trigger logic Users can determine the logic conditions under which Fast Triggers and Validations are generated. Whether they appear or not can be checked by monitoring the statistics gathered internally by the Trigger Unit.

Ge and BGO card setup The data acquisition cards contain several parameters per channel that critically affect the operation of the cards. We should implement some sort of check of the user's setup, highlighting inconsistent and "abnormal" settings. Wherever possible this should read the user's settings direct from the Ge and BGO cards. Settings must also be consistent with those in the Trigger Unit.

Timing considerations Using the Resource Manager FALA facility it is possible in principle to capture critical signal waveforms from VXI lines and from within data acquisition modules. Some automatic or semi-automatic check that these are within specification should be feasible.

Event Building and Sort Processing Apart from the essential syntax checking of a user's program there is little one can do in general. Possibilities include analysing a program to answer questions such as "do all paths through the program do something 'useful', e.g., increment some spectrum?" and "which spectra might be incremented by this program?" A more general aid might be an offline program emulator. This would take a sort program and trace its execution (listing spectrum increments, for example) given a specific input event. Such a tool might help elucidate sort language semantics for tyro users or provide a debugging aid for complicated programs.