Monitoring Protocol

24 July, 2001

1. Introduction

During normal data taking, readout of monitoring data is initiated when the *collect status* bit is set in the *L1 Trigger Qualifier* (L1_QUAL) word received by the FRC as part of the SCL data. A description of the SCL data word and L1_QUAL can be found in the FRC document prepared for the STT review[1]. Monitoring data is collected from the various cards in the system by the in-crate CPU, which then passes this information on to the Trigger Framework(?). Each card in the system decides whether to prepare its monitoring data for collection based on the L1_QUAL word in the header of the T/R data from the FRC. The main actions of the monitoring process taken by the FRC, the CPU and the other cards in the system are shown in Fig. 1.

2. Implementation of Monitoring Protocol

The monitoring task in the STT is initiated in the FRC (by the Buffer Manager) and controlled mainly by the CPU after that. Given below is the monitoring sequence as we foresee its implementation in the STT. This information is also repeated in Fig. 1. In the list, *DB* is used to denote a generic logic daughterboard (FRC, STC or TFC).

a) **FRC (BM):** Get the monitoring request from the Trigger Framework. This is signaled by the *collect_status* bit being set in the *L1_QUAL* word of the SCL information.
This task is done by the FRC, since it is the first place that SCL information arrives at the system.

b) **FRC (TRDF):** Pass monitoring request to the other cards in the system.
This is done as part of the T/R Data header.

c) **DB:** Interpret *collect_status* bit in T/R Data.
Each card must decode the L1_QUAL *word in the T/R Data header.*

d) **DB:** Each card collects monitoring data, puts it into a VME-accessible register.
It is up to the individual cards how they can collect and store their monitoring information most efficiently.

e) **DB:** Each card signals to the CPU that its monitoring data is ready.
*The CPU is informed through VME-accessible registers (*MON_DONE[i]*) that are set to one when the source has finished collecting monitoring data.*

f) **FRC (BM):** Reserve the VME bus for the CPU to collect monitoring data.
This multi-step process is only begun after the FRC has collected its monitoring data so that the CPU need not hog the VME bus for longer than is absolutely necessary.

i. Check for the VBD or SCL Init using the VME bus. This is accomplished by reading the arbitration registers in the BM**Error! Bookmark not defined.**.
ii. Wait for the VME bus to clear if it is busy.
iii. Set the *mon_req* arbitration register to 1.

g) **FRC (BM):** Signal the CPU that monitoring data is (or will be) available.
Send a monitoring interrupt to the CPU.

h) **CPU:** Collect monitoring data from each board (*i*).
This could be done using a simple loop over all the boards in the crate. The loop should start with the FRC since it is guaranteed to have monitoring data ready.
i. Wait for MON_DONE[*i*] = 1.
ii. Read monitoring data for board *i* over VME.
iii. Set MON_DONE[*i*] = 0.
iv. Repeat for board *i*+1

i) **CPU:** Inform FRC (BM) that monitoring data is all collected.
Set register MON_ALL = 1 *in FRC (BM).*

j) **FRC (BM):** Release VME bus.
Set mon_req *arbitration register to 0.*

k) **FRC (BM):** Finish up.
Set MON_ALL = 0.

VME-accessible control registers that are required in this scheme are listed in Table 1.

Location	**Name**	**Description**
BM	*MON_REQ*	Indicates that VME is being used by Monitoring. Also used to signal CPU if interrupts are not used.
BM	*MON_ALL*	Indicates that CPU has finished collecting monitoring data.
Source *i*	*MON_DONE*[*i*]	Indicates that monitoring data for this source is ready for readout
Source *i*	*MON_START*[*i*]	In CPU-initiated monitoring – requests that this source write its monitoring data to its readout register
Source *i*	*MON_DATA*[*i*]	Block of VME-accessible memory where monitoring data is stored.

Table 1: VME accessible registers used in the control of monitoring.

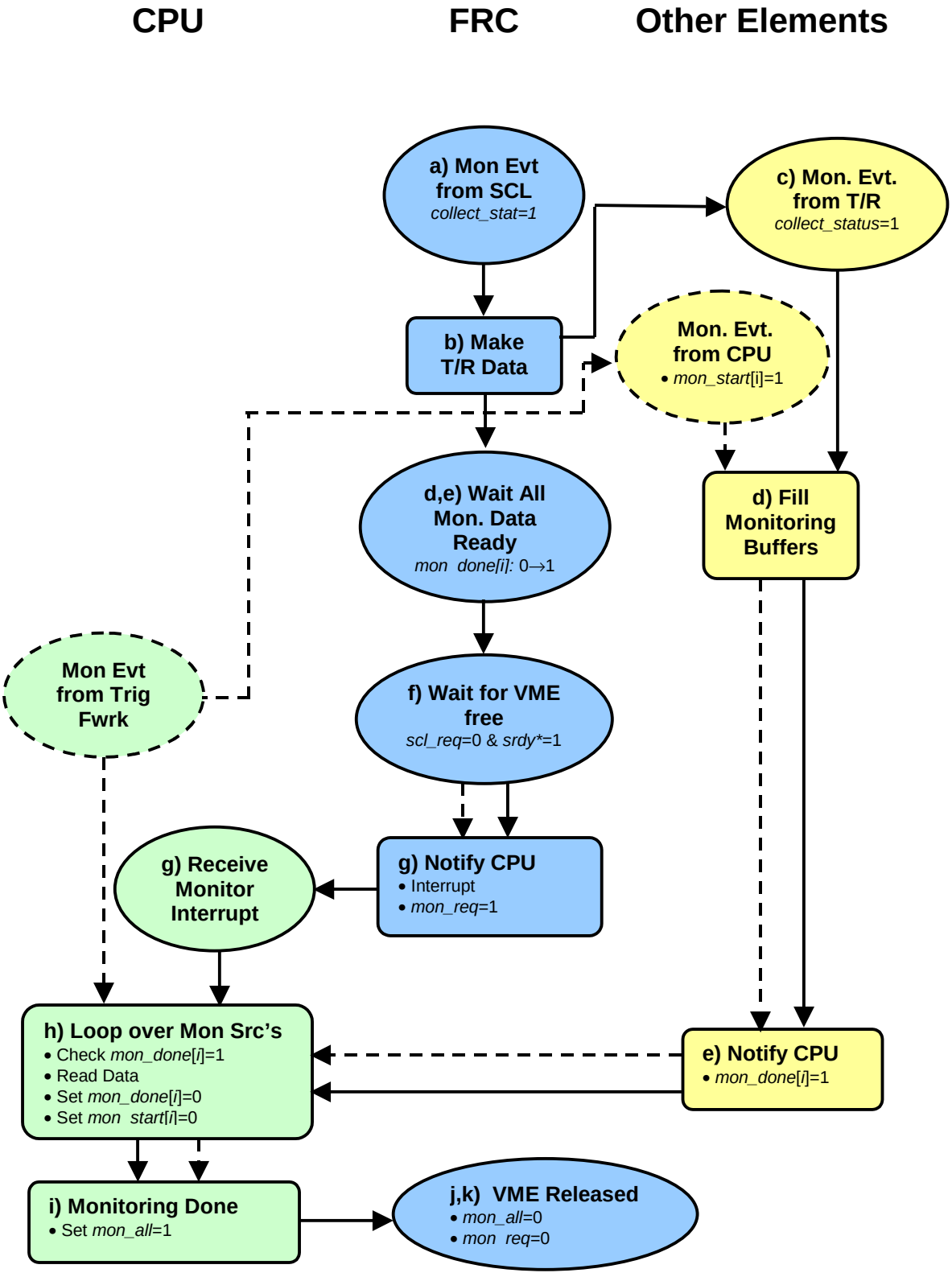

Figure 1: Flowchart for collection of monitoring data. Letters refer to steps described in the text. Dashed lines indicate CPU-initiated monitoring.

3. Interrupt Issues

The BM informs the CPU that monitoring data is ready to be collected by a VME interrupt. Since the BM talks only to PCI, it issues a PCI interrupt that is then passed to VME by the Universe II chip.This is also how SCL_INIT is transmitted to the CPU, requiring (at least) two separate interrupt lines to be used.

Note that the Altera MegaCore PCI interface only provides one interrupt line. The extra line must be included in the local-side logic designed by us.

4. CPU Initiated Monitoring

It is also necessary for the CPU to be able to initiate a monitoring event without the *collect_status* bit being set in *L1_QUAL*. This happens in two situations:

a) at the end of run, when the trigger framework requests the collection of monitoring data, and
b) if problems are encountered and a snapshot of the system's status is desired.

A possible complication in both of these cases is that the CPU cannot simply use the VME data lines to inform everyone of this request whenever it wants – the VBD may be hogging the bus.

In the present end-of-run monitoring collection scheme (case a), however, all previous events have been cleared from the system when an end-of-run monitoring request is passed to the CPU[2]. This means that all VBD activity will have finished by the time the CPU gets its monitoring request – so no special arbitration is necessary. The CPU does have to notify the rest of the system to latch its monitoring information though. This can most easily be accomplished by the CPU writing to a register (*MON_START*[*i*]=1) in each of the monitoring data sources. The sources would interpret this register being set as equivalent to *collect_status*=1 and would fill their monitoring data registers. The rest of the monitoring would proceed as normally. This scheme is outlined in Fig. 1.

Monitoring on request (case b) is even more straightforward. This is only done when there are problems and the run is paused, bypassing the possibility of conflicts with the VBD. Additionally, what we want in this case is not so much the standard monitoring information as a detailed system status. This is most easily obtained by designing a simple program that runs on demand in the CPU and reads all VME-accessible registers in the system.

References

[1] Q. An, et al, *The Fiber Road Card in the Silicon Track Trigger*, (14 February, 2000) http://www.nevis.columbia.edu/~evans/stt/frc/frc.pdf
[2] Jim Linnemann, statement in the STT Engineering Meeting (28 July, 2000) *to which he will be held come Hell or High Water*