November 4, 1994

STAR Trigger-DAQ Interface Specification Version 1.0

M. Botlo, H., Crawford, Leo Greiner, M. LeVine, V. Lindenstruth, C. McParland

Sign-Off Page

The STAR management and group leaders agree to the definition of the interface between the STAR trigger system and data acquisition system as defined in this document. Further changes require the written agreement of all involved parties.

Jay Marx

Richard Jared

Henry Crawford

Michael LeVine

Change Log

November 4, 94	First version accepted

1 Introduction

The interface between DAQ and Trigger depends on the architecture which is sketched in figure 1. All detectors send raw data to their individual DAQ front-end boards. One important aspect of the design is that, due to the space and accessibility constraints of a collider detector, all intelligent components of the DAQ readout chain were moved into the counting house. Consequently the raw data is sent with unidirectional high speed fiber optic links from the detector to the counting house. Trigger actions are signalled to the DAQ front-end systems by state machines on the readout boards of the various detectors using special command words passed from the front-end electronics through these fiber links.

The requirements for the various components shown in figure 1 are listed together with a brief justification in chapter 2. Chapter 3 defines the implementation and handshake mechanism of the shared memory interface between Trigger and DAQ (dark block in figure 1).

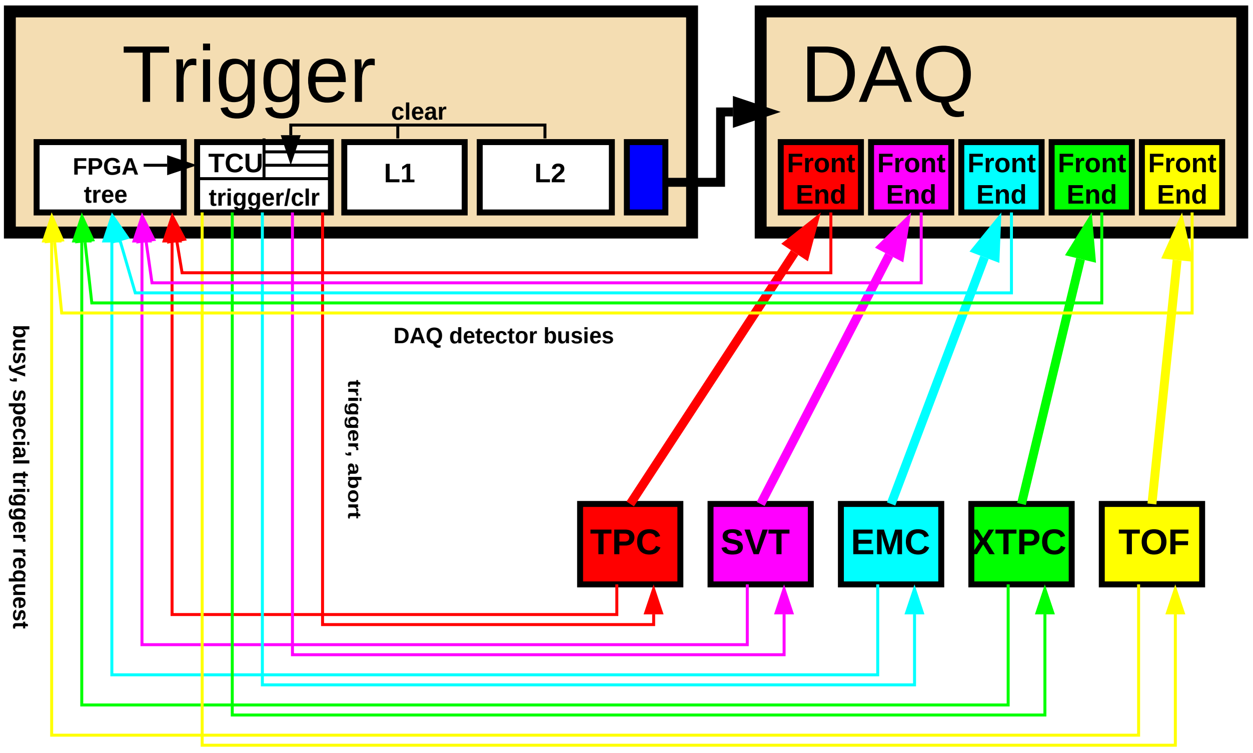

Figure 1: An over all view of the STAR detector, Trigger and DAQ systems. There are other systems like Slow Control and Experiment control that interface with these systems as well. They have been omitted for simplicity. The interface between Trigger and DAQ mainly consists of a shared memory and the DAQ-detector busy signals.

2 Requirements

2.1 DAQ detector busies

Requirement:

DAQ produces for each detector system an individual busy signal. The logical OR of this signal with the busy of the given detectors front-end electronics busy forms the detector busy used by the Trigger system.

Justification:

STAR does not have a common dead time. Triggers may involve only one particular detector (like EMC) or a coincidence of several detectors. The DAQ front-end systems implement elasticity buffers that may store a variable number of events. Consequently DAQ front-end boards associated with one particular detector produce a summary busy signal indicating that all internal buffers are filled. The logical *OR* of these busy signals is the DAQ detector busy which is produced by DAQ and sent to the trigger system as one signal as indicated in figure 1. Another busy signal produced by the detector front-end electronics indicates for example that the given detector electronics is busy digitizing an event. Consequently the trigger system would receive two busy inputs per detector. One for the readout boards on the detector and one for the DAQ front-end in the counting house.

Note:

Since all DAQ front-end boards have their individual detector busies it is possible to read out fast detectors like the EMC while other STAR systems like the TPC are busy. Consequently no additional global DAQ busy is needed for flow control purposes. If for example no data could be read out due to a tape problem the DAQ event builders would simply stop reading out the front-end systems. These systems would assert their busies when their buffers are full preventing any further triggers and consequent data overruns.

2.2 DAQ notification

Requirement:

DAQ is notified by Trigger only if a given event will not be cleared by L1 or L2.

Justification:

Due to this rule L1 and L2 clears are completely transparent to the DAQ system. No mechanism is required to forward any L1 or L2 aborts to DAQ. This requirement greatly reduces the complexity of the interface between Trigger and DAQ. Its sole functional requirement is to supply DAQ with information about the currently available events for readout.

Note:

The delayed notification of DAQ does not impose any performance burdens since each detector that needs to run at higher rates than the TPC can buffer its TPC related data.

It was stated before that the detector readout boards forward clears to the DAQ front-end boards as special commands. However this does not involve DAQ in a logical sense since the receiving state machines on the front-end boards will be able to perform that clear without any processor intervention.

2.3 Event Descriptor

Requirement:

Supply event description of all (L2 accepted) active events. The following words are required in the event descriptor:

❶ Bunch Crossing Number (64 bits)

The 64 bit bunch crossing number uniquely identifies the time throughout the lifetime of the experiment. Its upper bits are slowly changing and can be derived from the absolute system time.

❷ Trigger Command (16 bits)

This bit field defines which detectors were involved in the given event and have to be read out by DAQ.

❸ Trigger Word (16 bits)

This word defines the cause for firing a trigger. It specifies the kind of the given event like for example high multiplicity, minimum bias scale down event, pedestal event.

❹ Trigger Type (4 bits)

This word is also forwarded to the front-end electronics and from there to the DAQ front-end. It defines 16 different DAQ actions related to that event.

❺ Trigger Token (12 bits)

The trigger token is uniquely linked to an event until it is processed by DAQ. Trigger will not use a given trigger token until token has been returned by DAQ.

Justification:

The trigger system is the only instance within STAR that has complete knowledge about any event. DAQ needs to know which detectors were involved and need to be read out.

2.4 Trigger detectors raw data - Trigger history

Requirement:

Supply the complete raw data of the trigger detectors for readout by DAQ including a detector specific pre- and post history. The pre- and post history is limited to 10 bunch crossings for events in coincidence with either the TPC or SVT and 1 bunch crossing for events which involved only *fast* detectors like EMC or TOF.

Justification:

The raw data of the trigger detectors that caused the given event to be triggered needs to be in the data stream. The trigger detector data of the N_{pre} previous and N_{post} later events is required to determine pile-up. N_{pre} and N_{post} are configuration constants and depend on the detectors involved. The number of pre and post history events required for the TPC depends on the single track vertex position resolution (1cm). Given the drift velocity of the TPC of about 4cm/µs this corresponds to 250 ns or three bunch crossings. Correspondingly the upper limit of 10 as specified is very generous.

Note:

The complete raw data of all trigger detectors is a fixed length data block of 2048 bytes length.

2.5 Trigger summary data

Requirement:

Supply Trigger summary data of the trigger detectors for further use within L3. The size of this data structure is limited to 512 bytes maximum.

Justification:

Results of the L0, L1 and L2 analysis are needed within L3. For example the vertex position is needed by the SVT L3 analysis. The vertex position will be derived from the VPD raw data during the L0 analysis within the trigger system. Correspondingly the results can be made available to requesting L3 processors.

The size of the summary data block needs to be limited in order not to overburden the DAQ network bandwidth.

2.6 Maximum accepted trigger rate

Requirement:

The maximum L2 accepted trigger rate is limited to 1000/sec.

Justification:

Limiting the L2 accepted trigger rate simplifies the DAQ event builder design.

Note:

The highest TPC and SVT trigger rate is 100Hz. However this allows detectors like EMC and TOF to run about an order of magnitude faster if they do not require a coincidence with the TPC.

2.7 Notify DAQ about new events

Requirement:

Support asynchronous notification mechanism to DAQ in case of availability of more events.

Justification:

The DAQ implementation may require the trigger system to interrupt DAQ upon availability of new events.

2.8 Notify Trigger upon readout completion

Requirement:

Inform Trigger when trigger token is no longer needed and all related data structures in Trigger-DAQ interface are readout.

Justification:

Trigger needs to know when a given event is read out. At that point the trigger token that was associated to the given event is free and can be issued again. All shared data structures linked to the built event are free, too, after the event was completely built.

2.9 Trigger Token

Requirement:

Supply a unique 12 bit trigger token to all detectors and DAQ.

Justification:

The STAR detector system will consist of many independent detector systems, which will all have their individual dead time. Any combination of detectors is possible for coincident triggers. The readout time of the various systems varies greatly. Therefore it is possible that during the conversion of one detector another detector is already live and ready to take more data. The STAR trigger system supports to fire triggers to fast detectors even if their previously coincident event with a slow detector is still being processed. Therefore it is very important to have a mechanism that allows to uniquely identify to which event and detector any data buffer belongs. In order to uniquely identify a data buffer with a given event the trigger system will distribute a unique trigger token together with the trigger command itself. One way to make the trigger token unique is to use enough bits to make sure that the same trigger token will never be used again. For example a 64 bit bunch crossing number would be a good example. However cost considerations require the number of bits transferred per trigger to be as small as possible. Another approach is to make sure that the same trigger token is not used again until the event is assembled. In this scenario the number of bits required for the trigger token is defined by the maximum number of outstanding triggers that are allowed by the data acquisition system. An upper limit for this number is defined by the highest accepted DAQ trigger rate (1000/sec) and the maximum DAQ latency (2s). Correspondingly a 12 bit trigger token would be sufficient. The trigger system will reuse a trigger token only if the corresponding event was readout and assembled by DAQ and the appropriate trigger structures were cleared by DAQ in the Trigger-DAQ interface.

3 Implementation

The interface between the trigger system and DAQ is very simple. Its sole functional requirement is to supply DAQ with information about the currently available events for readout. There are no critical timing constraints since the rate of accepted events will be low (on average one event per ms). The simplest data exchange mechanism is a dual ported memory. The two ports of that memory are the Trigger L2 systems and the DAQ system. One dedicated processor (Trigger-DAQ interface controller) will maintain and update all data structures in that memory. DAQ will have one process performing all DAQ related actions.

Figure 2 shows a sketch of the data structures in the Trigger-DAQ interface that accommodate the stated requirements. Most of the memory will be write protected to ensure integrity of the data structures. Only the mail-box and flag regions are writable by DAQ as indicated in figure 2. The latest version of the CVS[1] managed C-include file *trigger_daq_interface.h* specifying the structures in detail will be accessible from the world wide web. It will be uniquely identified with a version code that is part of the CSR[2] space. This correct version code has to be read at start up time by every process accessing these data structures before performing any further actions. In case of a mismatch the program has to terminate with a fatal error.

The only required parameter to access the Trigger-DAQ interface is the base address of the *trigger_root* data structure. It depends on the address mapping of the individual processor and cannot therefore be defined globally. All pointers in all other structures of the Trigger-DAQ interface are with respect to this address. The *trigger_root* structure reflects the main organization of the Trigger-DAQ interface.

The handshake between Trigger and DAQ is basically a two word handshake. The Trigger-DAQ interface will start-up with *Trigger-DAQ mailbox pointer* and the *token return mailbox* being cleared by Trigger.

Any time there is a non zero value in the *Trigger-DAQ mailbox pointer* and a new event available the Trigger-DAQ interface controller will write the base address of the appropriate *trigger_header* structure (with respect to the *trigger_root pointer*) to that mailbox address and clear the *Trigger-DAQ mailbox pointer*. The DAQ processor will respond by updating that pointer if it accepted the trigger request. This handshake scheme prevents multiple requests to overwrite each other.

The handshake for returning trigger tokens is equally simple. DAQ writes the trigger token that is no longer needed to the *token return mailbox* but only if this word is equal to 0. The Trigger-DAQ interface controller responds by clearing that word no later than 100 μs after the posted request. To allow some book keeping within the trigger system of the L3 event abort rate DAQ needs to inform Trigger if the event related to the given token was accepted or rejected by L3. If the event was accepted DAQ returns the token unmodified. If the event was aborted it will return the token times -1. The Trigger-DAQ interface controller will use the absolute value of any value found in the *token return mailbox*. The sign of that value defines if the given event was accepted (>0) or rejected (<0).

1.Concurrent Version System - this system is recommended for software version control by the STAR SOFI group

2.Control-Status-Register

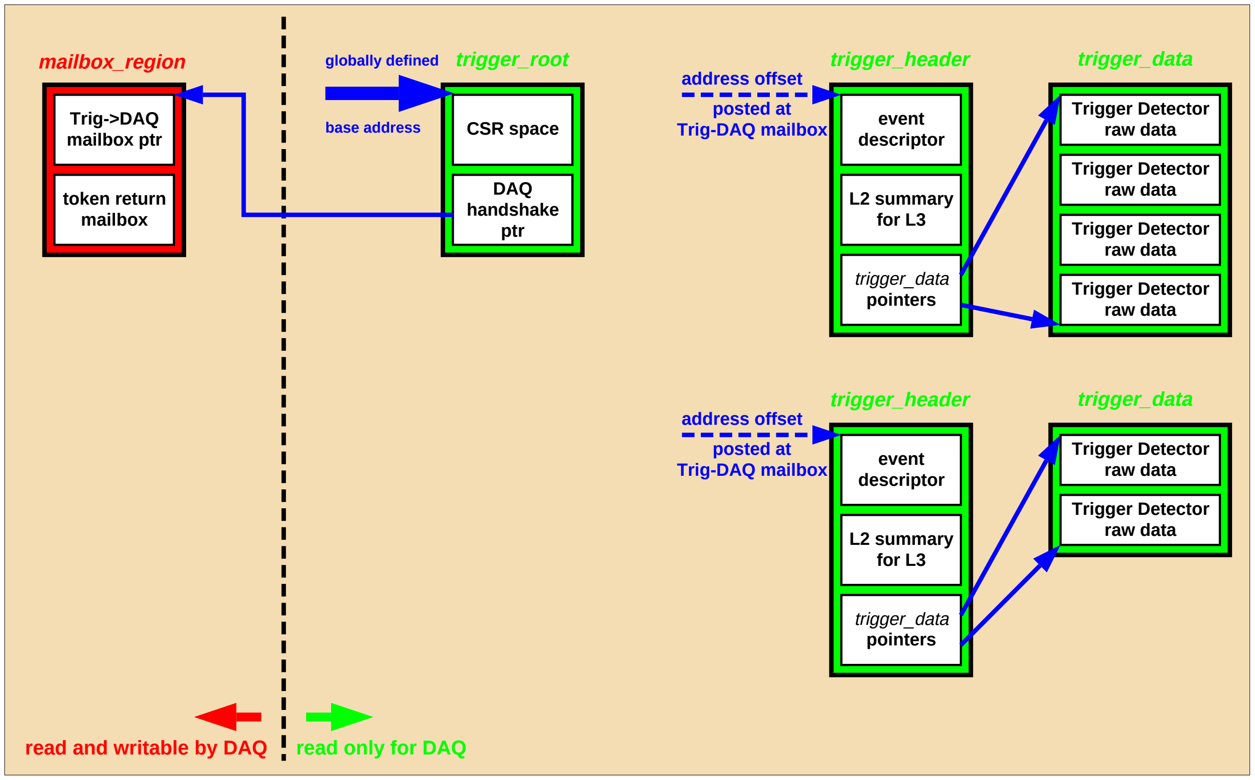

Figure 2: A sketch of the data structures in the Trigger-DAQ interface. The event descriptor field in the trigger header contains the data items listed in section 2.3. The L2 summary field is described in section 2.5 and the trigger data pointers define the data field containing the Trigger detectors raw data including the Trigger history (section 2.4).

Note: all pointers shown here are with respect to the trigger root base address.