CDF/DOC/TRIGGER/CDFR/4453
Version 2.1
9 March 1998

Specifications for Error Handling for the CDF II Trigger/DAQ System

J. Patrick[a], M.P. Schmidt[b], P.J. Wilson[c]
Fermi National Accelerator Laboratory[a], Yale University[b], University of Chicago[c]

Abstract

A specification is presented for the handling of errors within the CDF II trigger and data acquisition system. The focus is on the proper use of the CDF_ERROR* line.

1 Introduction

It is helpful to place the discussion of the handling of errors for the CDF II trigger/DAQ system in the context of the protocol already defined for fast flow–control[1]. Messages sent by the Trigger Supervisor to the front–end and trigger crates control the synchronous flow of data as it depends on Level_1 decisions and the asynchronous flow of data as it depends on Level_2 decisions. Signals returned by the front–end and trigger crates (DONE, BUSY, ERROR) provide the Trigger Supervisor with status information on the flow of data at different stages in the process.

The flow of data can be divided into a few time domains separated by natural divisions. The first period is associated with the synchronous Level_1 activities and can be considered to end when the data for events accepted by Level_1 have been moved into Level_2 buffers. The next period of activity includes the gathering of data by the Level_2 processors and the eventual and asynchronous (with respect to the 132 ns clock) Level_2 decision. The next division occurs with the receipt of the Level_2 decision by the TS and the passing of control for an event to the Scanner Manager and the Event Builder for Level_3.

The fast flow–control protocol defined for CDF II provides an effective means for responding to errors that occur in these time domains.

In what follows we discuss how the appropriate use of ERROR, DONE and BUSY allows the TS to detect and respond to errors occurring in the flow of data at different stages in the trigger/DAQ path.

Figures 1 and 2 provide a schematic overview of the trigger, data acquisition and trigger interface systems for CDF II.

2 Synchronous Level_1 Activities and CDF_ERROR*

The TRACER[2] modules detect CDF_ERROR* on the J3 backplane of the CDF Readout Crates[4] used by most subsystems. The TRACER on detecting the presence of CDF_ERROR* being set to .TRUE. sends the ERROR signal to the ERROR Crosspoints (*i.e.* sets ERROR to .TRUE. on the LVDS return line). The SRC[3] serves a similar function for the SVXII/ISL subsystems.

The CDF_ERROR* line is to be used **ONLY** to indicate errors associated with the synchronous operations of the Level_1 "pipeline" part of the front–end and trigger electronics. This should include errors occurring up to and including saving an event fragment in a Level_2 buffer.

If any system detects a problem indicating that further Level_1 messages will not be properly handled or that the synchronous operation of the Level_1 trigger path has been corrupted, the system should set CDF_ERROR* to .TRUE..

Some examples:

- Detection of loss of link synchronization for a TAXI, Hotlink, Glink, etc. connection falls in this category. This is true whether these links are being used for control messages, Level_1 or Level_2 data; the important point is that they involve synchronous operation with respect to the 132 nsec crossing CLOCK.
- The loss of synchronization as can be detected by using the B0 marker[6] also falls in this category.
- Check sum, parity errors, etc. in the data coming over these links would also qualify, especially if they are happening frequently.
- Also in this category would be the (potential) detection of invalid (or scrambled) Trigger Supervisor messages. For example, the SRC will check that the TS sends no more than 4 Level_1 Accepts without a Level_2 decision.

2.1 Error Crosspoint

The ERROR Crosspoint provides the TS with the .OR. of ERROR for *all* the crates in the partition. The identity of the first system(s) to indicate an error will be latched by the ERROR Crosspoint; subsequently the ERROR Crosspoint can be interrogated to see how many systems have ERROR set to .TRUE..

From the TS point of view, the absence of ERROR means that Level_1 Accept/Reject messages can continue to be sent; the presence of ERROR causes the TS to issue a STOP[1] message to all elements in the partition.

It is recognized that there can be a difference in severity between the various possible sources of ERROR. For example loss of synchronization on a link or corrupted control messages would presumably be considered fatal, in the sense that it is likely that subsequent events would be corrupted. However, the detection of a dropped bit in a data stream may not be considered fatal, in the sense that it is possible that subsequent events will be intact.

The issue of how to handle errors having different intrinsic severity is discussed later. Before pursuing further specifics on ERROR, we discuss the handling of errors for the asynchronous aspects of the trigger/DAQ system.

3 Gathering of Trigger Information for the Level_2 Decision

The formation and gathering of trigger information for the Level_2 decision involves a mix of synchronous and asynchronous activities and the Level_2 decision itself is inherently an asynchronous activity.

Much, but not all, of the information sent to the Level_2 processors is derived during the formation of the Level_1 trigger. It is expected that several types of links (Hotlinks, etc.) will be used to serially transmit the data to the Level_2 processors. This information is sent to the Level_2 processors as a result of receipt of a Level_1 Accept message. In many cases the information sent to the Level_2 processors will also be part of the data residing in the source's local Level_2 buffers, where it waits, pending the Level_2 Accept message, to be included in the data stream to Level_3.

The trigger information sent to the Level_2 processors will be part of what the Level_2 processors will consider as their 'data'. This data will also be stored in Level_2 buffers for inclusion in the data stream given a (Global) Level_2 Accept which in the end comes from the TS.

If a system detects a problem sending its trigger data to the Level_2 processors (loss of sync on a link) there are two possibilities:

1. The problem detected also indicates that the synchronous operation of the Level_1 trigger is corrupted. If this is the case then CDF_ERROR* should be set to .TRUE.
2. The detected problem does **NOT** indicate troubles with the synchronous operation of the Level_1 trigger. The system may try to resend the trigger data, assuming that there is adequate buffering to cope with data from subsequent Level_1 Accepted events. If the number of retries exceeds an agreed upon threshold or if the local buffering required for retries will be exceeded, then, give up and set CDF_ERROR* to .TRUE.

If the failure for trigger data to be transmitted to the Level_2 processors is only detectable at the Level_2 processors (receiving end of a serial link), then it will fall to the

Level_2 processors to report a L2_TIME_OUT to the Trigger Supervisor via a private protocol (under discussion[5]). The TS will treat L2_TIME_OUT as an error condition, and thus send the STOP message, and require servicing by the TM.

4 Transferring Control of the Event to the Scanner Manager

The fast control protocol defined for CDF II provides a mechanism for identifying errors associated with the transition from the asynchronous Level_2 activities to the passing of control for an event to the Scanner Manager. This is the DONE handshake.

4.1 DONE Crosspoint

The DONE Crosspoint requires *all* elements in a partition to set DONE to .FALSE. for at least three (3) 132 ns clock cycles on receipt of a Level_2 Accept message, **AND** then to set DONE back to .TRUE. when the Level_2 buffer specified in the Level_2 Accept message has been freed for use by a subsequent Level_1 Accept message. (See Figure 3 for a schematic timing diagram). If either condition is not met for any element of the partition, a DONE_TIME_OUT occurs (after a selectable time of 2 to 100 msec) and determination of blame commences. The TRACER and SRC modules determine and send the DONE status line to the DONE Crosspoint.

Situations in which the the front–end or trigger crates fail to process a Level_2 Accept message would be indicated by a failure to acknowledge the Level_2 Accept message — that is DONE would not be set to .FALSE.

An example would be a scrambled TS message. For example the SRC will be checking the Level_2 Accept messages to see if they correspond to a buffer for which there has been a Level_1 Accept but no Level_2 Reject.

If the Level_2 Accept message is received, but it turns out that the Level_2 buffer can not be freed for any element in a particular crate, it is expected that DONE for that crate will not be set back to .TRUE.

Reasons for DONE not being set to .TRUE. would include situations in which a problem is detected by a processor in the crate (*e.g.* VRC or TDC DSP) during readout after a Level_2 Accept message. Some specific examples are:

- VME bus errors
- Bad bunch ID or Level_2 buffer number in header word
- TDCs never return TDC_DONE

- Bad word count from TDCs or other variable length devices
- Corrupt data, if the processor looks at it and can tell (not many crates will be doing this)

It seems reasonable that difficulties with the TDC DSP (implying a need for a reboot) would also fall under this category (e.g. the third example above).

Bottom line: If a system can not or will not move data out of the Level_2 buffers, this should be indicated by the system not responding with the DONE handshake.

The DONE Manager (DM; a program in a MVME162 or equivalent) services the DONE Crosspoint and determines whether the timeout is due to a failure at the beginning or end of the DONE handshake and which element (crate) in the partition has failed to respond with the DONE handshake.

The DONE Crosspoint communicates the success or failure of the DONE handshake (for the partition) directly to the TS for each Level_2 Accept message. The TS requires a successful DONE handshake to proceed with sending Level_2 Accept messages for subsequent (queued) events that have been accepted by the Level_2 trigger.

A DONE_TIME_OUT is treated by the TS as an error condition. When a DONE_TIME_OUT occurs, the TS will normally [1] send a STOP message and require servicing by the TM.

The Trigger Manager (TM; a program in a MVME162 or equivalent) services the TS. The TM will communicate over Scramnet with the DM and then Run_Control by Ethernet to initiate actions as required to determine and repair the local cause (crate or board level) for the DONE_TIME_OUT. The TM will also communicate the DONE_TIME_OUT information to the Scanner Manager via Scramnet. Communications between Run_Control, the TM and the Scanner Manager will be used to determine when the TS will send the RECOVER message following a DONE_TIME_OUT.

It is very desirable that the ReadOut Controllers make every effort to send a valid event fragment to the VRB, even if some error has been detected, even if a DONE_TIME_OUT is going to be caused by the error. This is to keep the

[1] However for diagnostic purposes (or other reasons), the TS need not immediately send a STOP message in response to detecting a DONE_TIME_OUT. In principle, the TS can continue to send Level_1 Accept/Reject messages until the Level_2 buffers are full and then send Level_1 Reject messages. This is possible because the TS operation is controlled by independent finite state machines (FSM) that communicate via the Buffer Manager for the Level_2 buffers which resides in the TS.

The Level_1 FSM governs the changing of EMPTY buffers to FULL; the Level_2 FSM controls the changing of FULL buffers to PENDing for Level_2 Accepts or EMPTY for Level_2 Rejects; the ReadOut FSM controls the status of the Level_2 buffer, PENDing → BUSY → EMPTY, during the DONE handshake. The TM can interrogate the TS to determine when the TS has processed Level_2 decisions for all of the Level_2 buffers (or else a Level_2 Trigger time out has occurred.)

Event Building system from getting out of sync and producing a flood of errors which then need to be flushed.

It would be useful to define an error bank ('ERRD') that can be used to flag and identify problems detected during readout.

It is recognized that some errors may be detected during the readout, *after* DONE has been set to .TRUE. For example for the TDCs, DONE will be set to .TRUE. as soon as their internal formatting is completed (thereby freeing the Level_2 buffer). As the TDC data is read out by the VRC it could be determined that the data was bad. In this case information is required in the data stream (*e.g. ERRD*) to flag and identify that the data is corrupt and that the TDCs will need probably need to be reset.

5 Flow of Events to Level_3 and BUSY

The next division in the data flow is between Level_2 and Level_3 activities.

The fast control protocol defined for CDF II provides a mechanism for identifying (at least at a low level) when problems occurring at Level_3 are affecting the flow of data past the Level_2 decision. This is accomplished with the BUSY return line from the TRACER/SRC to the TS.

The TRACER/SRC is supposed to determine the status of the DAQ/SCAN buffers and set BUSY to .TRUE., if necessary. If there are no more remaining DAQ/SCAN buffers in which to store data for subsequent Level_2 Accept messages, BUSY should be set to .TRUE.

If BUSY is .FALSE. the TS is free to send another Level_2 Accept message as soon as DONE is set .TRUE.; if BUSY is .TRUE. the TS is required to holdoff sending additional Level_2 Accept messages. In order to prevent the TS from sending another Level_2 Accept message, BUSY set to .TRUE. must be detected at the TS at least 132 ns *before* DONE set to .TRUE. is detected by the TS. (See Figure 3 for a schematic timing diagram).

5.1 BUSY Crosspoint

The BUSY Crosspoint sends the .OR. of BUSY for the crates in the partition. The TS responds to BUSY set to .TRUE. by holding off the sending of further Level_2 Accept messages. If the TS can not send Level_2 messages the Level_2 buffers will fill up and the TS will send Level_1 Reject messages and deadtime will be incurred. Once the .OR. of BUSY is .FALSE., the TS may resume sending Level_2 Accept messages.

If BUSY remains .TRUE. for a long time (about 1 second), a BUSY_TIME_OUT will occur and the TS will receive a signal to this effect. A BUSY_TIME_OUT is treated as an error condition by the TS. On receipt of the BUSY_TIME_OUT the TS will issue the STOP message.

The TM will communicate over Scramnet with the DM and then Run_Control by Ethernet to initiate actions as required to investigate the local cause for the BUSY_TIME_OUT. The TM will also communicate the BUSY_TIME_OUT information to the Scanner Manager via Scramnet. It is expected that the Event Building system and Level_3 systems will need to be flushed since the occurrence of a BUSY_TIME_OUT will presumably indicate serious trouble with these systems.

If a TRACER/SRC has set BUSY to .TRUE., and it nevertheless receives more Level_2 Accept messages from the TS, it is appropriate for the device to ignore these messages. It is recognized that by NOT responding with the DONE handshake (as discussed above), a DONE_TIME_OUT will result.

6 Summary for DONE, ERROR and BUSY

- **ERROR will be used to indicate problems with the synchronous Level_1 activities, indicating that response to subsequent Level_1 messages is likely to be corrupted.**
- **L2_TIME_OUT will be used by the Level_2 Trigger System to indicate that the assembly of information for the Level_2 decision is incomplete.**
- **DONE handshake will be used to indicate the the receipt of a Level_2 Accept message and the freeing up of the associated Level_2 buffer. DONE_TIME_OUT will be use to indicate that a hangup has occurred during the DONE handshake.**
- **BUSY will be used to indicate (at a low level) potential difficulties with moving events into Level_3. BUSY will be used to throttle the sending of Level_2 Accept messages.**

7 Further Discussion on ERROR

Much of the attention and discussion for board designers is focused on ERROR and the Level_1 activities. In the following, 'system' refers to any system, subsystem or individual module that can generate the CDF_ERROR* signal.

7.1 Rules for using CDF_ERROR*

Rules for implementation are:

1. *All* sources of CDF_ERROR* should be able to locally latch the "cause" of the error and any other pertinent information useful for diagnostics. This is crucial.

2. *All* sources of CDF_ERROR* must be maskable (enable/disable) at the lowest level, at least at the individual VME module (i.e. not just at the TRACER or SRC).

3. If a system is the source for CDF_ERROR* it should, if possible, internally force the equivalent of a STOP (*i.e.* pretend that CDF_HALT* was asserted for that system.)

   The purpose of this rule is to allow a system to capture local information on the source of an error and avoid having that information being lost due to a long stream of errors. Note that the latency for CDF_ERROR* detected as .TRUE. for a module on a crate to result in a STOP message returning from the TS to the crate is on the order of a microsecond, including the propagation delays.

   It is recognized that some systems are not really able to halt the Level_1 pipeline. It is also recognized that following this rule may cause other systems (downstream) to detect an error. However, this is likely to be unavoidable in any case.

4. Once the STOP message from the TS is received, systems should not set the CDF_ERROR* to .TRUE. if they have not already done so (except, as specified below, after the RECOVER message is sent). That is, the STOP message, in and of itself, should not cause CDF_ERROR* to be set to .TRUE.

5. *All* systems should set CDF_ERROR* to .FALSE. upon receiving the RECOVER message from the TS. *All* systems should allow a local VME access to force a RECOVER.

   This rule assumes that the local cause of the error is also removed by this (the RECOVER or reset) action.

   - In many cases it is expected that the actions executed locally on receipt of the RECOVER message will clear the error throughout the partition. This is the simple case. Communications within the Trigger System Interface (between the TM, TS and DM over Scramnet) will allow the possibility of attempting a RECOVER *without* resorting to Ethernet transactions (some have called this 'Auto-Recover').

The DM will keep track of the frequency of errors for each crate in the partition and then inform Run_Control when further intervention (via Ethernet) is required (*e.g.,* when the agreed upon frequency of errors is exceeded.)

- If the RECOVER message is not sufficient to reset the error, then a more serious situation presumably exists and further intervention will be required. In this case, it is expected that the system will leave CDF_ERROR* set to .TRUE. Communications between the Trigger Manager and Run_Control will provide a means to detect and respond to this situation.
- It is possible that a system will find itself unable to properly respond to a RECOVER message, even though it had not set CDF_ERROR* to .TRUE. In this case the system should set CDF_ERROR* to .TRUE. Again, communications between the Trigger Manager and Run_Control will provide a means to detect and respond to this situation.

7.2 TS Response to Failed ERROR Recovery

The TS will (normally) not issue a RUN message until ERROR is .FALSE., DONE is .TRUE. and BUSY is .FALSE. Communications between elements of the TSI (specifically the TS, the Trigger Manager (TM), and the DONE Manager (DM)) and Run_Control, if necessary, will provide this control.

7.3 Classes of Severity for CDF_ERROR*

There appear to be two main categories of severity for CDF_ERROR*.

1. Loss of synchronization, with the expectation that all further data is corrupted. This is fatal and CDF_ERROR* should normally be set to .TRUE. immediately.
2. Occasional dropped bits, check sum, parity errors, etc. with the expectation that further data may still be intact. This is not necessarily fatal unless it occurs too often. The threshold for setting CDF_ERROR* could be governed by a local counter operating within the individual system or module. The threshold count would need to be loadable via VME.

It is not desirable to have 'errors' triggering the DAQ system (via FRED) as part of normal operation. There is a real danger that the higher level DAQ system would become quickly overwhelmed with 'error' data.

We can consider the possibility of having certain 'errors' result in a change in the SCAN LIST to a diagnostic one. Some variant of this under TM/TS control might be suitable and possible. If required, the TM could instruct the TS to change the Readout List (normally established by the Level_2 processors[1]).

In this context, the concept of an "ERRD" bank could be useful for conveying diagnostic information to higher levels. Care must be taken that this information (in the form of an "ERRD" bank or additional banks) would not crash the Level_3 system. The "ERRD" bank could be used by Level_3 to tag events in question and route them to a special consumer.

8 RECOVER

The purpose of the RECOVER message is to instruct all systems that any data remaining in the Level_1 "pipelines", Level_2 or DAQ/SCAN buffers is now considered trashed and that the Trigger/DAQ system is about to be resynchronized with respect to the Tevatron by the subsequent RUN message. Communications between the Trigger Manager, Scanner Manager and, if needed, Run_Control will determine when it is appropriate to send the RECOVER and RUN messages.

Precisely what operations are required in each system in response to a RECOVER message is not specified; the only requirement is that the systems will correctly respond to the RUN message (which will occur no sooner than 10 msec after the RECOVER message, by popular demand).

An example of a special case with respect to the use of the RECOVER message was identified for the (SVXII/ISL) SRC/SVT subsystem. Several Glink connections are used in this system, including the link for transferring Level_1 Accept data between the FIB (Fiber Interface Boards) and both the VRB(s) and SVT (via optical splitters).

It is viewed as preferable if the RECOVER message forced a resynchronization of the various Glink connections in this subsystem **ONLY** if loss of synchronization on one of these links was the reason for the setting of CDF_ERROR* to .TRUE. This avoids unnecessarily resynchronization the Glink connections, as they are believed to operate most reliably if left running, once stable.

Prior to the Fall 97 DAQ workshop, the SRC had knowledge of the status all of the Glink transmitters and receivers in this subsystem, except for the Glink receivers at the SVT. It has been agreed that a dedicated status line will be run from the SVT to the SRC to provide the SRC with knowledge of the SVT Glink receiver status. In this way the SRC can govern whether a RECOVER message will force a resynchronization of the Glink connections. There will be no need to resynchronize the Glink connections if the SRC has determined that the ERROR causing the RECOVER message is due to a source other than the Glink connections within the subsystem.

In any case it will still be possible for a VME access to force a Glink resynchronization.

There may be other subsystems that may require similar treatments in order to optimize the effectiveness of the STOP/RECOVER/RUN sequence.

9 Event Building Errors

The response of the system needs to be defined for the case in which errors are detected during event building and/or in the Level_3 system. The event building system may find that the event fragment word counts are not reasonable or that the event numbers of the fragments are not correct. It may turn out that a VRB has not received all of its event fragments. If the Scanner Manager becomes aware of such errors occurring for a large fraction of events, the Trigger Manager can be notified to request that the TS send the STOP message.

Acknowledgements: We thank J. Bao, M. Campbell, B. Downing R.G. Feild, C. Gay, E. James, M. Kasten, A.J. Martin, K. Ohl, T. Shaw, R. Vidal and many others for their help in defining, refining and establishing this specification.

References

[1] "Trigger Supervisor Protocols for CDF II", R.G. Feild, A.J. Martin and M.P. Schmidt, CDF NOTE 4469, February 1998.

[2] "Specification for a TRigger and Clock + Event Readout Module (TRACER)," T.M. Shaw, August 1997, `ftp://www-ppd.fnal.gov/ett_ftp/CDF_DOCS/TRACER/`.

[3] "Silicon Readout Controller - SRC Users Guide," Harvard Group, January 1998, `http://www.hepl.harvard.edu/maria/src/src_update.html`.

[4] "A Standard Front–End and Trigger VMEbus Based Readout Crate for the CDF Upgrade," T.M. Shaw and G. Sullivan, CDF NOTE 2388, July 1997.

[5] "Global Level 2 Trigger and the TS $\leftrightarrow$ L2 Protocol", R.G. Feild, A.J. Martin, M.P. Schmidt, M. Campbell, and S. Miller CDF NOTE 4519, June 1998. This is largely extracted from "The DAQ Manager's Notebook," R.G. Feild, A.J. Martin and M.P. Schmidt, CDF NOTE 4499, under revision from an earlier 'upgrade' note.

[6] "Specification of DAQ/Trigger Synchronization Checking Protocol and VME Board Header Word," K.E. Ohl, J. Wahl, and P.J. Wilson, CDF NOTE 3145, May 1995.

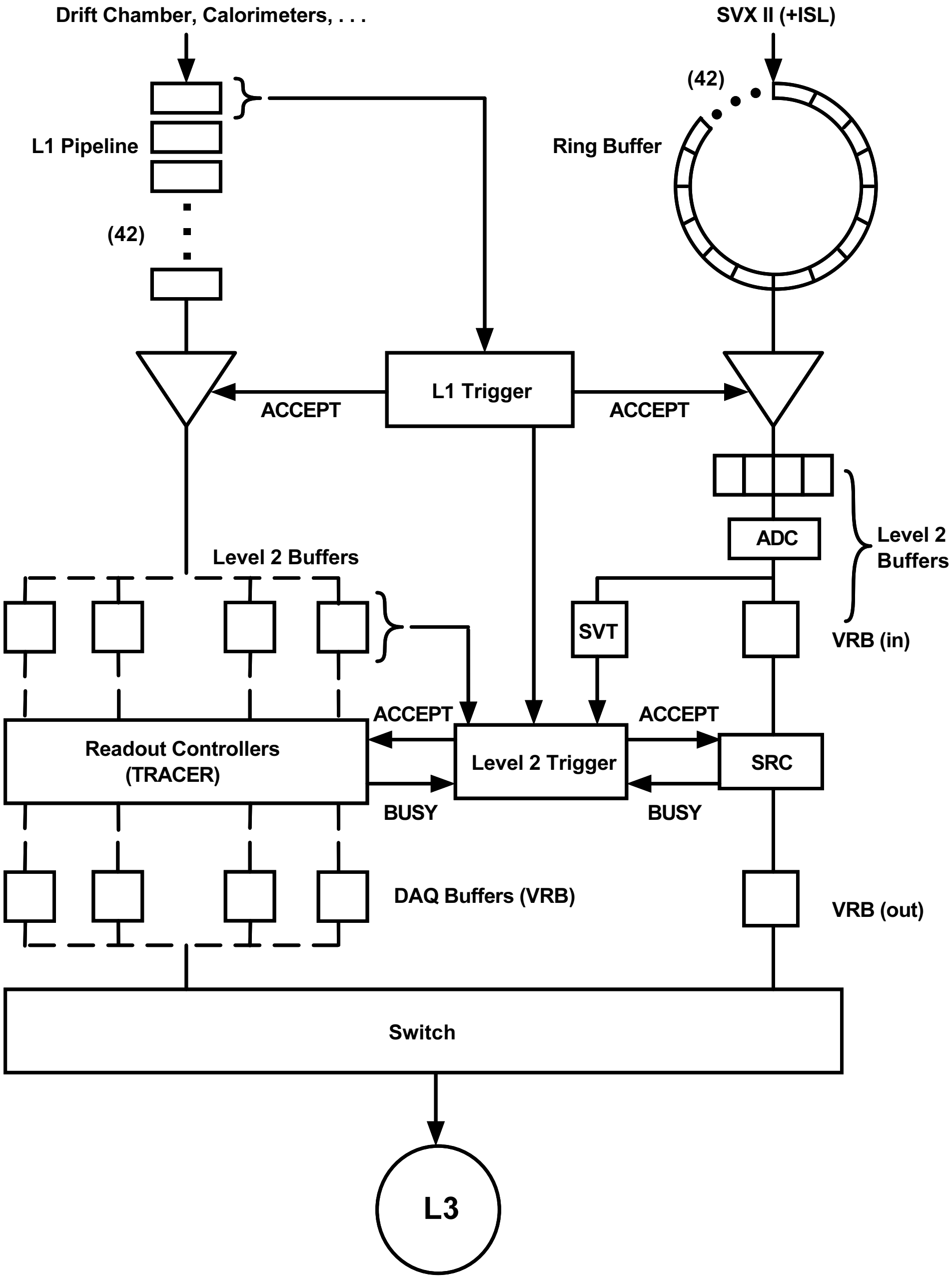

Figure 1: A schematic of the CDF II trigger/DAQ system, showing data flow from the front–end and trigger to the Level 3 farm.

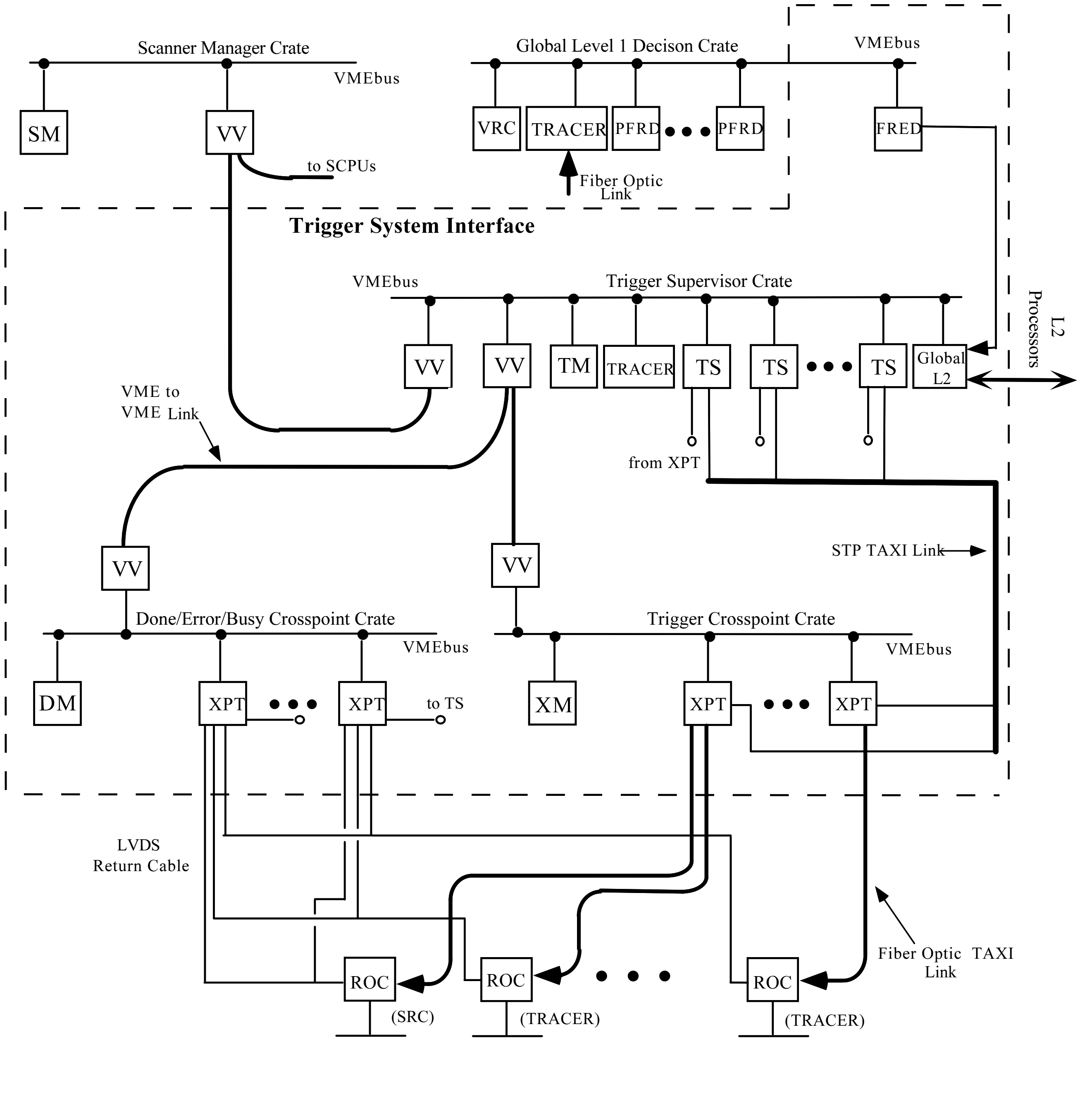

Figure 2: Block diagram of the Trigger System Interface.

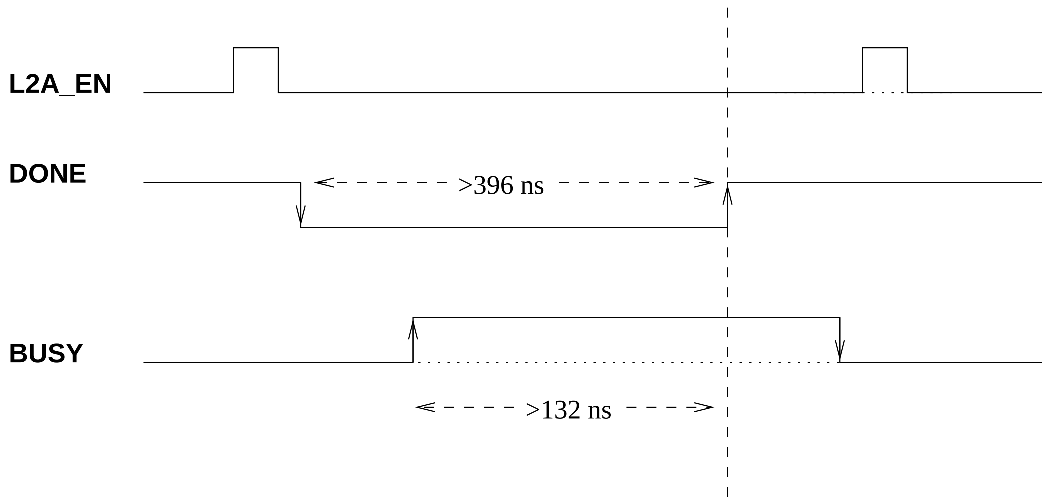

Figure 3: Schematic representation of the relative timing of Level_2 Accept messages (**L2A_EN**) from the Trigger Supervisor and the **DONE** and **BUSY** signals from the TRACER/SRC modules The Figure shows the requirement that the DONE handshake be complete **AND** BUSY must be .FALSE. before another Level_2 Accept message can be sent. Note that BUSY should be set to .TRUE. before setting DONE to .TRUE. in order to holdoff further Level_2 Accept messages. The Figure is not to scale.