SCL init

July 26 2000

Q. An, H.G. Evans, G. Steinbrück

In the first part of this document, we describe under which conditions and how the STT requests an SCL initialize. In the second part we describe how the FRC deals with an SCL init request from the SCL hub.

The STT requests an SCL initialize under the following conditions:

- BX or turn number mismatch (detected by TRDF) (L1_ERROR)
- N consecutive BOE missing ………………………(L1_ERROR)
- N consecutive EOE missing ………………………(L1_ERROR)
- BX mismatch detected by buffer controller between data and BX number that follows the L1_buffer number……………………. (L1_ERROR)
- BX mismatch detected by buffer controller between data and BX number that follows the L2_buffer number……………………...(L2_ERROR)
- Other?

* n is a (small) number set at initialization

Sequence (at TRDF)

To ensure that the data and the SCL information are in synch, the Trigger-Road Data Formatter (TRDF) checks if BX and turn number match. In case of a mismatch, SCL init is initiated immediately with the following sequence:

- TRDF detects BX or turn number mismatch
- Stop processing CTT and SCL data
- Send L1_error to hub (via SCL mezzanine card)

Note that at this point the STT does not have to initialize itself. It waits for an INIT_SECTION request from the hub and reacts to it in the same manner as it would in case of an init request by someone else in the system.

To think about:

- Error logging

When the STT requests an SCL init, it has to provide error logging (see Geographic Section document[i]). An external clock is required for time-stamping events.

Sequence at TRDF for missing BOE, EOE

This case is different from the event number mismatch in that the TRDF does not request an SCL init immediately when a BOE or EOE marker is missed. SCL init is only requested when the number of consecutively missed markers exceeds a preset value. Events with a missing BOE or EOE are tagged by including an error marker in the trailer. The sequence for SCL init after n consecutive misses is:

- If BOE (or EOE) missing[1], increment counter. Otherwise reset counter.
- If counter overflows, stop processing CTT and STT data
- Send L1_error to hub (through SCL mezzanine card)

SCL init requested by a Buffer Controller

The buffer controllers also check the event numbers imbedded in the data against the event numbers that immediately follow the L1_accept or L2_accept messages. For L1 events, the data is checked when it enters the input fifo on the buffer controller. (Input fifos are necessary for error checking, otherwise the L1 data could be transferred directly from the buffer on the daughter board to the buffer memory on the buffer controller). For L2 events, the data is checked when it is transferred from the buffer to the output fifo.

The sequence for a L1 events is:

- BC detects BX mismatch between data and the BX following the L1 accept message
- Stop processing
- Send L1_error to BM (via status on J3 backplane)
- FRC sends L1_error to hub (via SCL mezzanine card)

The Sequence at for a L2 event is:

- BC detects mismatch between the data and the BX number that followed the L2_accept message.
- Stop processing
- Send L2_error to BM (via status on J3 backplane)
- FRC sends L2_error to hub (via SCL mezzanine card)

[1] See Trigger/Road data transfer document for checking for missing BOE/EOE marks.

How the FRC deals with an SCL init request

The FRC provides the interface with the SCL hub and has to forward SCL init requests to the CPU which distributes them to the rest of the STT system. When the FRC receives an init_section request from the hub via the SCL mezzanine card, it has to inform the CPU to coordinate the SCL initialize sequence for the STT. Here we have broken up the SCL init sequences for the various units in the STT.

For the Buffer Manager (BM) the sequence is as follows[2]:

1. INIT_SECTION asserted on input
2. Set INIT_ACK to SCL mezzanine
3. Set L1_BUSY to SCL mezzanine
4. Wait for SRDY or MON_REQ to clear (finish current VME action)
5. Send SCL_INIT interrupt signal to CPU +set SCL_REQ register (polling[3])
6. Receive SCL_READY[i], where i=BM
7. Release SCL_REQ
8. Clear INIT_ACK to SCL mezzanine
9. Clear L1_BUSY to SCL mezzanine

The CPU distributes the SCL init request to the rest of the system. Communication between the CPU and the components is done using two VME readable registers for each unit i, SCL_READY[i] and SCL_DONE[i]. The sequence is:

1. Write SCL_READY[i] signal bits to each VME memory register (BC, DB, MB)
2. Wait until SCL_DONE[i] received from all elements (poll)
3. Acknowledge by clearing SCL_READY[i] signal bits

Note that in this scheme the CPU can enforce the order in which certain units have to initialize by sending a SCL_READY[i+1] only after SCL_DONE[i] is received. This is important for example in the case of the road formatters which should initialize only after the road receivers have been initialized to ensure that no more road data is flowing from the receivers to the formatters. For a timing diagram that illustrates an SCL init sequence, see Fig. 1.

For each component, the SCL init sequence is as follows:

[2] Note that the BM is treated the same as all other units in the STT regarding SCL init: After it informs the CPU that an CL init request was issued by the hub, it waits for the CPU to tell it to initialize.

[3] For more flexibility we propose to implement SCL_REQ as a VME readable register that the CPU can poll in addition to an interrupt.

1. Each Component receives SCL_READY[i] from CPU
2. Input elements clear data
    a. Disable latching input data
    b. Wait n clock cycles w/o valid data
    c. Enable latching input data
3. Processing elements (SCLF, TRDF) reset inputs
    a. Reset input FIFOs
4. Reset all Control/Status/Monitoring registers (except SCL_INIT registers)
5. Reset L1_ERROR and L2_ERROR
6. Each component sets SCL_DONE[i]
7. Wait for CPU to Clear SCL_READY[i]
8. Clear SCL_DONE[i]

Action	SCL_DONE[i]	SCL_READY[i]
SCL_READY	0	0
CPU: SCL_INIT	0	1
Component inits	0	1
Component done	1	1
All done: CPU ack	1	0
Component clears done	0	0
Ready	0	0

Table 1: Sequence for SCL handshaking of CPU and Component i. Note that the CPU clears SCL_READY[i] only after all units send SCL_DONE.

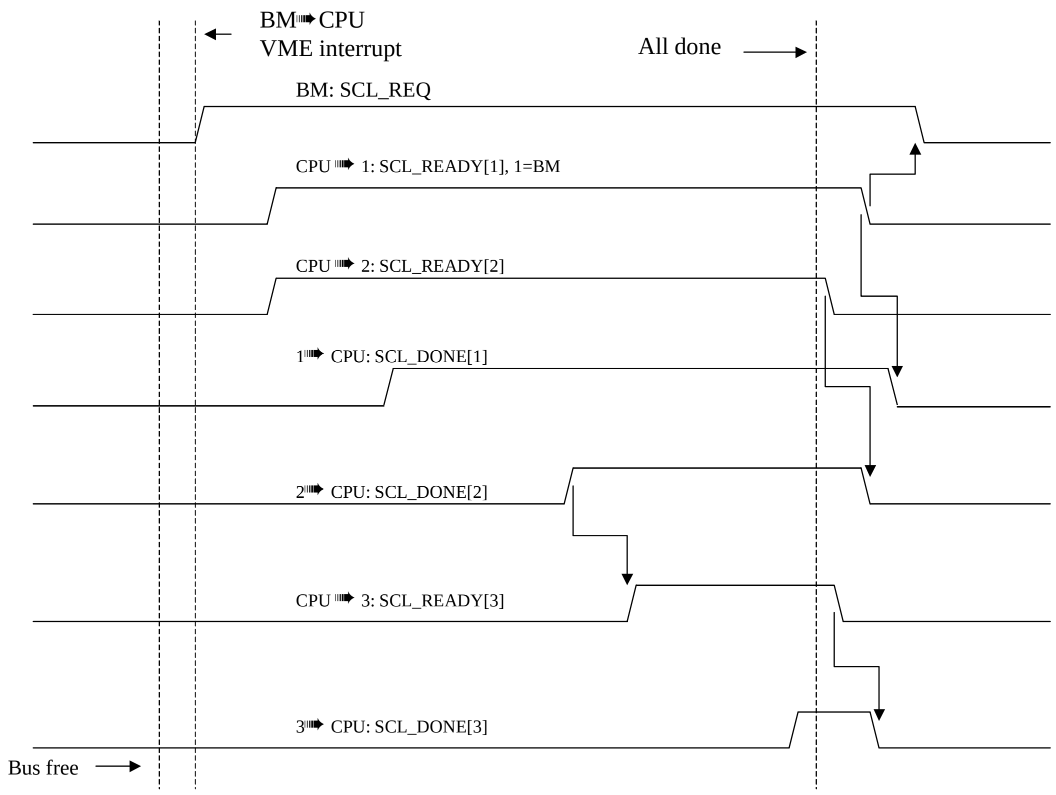

Figure 1: Timing diagram for SCL init handshaking. Here unit 3 receives SCL_READY after unit 2 has sent SCL_DONE. Note that the CPU releases SCL_READY[i] after it has received all SCL_DONE signals. The SCL_READY for the buffer manager should be released shortly after the other SCL_READY signals since the buffer manager clears INIT_ACK to the SCL mezzanine card upon release of its SCL_READY.

Error logging for SCL INIT on the FRC

For error logging requirements see the geographic section document. We have not yet thought about this carefully but are aware of the fact that we have to implement this on the FRC using some kind of VME accessible memory chip that has the required time keeping features.

The record has to be 16 events deep and looks approximately as follows:

Time 7+1 bytes	**BX 1+1 bytes**	**Turn 2 bytes**	**Status 2 bytes**	**Error type 2 bytes**							**User defined 240 bytes**
				L1	**L2**	**BOE**	**EOE**				

Table 2: Error logging record for one event.

References:

[i] Geographic Section Document available at http://www-d0.fnal.gov/~d0upgrad/d0_private/reports/ecb.html.