ATM Forum RBB PHY Contribution AF 98−0305

ATM51 Self Synchronous Scrambler

DJ Greaves Virata Ltd.
A Quayle, R Dearnaley, BT Ltd.

This contribution recommends that the new ATM51 system should use a 25 bit self−synchronous scrambler. Several detailed options are discussed and a recommendation made.

A known problem with the ATM25 standard is that it has strong spectral components at the frequency of repetition of its scrambler. The scrambler has 1023 states and is clocked at 6.4 MHz, putting the lines at 6.26 kHz spacing. This spacing is undesirably coarse and will become doubly so if exactly the same system is used at double rate for ATM51. The spacing would then be 12.5 kHz. A longer scrambler is needed so that the spectrum appears more like white noise for EMC measurement and short wave radio interference.

A disadvantage of a longer scrambler under the current, explicitly synchronised, system of scrambler operation is the time to recover after a transmission error. The scrambler should not be reset more often than it would naturally repeat, so reset intervals need to be spaced further with the longer scrambler.

This contribution recommends that a self−synchronous scrambler is used, so that no resets are necessary. All cell headers can be sent with the X_4 start of cell marker: the X_X sequence need not be used. The suggester scrambler polynomial is x**25 + x**22 + 1 and this should be clocked four times per nibble of data (as in ATM25). This is believed to repeat every 33554430 bit times, so will repeat at a rate of 1.53 Hz when clocked at 51.2 Mbps. The scrambler is never reset, even if an X_X occurs.

Options over Details

Since for correct self−synchronous scrambler operation, the same data must fed into the scramblers at each end of the link, it is necessary to defined the scrambler action when the line is carrying an X symbol and the command modifier nibble directly after the X if this nibble is not scrambled.

If the scrambler is to be clocked when an X is being carried, then a value to be clocked into the receiver in place of an X must be defined. Zero is a suitable value. The transmitter scrambler must then have clocked into it the value which would give zero after scrambling and this can be determined by looking at the current scrambler state. This is more complicated than the alternative option, which is not to clock the scrambler when an X is on the line. We therefore recommend the second option.

Exactly the same discussion can now be had regarding the escape modifier nibble sent after the X, which is 4, 8 or 9. In ATM25, this is NOT scrambled and therefore the scrambler action during this symbol must be separately defined. In ATM25, the scrambler has no input, so a suitable action was just to clock the scrambler. For the self−synchronous scrambler for ATM51, an input would be required and this requires careful attention. Alternatively, two options that do not require this separate definition are: a) not to clock the scrambler during escape modifier nibbles or b) to scramble the nibbles. Since we wish to minimise the gate count increment to an existing ATM25 implementation (where the commands are inserted after the scrambler in the transmit direction), we recommend that the self−synchronous scrambler is not clocked during an escape modifier nibble. This enables the command modifier to be inserted just as before.

Summary: At both ends of the link, a self−synchronous scrambler is used which is not clocked during either the X or the modifier nibble after an X.

Reminder points

It was noted at the last meeting that, for ATM51, an idle pattern of empty cells with GFC=0, VPI=0, VCI=0, PT=0, CLP=1 and valid header HEC should be sent when there are no cells to send. This enables unambiguous signal present detection.

It was noted at the last meeting that the X_9 RDI symbols should only be inserted at cell boundaries. They should not be inserted mid−cell or between the SOC character and the cell header.

David Greaves et al.

−−−−−−−−−−−−−−−−−−−

Richard Proctor Adds:

I have taken the analysis further, to take into account the effects of missing the start of cell indicator and errors at the end of the = previous
cell.

The procedure was to generate :

[End of previous cell]	4 bytes
[Start of cell indicator]	2 symbols
[Random cell header]	4 bytes
[correct HEC]	1 byte

Then this is scrambled, 4b5b encoded, start of cell added, NRZI encoded, errored, NRZI, 5b4b, descramble, check crc, check cell.

The errors were random invertions of the data, a few of these may have occoured too early in the data stream to affect the cell header, a few (particularly when 2 errors injected) could invert the same bit twice, thus eliminating the error.

The resulting data is fed through the NRZI decoder then.

The data starting at the start of cell header is first checked for the start of cell indicator.

It is then checked for false, unexpected X symbols.

Then it is 5b4b decoded, possibly encountering coding violations.

It is then Descrambled. (This is still using the originally proposed scrambler, though I can easily change it).

The HEC is then checked.

If the HEC passes, the result is compared with the original cell, a Bad cell is one that has passed all the checks but is not the original cell. (It may still fail to have a header that relates to an actual in use = VC).
A Good cell is one that matches the original cell. This can be either because the errors where all in the previous cell, or the errors cancelled.

This summary is over 1000000 cells in each test.

Errors	No Start	Unexp X	CodeVio	HEC fail	Bad Cell	Good Cell
1	110732	71031	184423	570225	0	63589
2	207619	116944	264584	395285	1473	14095
3	292728	146099	289072	268784	1130	2187
4	367340	163630	284405	183225	805	595
5	433418	171871	267750	126351	480	130
10	668271	155011	152454	24175	89	0
15	799243	113744	80838	6148	27	0
20	874727	78907	44269	1990	7	0

Thus the proposed scrambler still looks ok to me. =20

Richard