SGS-THOMSON MICROELECTRONICS

APPLICATION NOTE

AN OVERVIEW OF THE SERIAL DIGITAL INTERFACE

SUMMARY **Page**

I - REVOLUTION OR EVOLUTION ?

Revolutions in the technological world are rare, most developments are the result of evolutionary changes rather than instantaneous ones. The impact of new technologies in the programme making environment has followed a similar course - valves gave way to semiconductors, discrete logic to microcomputers to name but two, not to mention 405 and 625, monochrome and colour and many, many more. The changing technologies co-existed quite happily, as indeed many still do today, if for no other reason than the fact that, whilst they might change completely what goes on inside a piece of equipment, its input and output requirements remained largely unchanged. And then came digits !

Whereas the traditional new technology product could be simply 'slotted-in' to the existing environment, generally with an immediate improvement in productivity or performance, and most likely both, digital implementation brought with it the possibility, and indeed probability, of a need to change the world outside as well as that within. Some digital developments, of which the D-2 composite digital VTR is a good example, followed the traditional approach of interfacing directly with the existing analogue environment. Many others however, even if able to do so, could only be used to their fullest capabilities in an alldigital environment.

I.1 - The Digital Island

Nowhere has the impact of digital technology been felt more fully than in the world of video post production. The demand for graphics, animations and related sequences of ever increasing complexity has brought with it rigorous demands of the equipment, of which that most frequently discussed is the need for transparency. A typical animation, such as a programme title sequence, is the result of many video signals superimposed on top of one another, the concept of layering. Rarely is it possible to bring all the layers together in one pass, consequently many generations of re-recording are needed to build up the final sequence.

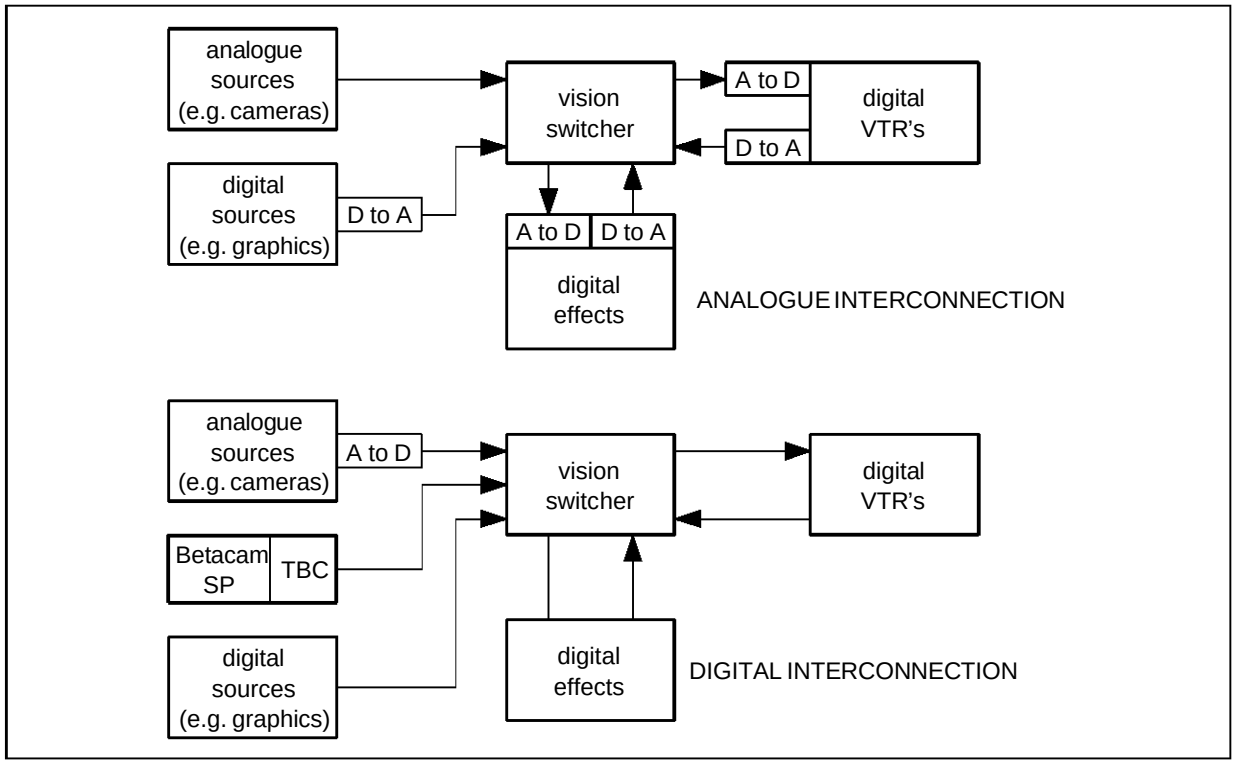
Four different types of equipment will generally be used : graphics origination (a paint system), a vision switcher, a digital effects device and, of course, several VTR' s. Whereas the individual components of the system offer transparent signal processing internally there remains the problem of how they interconnect.

Analogue interconnection, whilst a relatively simple process, introduces degradation at every transition from analogue to digital and back again. Of course there has not always been much choice, as the digital vision switcher is only now becoming commonplace. Figure 1 shows a comparison between two nominally identical post production suites, one using an analogue switcher and the other digital. Notice the number of analogue/digital transitions involved in just one pass through the analogue based system, not to mention the number of A-to-D converters needed were the digital vision switcher to be used with analogue interfacing.

Of course the example of Figure 1 overleaf only takes into account two possibilities of what is, in fact, a four way situation. Not only is there the option of digital or analogue, but also composite or component. As many digital effects devices must operate with component signals there may well be additional sources of degradation in the coder/decoder combinations needed with composite interfacing.

Once all the equipment is implemented digitally it might seem foolish not to ensure that all interfacing remains digital as well. On the surface this would appear to be straightforward, surely less demanding than returning to analogue. The requirements for interconnecting digital equipments are both simpler, and yet more difficult, than the equivalent analogue . Whilst the problems traditionally associated with analogue signals, such as frequency response and non-linearity are less of a problem the range of frequencies is generally rather greater, as is the number of interconnections. This latter point arises, of course, because the digital signal is typically a parallel one, having at least 8 bits plus a clock.

Figure 1 : Comparison of Analog and Digital Interconnection



With many post-production suites operating with component signals (a digital effects device must process the signal in component or primary form) three digital signals are involved. Even if some form of multiplexing is introduced to combine all on to one link, as with CCIR 656, there are still eight interconnections, as well as clocking information. Interconnectionis no longer simple, the single coax with its associated BNC or MUSA connectors is replaced by a multicore cable and multiway data connector. The implementation of even a simple patch panel is a task not undertaken lightly, and limits are imposed on the separation of equipment, owing to such problems as degradation of the data streams, and bit-slipping, caused by propagation differences between the different data bits such that all the bits of a data word do not arrive at the destination together.

As a result of these difficulties the use of digital equipment in an all-digital environment has generally remained in small, largely self contained units, and gave rise to the concept of a digital island in an otherwise analogue world. Increasingly however the programme maker sought ways of interconnecting the islands and, importantly, ways in which he could take advantage of the existing infra-structure represented by conventional cable routes. The only answer was serial data transmission, but how could this be achieved with a transmission requirement well in excess of 200 Mbits/second?

I.2 - Serial Interfacing

The idea of a serial interface for digital equipment has been with us for many years. For a long time however the only possible solution appeared to involve the use of fibre optic links. Whilst advantageous in many respects such a solution was hardly likely to appeal to the programme maker with a considerable investment in conventional distribution systems, not to mention the problems involved in developing an optical routing crosspoint. Eventually a more conventional solution emerged in the form of CCIR-656, the serial implementation of CCIR-601. Not serial in the true sense of the word, in that the 8 bits were carried on separate conductors, but the Y, C_R and C_B components were interleaved.

Other problems arose when some of the fundamental concepts of serial data transmission were contemplated. Ideally such signals should be 'self-clocking', that is no external reference to identify the timing of the data bits is needed. The range and occurrence of the serial data patterns has to be carefully controlled in order to optimize the frequency distribution of the serial signal, and minimise any DC component. Many conventional techniques used are difficult, if not impossible, to implement using existing technology at the data rates involved.

II - AUDIO DEVELOPMENTS

One other development which greatly affects the question of signal routing is that of digital audio. The growth of digital audio applications is well charted, spurred on not only by the capabilities of the technology available but also, as always, by the ever increasing demands of the programme maker.

In much the same way as with video, many of the demands are associated with post production work. Multi-track dubbing, track bouncing and audio sweetening all call for the highest possible quality, something which was often severely restricted with the limited signal-to-noise ratio and dynamic range of the analogue VTR. Post-production invariably meant transferring to dedicated audio machines for that part of the work.

The digital VTR has removed many of these difficulties, with a typical complement of four tracks of CD quality audio many users need look no further. However to take full advantage of such machines a digital mixing console is needed. Such consoles are becoming available, but once more the question is raised - how are the VTR's to be routed to the console?

Of course much work has already been done in this area, with the development in particular of the AES/EBU interface. As well as offering all the usual advantages of digital transmission it eliminates one of those problems that has long plagued the programme maker - channel phasing. Because left and right signals share the same transmission path differential propagation delays, with their associated problems of image shifting and compatibility, are avoided.

There would be considerable benefits to be gained were any serial interface able to support the audio data as well as just the video, may be a form of digital "Sound in Syncs".

II.1 - Other Data

Video and audio signals are not quite all the information involved, one other signal is generally required as well - timecode. A post-production suite without timecode is virtually unthinkable, the information it provides is as important as the basic video synchronizing signal itself. It would be a pity if an interface that provided simultaneous connection of audio and video was unable to handle the timecode as well.

III - A PRACTICAL SERIAL INTERFACE

A practical solution to the problems of serial interfacing is now available. Developed by Sony, and based on a number of specially designed LgI chips the interface deals with all the points raised in the preceding paragraphs. The interface combines digital video - component or composite, 525 or 625, NTSC or PAL - with four channels of AES/EBU audio together with vertical interval timecode in the video, as well as the possibility of user data as well.

In spite of a bit rate of 270 Mbits/second for 625 line component working the interface offers separation distances in excess of 300 m using standard studio co-ax and connectors such as BNC's and MUSA's. Unlike many other systems the bit rate is not increased over that represented by the digital video signal itself, however for reasons that will emerge later the interface offers a video signal resolution of 10 bits, rather than the 8 to which most systems are currently restricted.

The next section offers an overview of the interface in very general terms, however those needing a more detailed description, or perhaps a gentle reminder of some of the terms and concepts associated with it, are directed to the Appendices which follow.

IV - THE SERIAL INTERFACE

As shown in Figure 2 the serial interface comprises four main components. At the source digital video and auxiliary data - which includes the AES/EBU audio - are combined, after which data serialisation takes place. These functions are matched at the destination by their complements - de-serialisation and data separation. Each of the four functions is implemented in a dedicated VLSI chip.

In order to see just how the interface is implemented two distinct aspects must be considered - the hardware involved and the nature of the digital signal. The latter aspect is considered first.

IV.1 - The Digital Data Stream

As may be seen from Figure 2 the first task of the interface is to combine the various data streams: video and auxiliary data. The process of combining is dictated by the video information, the other data being 'slotted into' gaps in the video signal. But just where are those gaps?

If we take a look at Figure 3 we can see that a digital component (CCIR-60 1) signal appears to have quite a large amount of unused data capacity. In fact between the end of active video (EAV) and start of active video (SAV) words some 280 unused words exist (268 - 525/60), with rather more during the vertical blanking interval.

The D-2 composite signal represents a very different state of affairs. In order to retain an accurate picture of the SC/H relationship the sync reference edges are encoded together with the burst. This means that for a 625 line PAL signal only some 70 samples, during the sync bottom interval, are available. For 525 line NTSC the figure is even lower (about 60), this is because the sample rate (4 x fsc) is rather lower (14.3MHz as opposed to 17.7MHz) and the sample period correspondingly longer.

IV.2 - Data Resolution

In recent years much discussion has taken place on the subject of the resolution required of a digital video signal. Whilst 8 bits may well be satisfactory for display purposes there are advantages to be gained through the use of a greater number of bits at intermediate stages. One reason stems from the processing of the digital signal, in a switcher or linear keyer for example.

Typically the signal will be multiplied by a constant, for fading, or a second signal for keying purposes. When two 8 bit values are multiplied the result is one of 16 bits, just as two, two digit, decimal numbers yield a four digit product (99 x 99 = 9801) The video effect desired can be described by the 8 most significant bits - but what should be done about the least significant data?

Simply discarding it is likely to result in noticeable errors, however even data rounding to the nearest 8 bit value can also result in imperfections, particularly if several functions are cascaded. By increasing the data resolution such imperfections have a lesser significance, consequently preserving a greater number of bits at least through the signal processing stages offers considerable benefits. This would allow future digital effects devices, for example, to pass an anti-aliassed key to the switcher without any of the artifacts associated with 8-bit rounding. Only after processing in the switcher might the 8-bit value be re-established.

There is, of course, a second key point to consider. With the CCIR-601 signal some 220 quantizing levels are available to the luminance signal, as shown in Figure 5 overleaf. This contrasts strongly with the situation for D-2composite signals. Because the range of quantizing levels has to encompass not only the chrominance component but the sync pulse as well the number of luminance levels is considerably reduced at 147. There is thus a much greater susceptibility to contouring effects, something that could be largely avoided if a greater number of bits were to be used.

Figure 2 : Overview of the Serial Interface

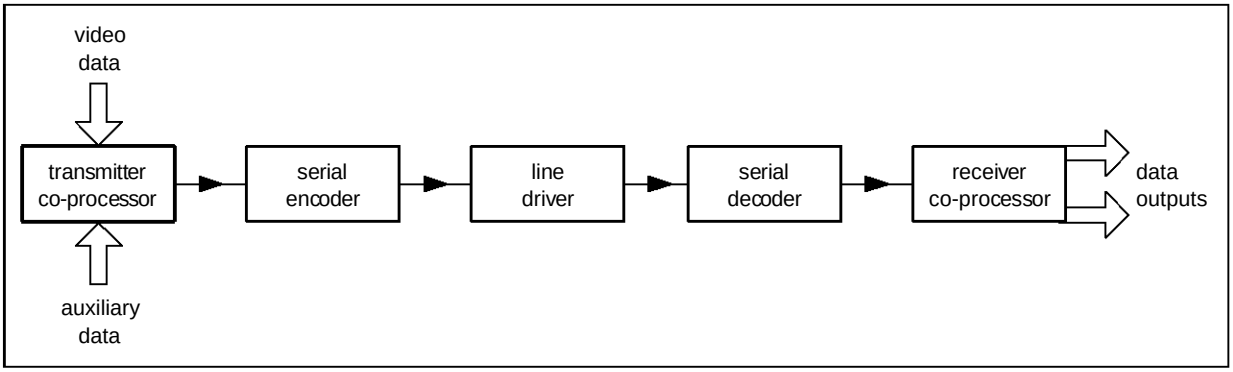
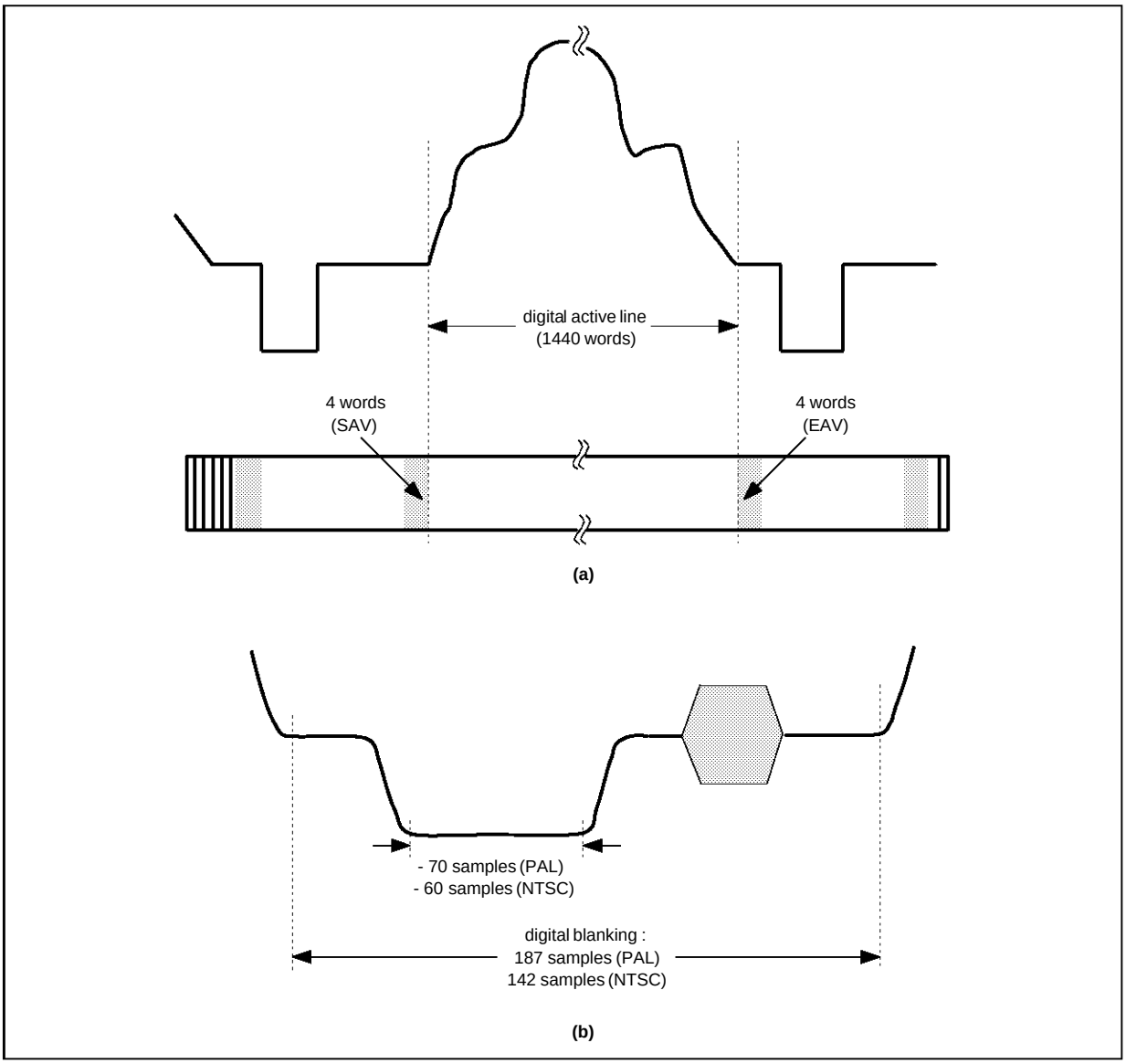


Figure 3 : Horizontal Timing relationships for : (a) component and (b) Composite Digital Video Signals



IV.3 - Identifying the Inserted Data

Figure 3 showed those periods which might be used for inserting non-video data. The next step is to see how these periods are flagged, so that a receiving device can extract this information correctly. The process of data identification is shown in Figure 4, where it will be seen that a number of different data words are used.

The primary data identifier is the TRS - Timing Reference Signal - a sequence of 3 consecutive data words with the specific values of 3FF, 000, 000 (values are in HEX with a word size of 10 bits). These data values should not occur during normal video data, Figure 5 shows the range of data values for both component and composite signals.

In fact four data values are reserved : 000, 001, 002 and 003. These allow the interface to be used with either 10 bit video inputs, or 8 bit values with 'dummy' LSB's.

After the TRS comes a single data word - the Line ID. This provides timing information - two, four or eight field sequence information, together with information about the location of the current line within the field.

The next word is the Auxiliary Data Flag, which indicates the presence of non-video data, it is followed in its turn by the Data ID, another single word used to indicate the type of data - AES/EBU formatted audio data for example. The remaining data words comprise the data itself, formatted in blocks having a maximum length of 255 words, together with a checksum for validity checking. A full description of the block format, and the way in which AES/EBU data is processed, is given in Appendix 2.

IV.4 - Data Serialisation

The output of the combiner is a 10 bit parallel signal comprising video data with embedded ancillary information. This signal is converted to serial form for transmission, two processes being involved.

As explained in Appendix 2 there are many different ways of serializing a data signal. That chosen is usually the one that best matches the resulting signal to the characteristics of the transmission path. There is also the requirement that any clock information needed to decode the signal can be extracted from the signal itself.

The ideal signal in this case is one in which the energy content is well distributed across the frequency spectrum represented by the data rate, which in this case has a maximum value of 270MHz (13.5MHz Y sample rate + 6.75MHz C_R + 6.75MHz C_B x 10 bits) for CCIR-601 signals. Furthermore to avoid signal distortions, particularly those which might affect the shape of the data bits, the signal should be DC free with little energy at the lower frequencies. DC free conditions are achieved if the signal has an equal number of 0's and 1's, whilst the low frequency content is minimized if long 'runs' of all 1's or all 0's is avoided. In fact there is a maximum length of all 0's in this case because an all zero parallel word is reserved for TRS use only.

In order to break up long sequences of 1's and to better distribute the energy spectrum the data signal is randomized, or 'scrambled', following which logic 1's are converted to transitions (+ve to -ve and vice versa). This results in a signal with a large number of transitions, a feature essential to clock recovery. It should be apparent that a clock signal is needed to allow the receiver to determine whether a transition, and hence a logic 1, has been received. Figure 6 shows the process of serialisation in graphical form, a fuller description will be found in Appendix 2.

IV.5 - Transmission

Whilst the signal produced by the serialiser IC can be used for direct transmission over short transmission lines, within equipment for example, co-ax cable will generally be used. For these applications another IC is available offering cable driver and distribution amplifier functions.

Figure 4 : Identifiyng the Inserted Data

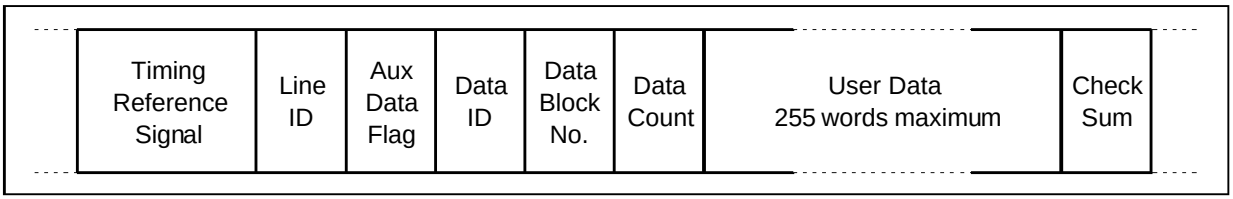


Figure 5 : Range of data Values for Digital Video Signals

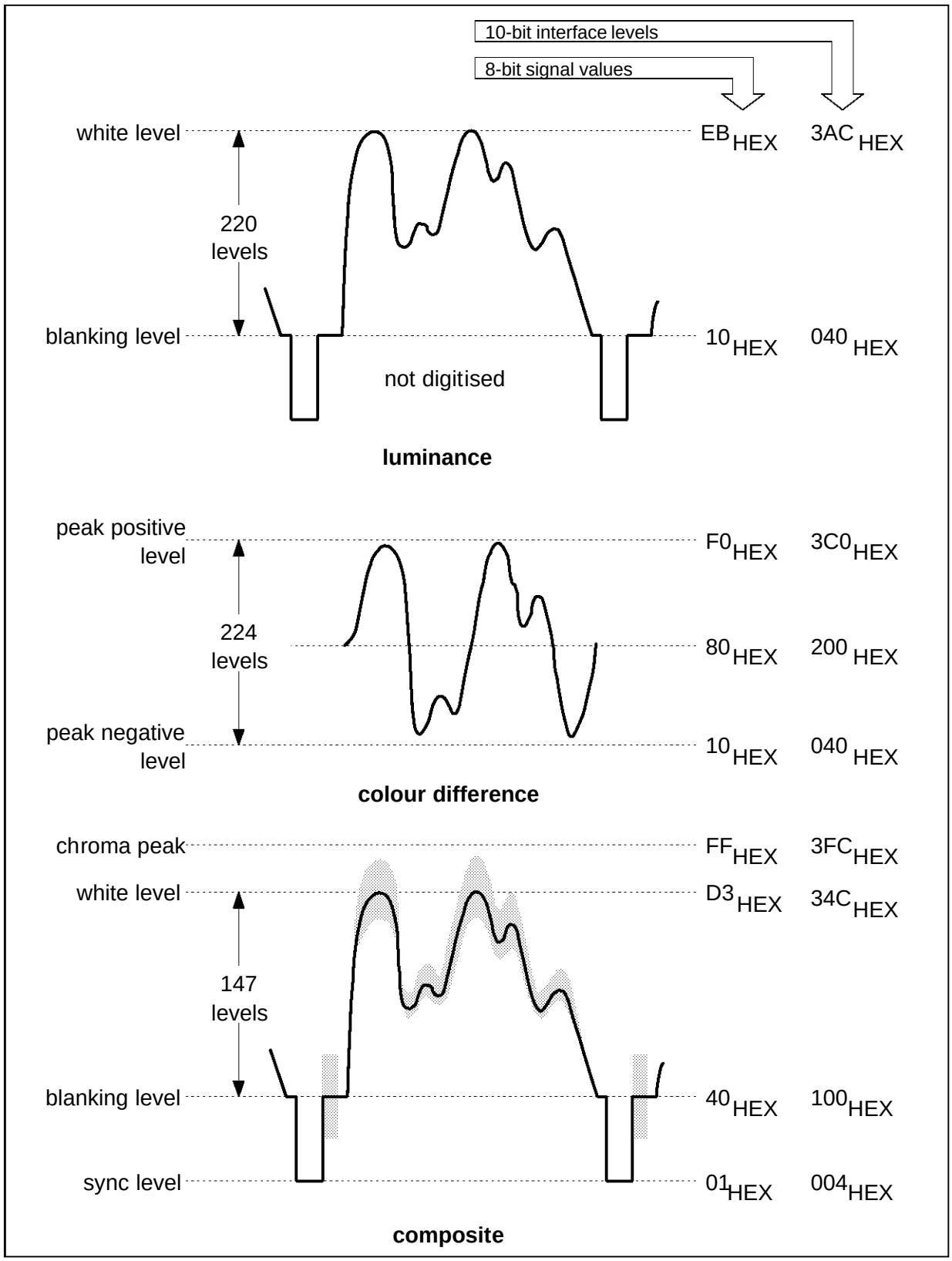
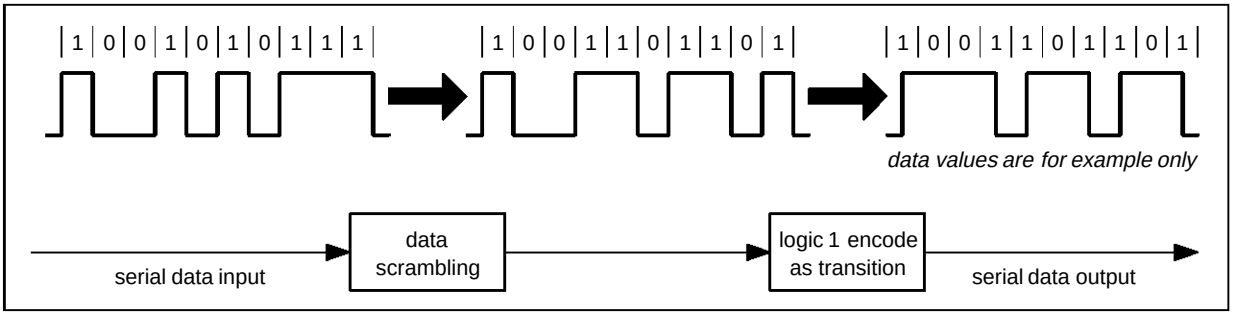


Figure 6 : Data Serialisation



V - AT THE DESTINATION

Not surprisingly the processes involved at the destination are largely the reverse of those undergone at the source, namely de-serialisation and data separation. Before any action can take place however it is essential that a clock signal is derived, in order that the individual data bits can be identified, and the best possible data signal obtained.
The de-serialiser chip has a built in cable equalizer, offering adjustment free equalization for cable runs up to at least 300m. For longer runs the de-serialiser can be used as a signal regenerator, the data signal is detected and re-clocked, but with no change in the serial pattern.

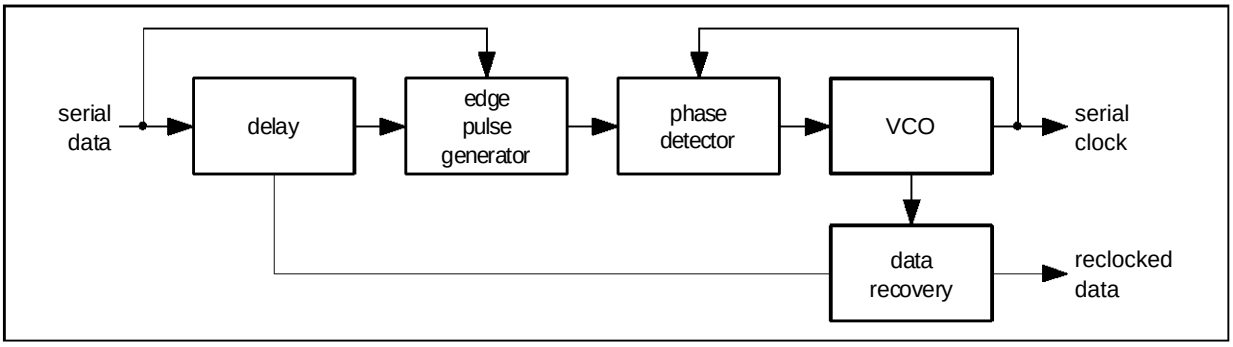
V.1 - Clock Recovery

As explained in Section 4.5 on data serialisation one of the requirements was to ensure sufficient data transitions to permit the clock information to be recovered. This is largely achieved through the encoding of low Tic 1's as transitions. A voltage controlled oscillator (VCO) is synchronized to the data transitions and thus the clock information can be extracted. Figure 7 shows this.
One of the features of the serial interface that has already been described is its ability to operate with both CCIR-601 and D-2 format signals, 525 NTSC and 625 PAL. This is a demanding requirement for the VCO, which has to operate over a 2:1 frequency range, from the 143MHz clock for NTSC composite to the 270MHz needed for CCIR-601 signals.

V.2 - De-serialisation

De-serialisation sounds a simple enough process, each sequence of 10 bits is clocked into a shift register, whereupon the resulting parallel word can be clocked out at one tenth the rate. However there remains the small problem of identifying just which sequence of 10 bits constitutes a complete word, not five from one word and five from the next. Also, of course, the data signal was scrambled at the source. The first step in the de-serialisation process therefore is to unscramble the data signal. This is achieved through the use of an identical logic arrangement to that used by the scrambler, again Appendix 2 has more details.
Identification of the data words is achieved through the use of the TRS information which, we may recall, is a unique data pattern which cannot occur during normal data. The TRS occupies 3 consecutive words, 30 bits in all. The de-serialiser therefore employs a 30 stage shift register in order to store three consecutive words of data, or at least a period equal to three words.
The shift register outputs are compared with the data pattern corresponding to the TRS, as soon as a match is detected a synchronization signal is produced, which in turn controls the timing of the output, parallel, clock.

Figure 7 : Obtaining the Clock Information



V.3 - Data Separation

The remainder of the destination process is concern with extracting the auxiliary data from the combined data stream, and restoring the video component to its correct blanking or sync bottom values.

The process of data separation is fully explained in Appendix 2, however it is perhaps worth mentioning the fact that a number of different outputs are provided, including auxiliary data presence and error information. The error information is quite comprehensive, and reports the result of parity checks performed on the various timing signals and ID's, as well as the data itself and its checksum.

Video timing information, derived from the Line ID, is also provided.

VI - SIGNAL ROUTING

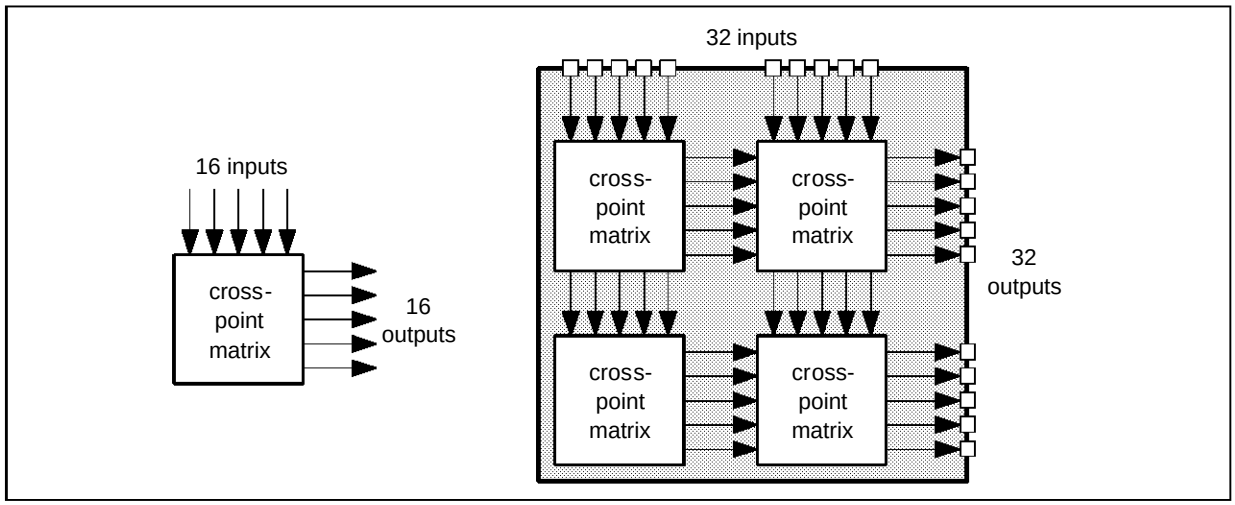
Whilst this article is mainly concerned with the characteristics of the serial signal itself and its hardware it would be incomplete without a mention of the requirements of signal routing. As explained earlier the serial signal can be distributed using conventional co-ax cable over distances of at least 300 m without additional equalization. This means that existing analogue circuits can be used for the serial digital signal. The data rate, up to 270 Mbits/second, is rather higher than that which conventional routing systems, or indeed technologies, are likely to handle.

Here again the power of large scale integration is revealed. In addition to developing chips to implement the interface Sony has also developed a crosspoint chip in order to facilitate the construction of the digital routing system.

A single crosspoint chip offers a 16 source x 16 destination matrix - 256 crosspoints in all. Four such chips arranged on a single printed board make up a 32 x 32 router, whilst an arrangement of four cards yields 64 x 64 and so on up to a possible 256 x 256. Figure 8 shows a typical arrangement.

By associating other matrices a total routing system, including separate audio, tally and control signals, can be assembled. As the serial crosspoint can handle the full bandwidth of the data signal CCIR-601 and D-2 signals can be mixed in any combination. The router control logic must know however which sources are composite and which component, and similarly for the destinations, as it would be inappropriate to route a CCIR-601 source to a D-2 destination, although a number of data converters, which essentially offer coding or decoding in the all digital environment, are under development.

Figure 8 : Rounting Matrix configurations



APPENDIX 1 : DIGITAL INTERCONNECTION CONCEPTS

This Appendix offers a reminder of the basic concepts associated with the processing of signals in digital form, together with a number of those associated with serial data transmission

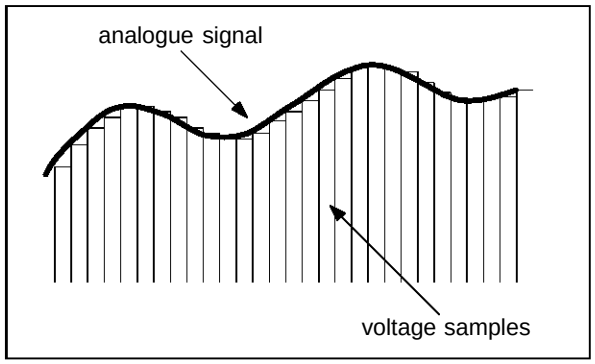
1 - OBTAINING THE DIGITAL SIGNAL

1.1 - Sampling

Sampling is the first step in any digital process. The continuously varying analogue signal is sampled at discrete time intervals, as shown in Figure 9 The result is a sequence of voltage samples which, if occurring sufficiently rapidly, allow the analogue signal to be reconstructed.

The rules which govern the sampling process are very well known, and generally require the sampling frequency to be at least twice the highest analogue frequency (the so-called Nyquist limit), and in practice a little above this, owing to the limitations of the pass filter needed to recover the original analogue component from the sampled version.

Figure 9 : Sampling of an Analogue Signal



1.2 - Quantising

The sampling process converts the timescale from one of infinite resolution to a series of discrete intervals. Although constrained to particular time periods the analogue samples are not restricted in terms of amplitude resolution. This is the task of quantization. Closely associated with the concept of signal coding quantization does for the amplitude domain what sampling did for time.

The signal is only permitted to exist at certain fixed levels as shown in Figure 10.

The number of !evels is, of course, determined by the resolution required, and in turn depends on the way in which the recovered pictures or sound are perceived by the viewer or listener. Put simply the change in brightness or sound intensity represented by a single step should not be discernible. In practice around 250 levels are used for video, with several thousand for audio.

There are many issues which govern the number of quantizing levels required, not merely the fact that the eye, or ear, may not be able to distinguish the change between one step and the next. If the steps are too coarse, i.e. too few levels, visible or audible defects will be noticed. In video the defect shows up as 'contouring' of the picture, whereas with audio signals the effect takes the form of quantizing noise, a granular type of noise which is only noticeable in the presence of a signal.

Increasing the number of levels increases the resolution and decreases the undesirable effects of contouring and quantizing noise. Particularly for audio a greater number of bits also increases the maximum dynamic range. In fact increasing the number of levels is not the only way in which quantizing effects can be reduced, other techniques can be used, of which the two most popular are nonlinear encoding and the use of dither.

When non-linear encoding is used the total number of levels is unchanged, instead the distribution of levels is altered. More levels are assigned to small amplitude signals with fewer for the large signal peaks. The technique reduces the subjective effect of the noise for the low level signal, however the dynamic range is restricted. Non-linear encoding is illustrated in Figure 11 overleaf.

Figure 10 : The effect of Quantising

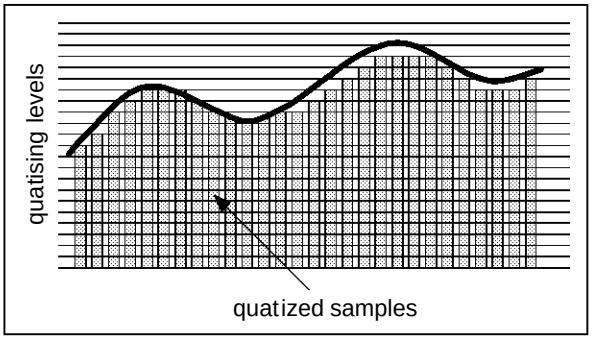
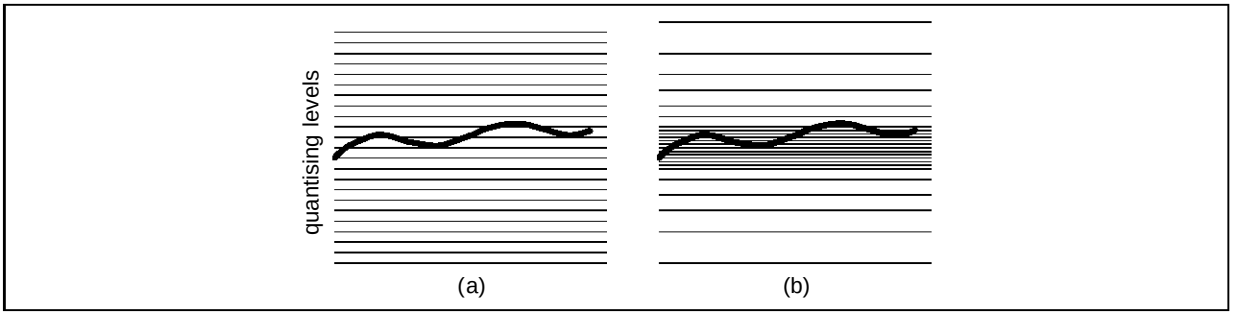
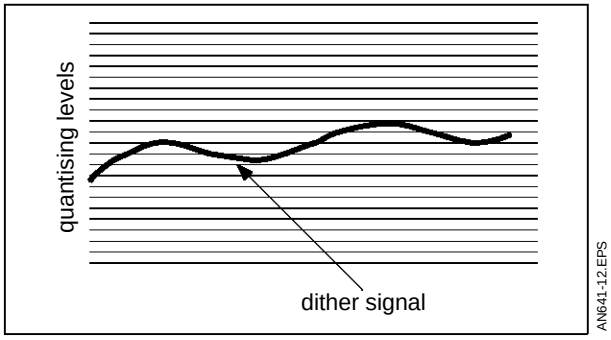


Figure 11 : Comparison of : (a) Linear and (b) Non-Linear Quantising



A different approach is through the use of a 'dither' component. This is a low-level, typically white noise of 1 quantizing level peak-to-peak amplitude, which is added to the analogue signal prior to sampling. The dither component ensures that even the smallest signal must cross at least one sample level - adjacent sample periods will typically correspond to adjacent sample levels. The filtering action of the decoding process will smooth out the effect of the dither and provide a more accurate picture of the original signal. Figure 12 shows the concept of dither.

Figure 12 : Using a Dither Component



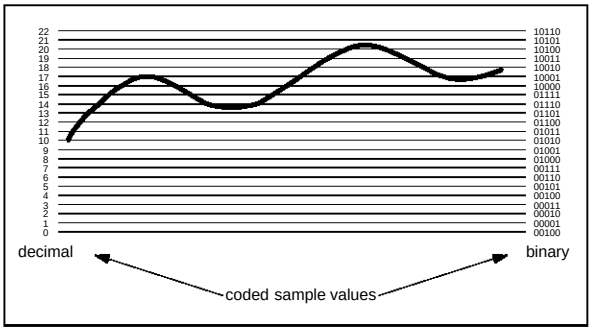
One interesting point to note is that much of the contouring that would normally be seen on a composite digital signal, which has only some 157 quantizing levels between black and white, is masked by the effective dithering action of the chrominance component.

1.3 - Coding

Coding is the final stage in the digitizing process.

The result of sampling and quantizing is still a series of voltage levels, albeit constrained to discrete amplitude levels and time intervals. Coding represents each voltage level as a number, generally expressed in binary form, as shown in Figure 13. Once binary representation is considered the link between coding and quantizing is obvious. The 8-bit coding typically used for video describes 256 brightness levels. Audio uses as many as 16 to 20 bits, the latter corresponding to over 1 million discrete levels.

Figure 13 : Sample Coding



2 - PARALLEL SIGNALS

We now have a parallel digital signal, that is one in which all the bits describing the value of a sample are present simultaneously and is the form in which many digital equipments process the signal. When it is desired to connect one unit to another parallel interfacing may be less than satisfactory, particularly in terms of the inconvenience of having to separately interconnect each bit.

Another problem when connecting one digital unit to another is the need to provide clock information. The signal is in the form of a sequence of samples, each represented by, say, an 8-bit number. At the destination the decoder must be able to identify the separate sample periods in order to recreate the analogue samples from their binary representations.

Parallel interfacing generally includes an additional connection in the form of a clock signal. The clock signal is derived from the master oscillator used to control the sampling and coding process.

Parallel interfacing therefore involves multiple interconnections. For relatively short distances this might be the most practical arrangement, however for longer distances and particularly if signal routing is contemplated, serial transmission is really the only practical solution. Even if all other points are

set aside the need for relatively complex cabling and connectors makes parallel interconnection rather impracticable.

3 - SERIAL TRANSMISSION

At first glance serial transmission is nothing more than sending the signal 'a bit at a time'. Only one link is required, however the data rate is increased by a factor at least equal to the number of bits. For example an 8-bit system is sampled at 1MHz, i.e. one sample every 1µs. When transmitted serially all 8 bits have to be accommodated in the sample period. thus the data rate increases to 8 Mbits/s. Not only has the bit rate increased but there is the problem of synchronization. The decoder must be able to identify not just each bit as it arrives but also which bit is which. The correct sequence of 8 bits must be reassembled, not some from one sample period and the rest from the next.

This requirement can generally be achieved through the inclusion of particular bit patterns which are not allowed during normal data. Logic associated with the decoder 'watches out' for the synchronizing bit pattern - often called the sync word - and is thus able to identify correctly the transmitted sequence.

Successful identification of the sync word enables the decoder to identify only the start of a particular data sequence. Correct decoding of the data which follows depends on the ability of the decoder to keep track of the bits as they are transmitted. Once again clock information is necessary, however as one of the reasons for adopting a serial transmission standard was to minimise the number of connections a separate clock connection should be avoided.

If clock information cannot be provided separately it must feature as an integral part of the signal, that is it should be 'self clocking'. A self clocking signal is one which guarantees to provide a number of transitions in a given period, a voltage controlled oscillator - VCO - in the decoder can lock to the transition information and thus regenerate the missing clock.

The production of a self clocking data signal is one of the functions associated with the concept of channel coding, and is discussed in the next section.

4 - CHANNEL CODING

Channel coding basically describes the way in which the 1's and 0's of the data stream are represented on the transmission path. Many different channel coding standards exist, all are geared to optimize some aspect of the serial signal : dc content, spectral distribution and clock recovery are generally the key factors.

4.1 - NRZ

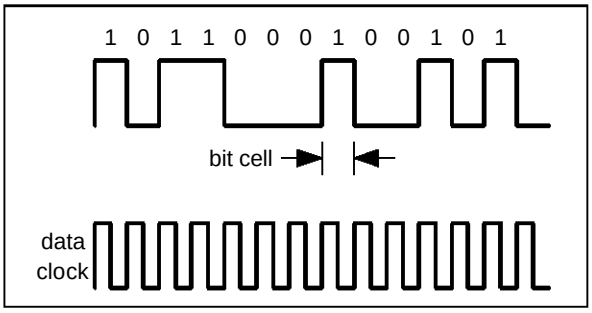
The simplest channel code is the one known as NRZ, or Non Return to Zero. Simple, combinational logic, signals are a good example of NRZ, where a logic 1 is coded as one dc level and logic 0 as another. Figure 14 shows an example of an NRZ signal. Figure 14 also identifies the periods associated with each bit, known as bit cells their duration corresponds to the period of the transmission clock.

NRZ is a simple code and is used within equipment. For serial transmission it has a number of drawbacks however, these are :

(a) it is not self clocking, a long string of all l's or all 0's has no transitions
(b) its dc content varies with the nature of the data stream the low frequency content tends to dominate.

For these reasons NRZ is rarely used for serial transmission, except for relatively low speed operations such as those associated with modem transfers.

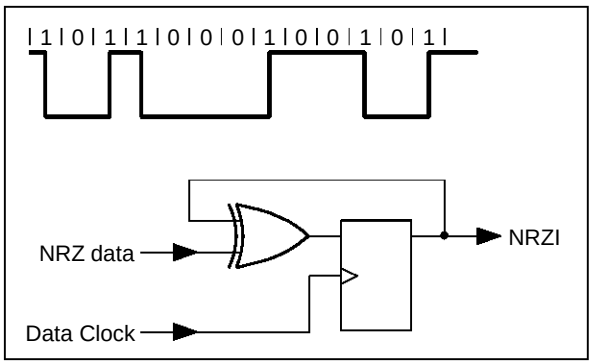
Figure 14 : NRZ Code



4.2 - NRZI

NRZI is a derivative of NRZ, Non Return to Zero Inverse. NRZI codes logic 0's as a dc level, logic 1's however are coded as a transition. Figure 15 shows the data signal of Figure 13 encoded as NRZI, together with a typical method for producing it. This data signal is now self clocking, at least so long as a practical limit is placed on the maximum number of zeros. This will generally be achieved by reserving the parallel all-zero word for sync purposes only. NRZI still has a relatively dominant lf content and is not dc free.

Figure 15 : NRZI code



4.3 - Bi-phase Mark

Bi-phase Mark, or Manchester Code, is a code well known to many broadcast equipment users as it is the form of channel coding associated with time-code. With this code all bits are identified by a transition at the cell boundary, however logic 1's have an additional transition at the mid cell point. Figure 16 shows this.

Because there is a transition for every bit the clock information can be recovered directly from the bitstream without the need for a VCO. This means that the signal can be decoded over a wide range of transmission rates, a situation similar to that of timecode recovery when machines are shuttling at high speed. Also because it is fundamentally the time between transitions which indicates a one or a zero the signal can be recovered irrespective of waveform polarity.

4.4 - Miller Coding

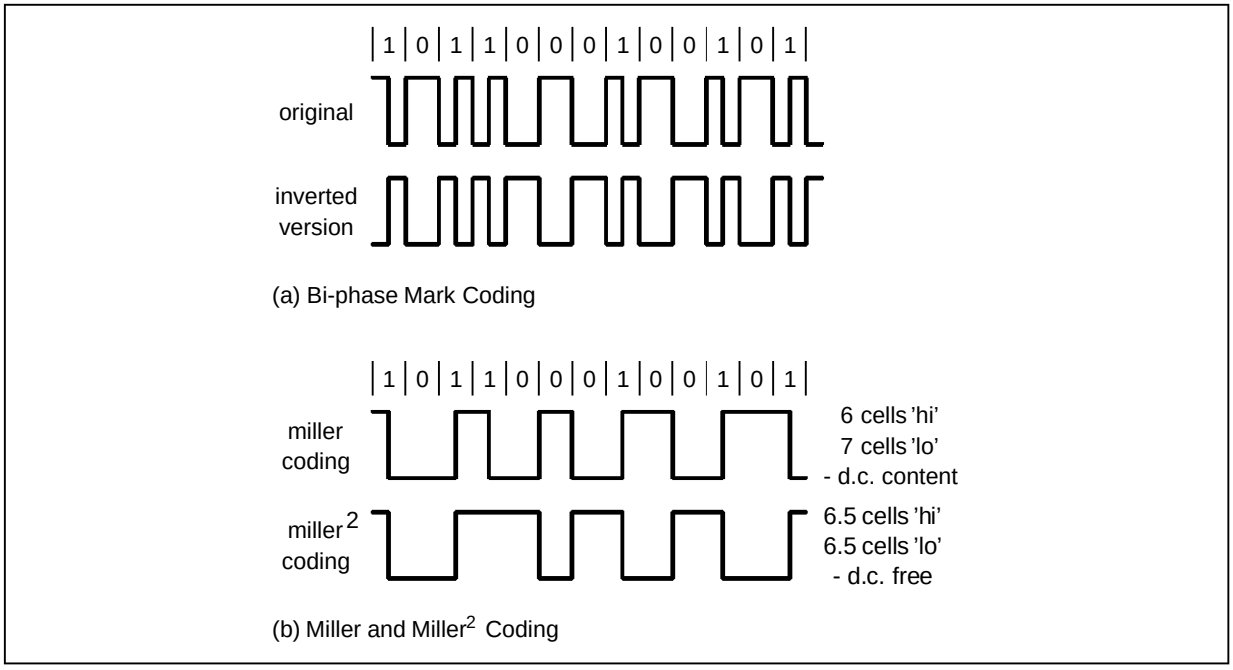
Miller coding, or Modified Frequency Modulation (MFM), like bi-phase mark, provides a transition for every bit. In this code a one is encoded as a transition occurring at the centre of the bit cell, whilst consecutive zeros have a transition at the cell boundary between them (this means that a pattern such as 10101 has no transitions at the cell boundaries). Figure 16 shows an example of Miller coding, together with a derivative, Miller Squared coding.

Miller coding is self clocking and has a relatively low l.f. content. It is not dc free however, and this can cause problems particularly with recording, and long cable runs. Miller Squared coding (so called because it was the result of a modification of Miller coding by a second, quite separate Miller!) has one additional rule. This states that the final transition of an even number of ones occurring between two zeros is omitted i.e.

01110 occupies five cells and has three transitions
011110 occupies six cells but also has three transitions.

As shown in Figure 16 this makes the code dc free. The D-2 composite digital video-tape recorder uses Miller Squared coding as the form in which the signal is recorded on the tape. Amongs other benefits the fact that no erase head is required permits such techniques as Read Before Write, where a signal can be taken off tape, combined with external information, and re-recorded over the original.

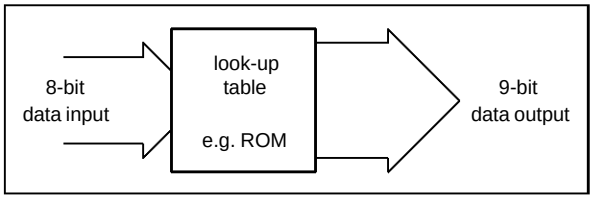
Figure 16



4.5 - Bit Mapping

This represents a very different approach in which an input parallel word is encoded as one having more bits. A typical code for video might be an 8/9 code where each 8-bit video word is represented by one of 9 bits, a word size which has twice the number of states as one of 9 bits. The coding process therefore maps each of the combinations of the 8-bit word to one of 9 bits. Through an appropriate choice of mapping - which 9 bit codes are used and which ignored for example - the channel code desired can be achieved. Coding is typically achieved through the use of look-up tables, such as that of Figure 17.

Figure 17 : Example of 8/9 Code Implementation



Whilst a considerable flexibility exists the data rate is increased over the minimum necessary and often results in a frequency spectrum with pronounced peaks. The l.f. content is generally reduced however.

Whilst a data serialiser has been implemented using an 8/9 coding technique it has not proved possible to develop it in single chip form. Also the data rate is already increased over that of the basic signal, which makes it difficult to contemplate either an increase in data resolution, for improved performance, or the inclusion of non-video information such as audio.

4.6 - Scrambled Codes

An alternative to the codes described above is the concept of the scrambled, or randomized, code. When scrambled the sequence of the data bits is changed, according to either a pre-determined sequence or one having a pseudo-random nature. Scrambling is of little value with streams of already random data, however it can be used to advantage with data that exhibits a high degree of correlation, such as that associated with video waveforms. In such cases scrambling can produce a better spectral distribution than many of the other codes. As the scrambled code aims to provide an equal number of 1's and 0's clock recovery is relatively straightforward.

Scrambling can be on a word basis, where the parallel data is scrambled before serializing, or on a bit-by-bit basis in its serial form. The CCIR-601 recorder uses code scrambling, according to a set of predetermined look-up tables, whilst the serial digital interface described next uses a pseudo-random sequence.

Bit scrambling systems are based on the production of a Pseudo-random Binary Sequence (PRBS), which in turn is combined with the transmitted data. Such a sequence can be generated by means of a shift register with associated feedback, as shown in Figure 18 generally such diagrams show the feedback signals as being combined by an adder. In fact modulo-2 addition is used, where :

0 + 0 = 0,

1 + 0 = 1, and

1 + 1 = 0.

This will be instantly recognized as an exclusive-or function.

Such a generator produces what is known as a maximum length shift register sequence, or more simply an M-sequence. A shift register having n stages has 2^n states, however as the all zero condition effectively 'locks-up' the sequence the number of valid states is 2^{n-1}. The sequence generator of Figure 18 therefore has 15 different states, as shown in Figure 19a.

Notice how the sequence contains eight l's and seven 0's, in other words there is a more or less equal probability of the generator producing a 0 or a 1. As the length of the register increases so does the probability factor tend to one half. Also of course the sequence becomes longer, an 8 stage generator would have 255 states, 9 stages 511 and so on.

As was explained earlier the PRBS is combined with the transmitted data in order to randomize it. Figure 19b shows such an arrangement.

Because the same random sequence is added to the signal for transmission, and then subtracted at the decoder, the recovered data is identical with the original. In practice of course the use of a separate channel is undesirable. This has led to the development of the so-called self synchronizing scrambler, as shown in Figure 20 where a nine stage generator is used. Notice in this example how a feedback connection is used for the scrambling generator, whereas a feed forward implementation is used by the de-scrambler. In both cases the same stage outputs are used.

Before leaving the subject of data scrambling it is worth mentioning the way in which the M-sequence generator is classified. Length is an obvious factor but there is also a 'shorthand' method of describing the feedback connections. This is by means of its so called characteristic Polynomial. For the nine stage registers illustrated in Figure 20 this is:

$$f(x)=x^9+x^4+1$$

Figure 18 : Pseudo-Random Binary Sequence Generator

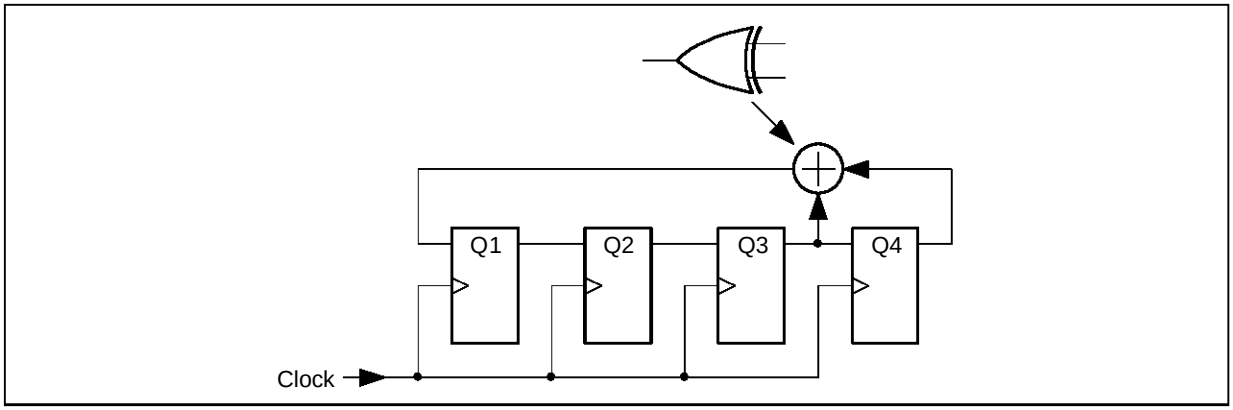


Figure 19a : States of the 4 stage M-sequence Generator

Q1	1	0	0	0	1	0	0	1	1	0	1	0	1	1	1
Q2	1	1	0	0	0	1	0	0	1	1	0	1	0	1	1
Q3	1	1	1	0	0	0	1	0	0	1	1	0	1	0	1
Q4	1	1	1	1	0	0	0	1	0	0	1	1	0	1	0

After this the pattern repeats

AN641-19.EPS

Figure 19b : Using the M-sequence Generator to Randomize a Data Signal

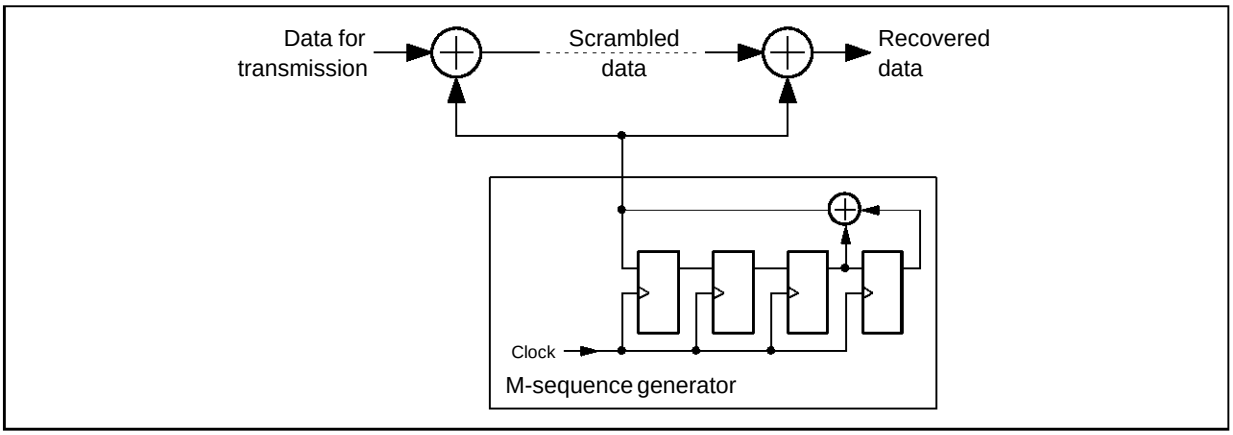
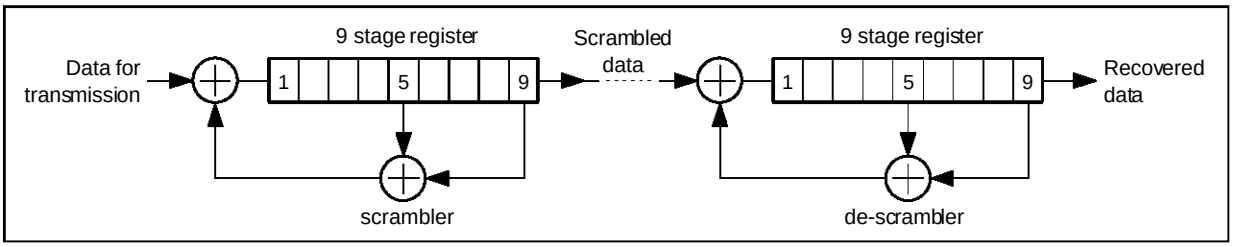


Figure 20 : Self-synchronising Scrambler



APPENDIX 2 : SERIAL INTERFACE - TECHNICAL DETAILS

This Appendix looks at the details of the hardware associated with the Serial Digital Interface, in particular the VLSI chips developed to support it. Whilst correct at the time of writing it is possible that certain details might be changed as a result of development.

1 - THE SERIAL DATA FORMAT

1.1 - What is Carried

As mentioned in the introduction one of the main sources-of impetus for the digital interface is the development of digital VTRs and the digital video switcher and its audio equivalent. It seems only natural therefore that the interfacing should best deal with interconnecting these equipments.
The serial digital interface offers one video channel, either to D-I or D-2 standard, both 525 and 625, together with a maximum of four channels of digital audio data plus Vertical Interval Time Code (VITC).

1.2 - The Basic Interface

Two distinct areas of signal processing can be identified when the operation of the interface is considered, data combining and serializing. Data combining takes place at source and relates to the bringing together of video, timecode, and audio digital data into one common signal. The signals are combined as parallel data, the combination which results is then serialized for transmission. At the decoder the processes are reversed, de-serialisation is followed by data separation.

1.3 - Data Transmission

As we have already seen the serial interface is designed to handle signals in either D-1 component or D-2 composite forms, to either 525/60 or 625/50 standards. Although component operation is the most demanding case from the transmission point of view, the composite situation is complicated by the relatively short periods available for the auxiliary data. This is because the D-2 video interface, unlike its D-1 counterpart, includes a digital specification of video synchronizing information - the sync edges and the colour burst. Throughout this Appendix the timing diagrams refer to composite operation, as typified by the PAL D-2 blanking interval shown in Figure 21.
Two other data signals, in addition to the video, are carried by the serial interface, synchronization and auxiliary data, of which audio is a good example.

1.3.1 - Serial Data Synchronization

Synchronization of the serial data is by means of a sequence of TRS (Timing Reference Signal) words. Three such words are used, corresponding to the 10 bit values $3FF_{HEX}$, 000_{HEX}, 000_{HEX}, i.e. the same data pattern as the D-I EAV and SAV signals. Note that as the value 000_{HEX} is reserved for synchronization it ensures that a long sequence of 0's cannot occur during actual data.
In fact, as the interface is designed to cope with both 8 and 10 bit data four values are reserved. corresponding to 000_{HEX}, 001_{HEX}, 002_{HEX} and 003_{HEX}. This allows an 8 bit word (reserved value 00_{HEX}) to be transmitted as a 10 bit word with two 'dummy' LSB's.
The TRS data is transmitted on three samples following the sync leading edge. i.e. samples 790 792 in the NTSC example of Figure 22 overleaf.
Immediately following the TRS is a Line ID. This ID indicates the field number (within a four field sequence for NTSC, eight field for PAL), as well as an indication of the position of the line within the field. Up to line 30 the line number is given explicitly, above this no indication other than the fact that the line number is greater than 30 is provided. Figure 23 shows this. The sync level samples which follow the Line ID, up to the trailing edge of sync, can be used for auxiliary data. Figure 24 shows which samples may be used, again the figures are for NTSC.

Figure 21 : D-2 Blanking Interval Showing Data Sample Periods (PAL signal)

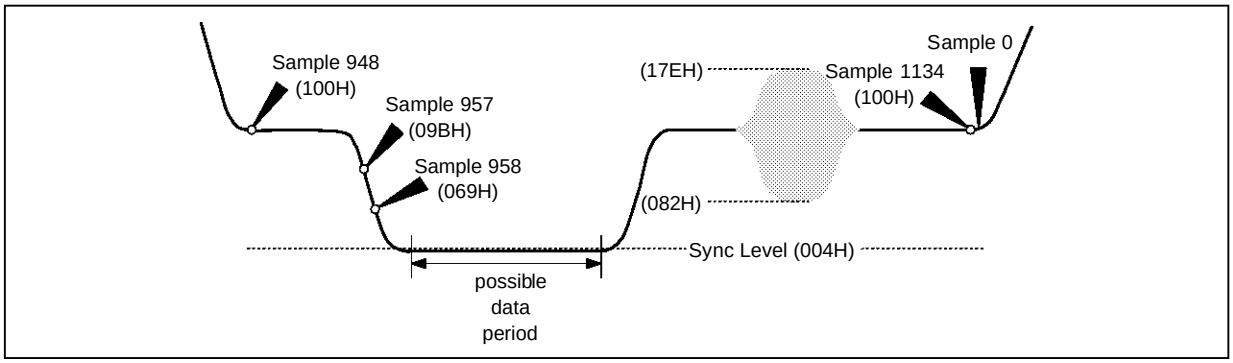


Figure 22 : Position of TRS Data

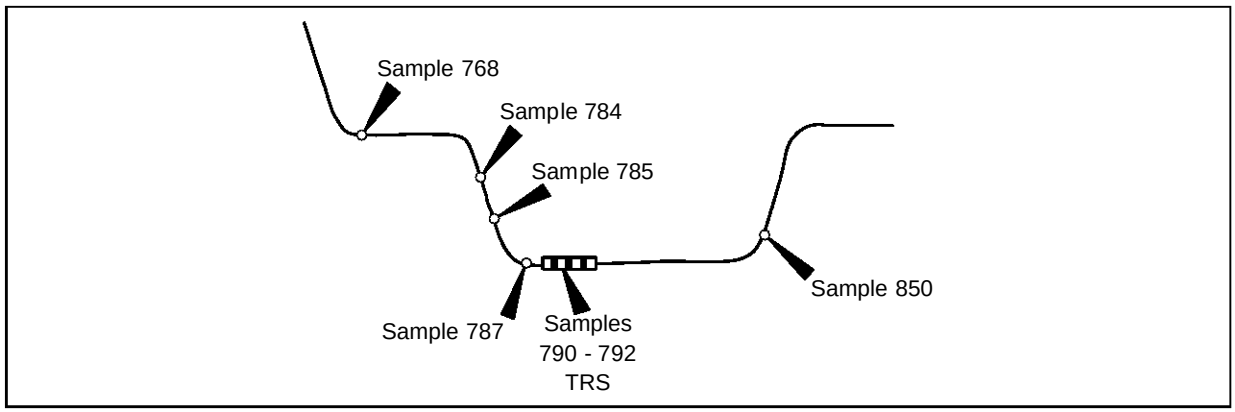
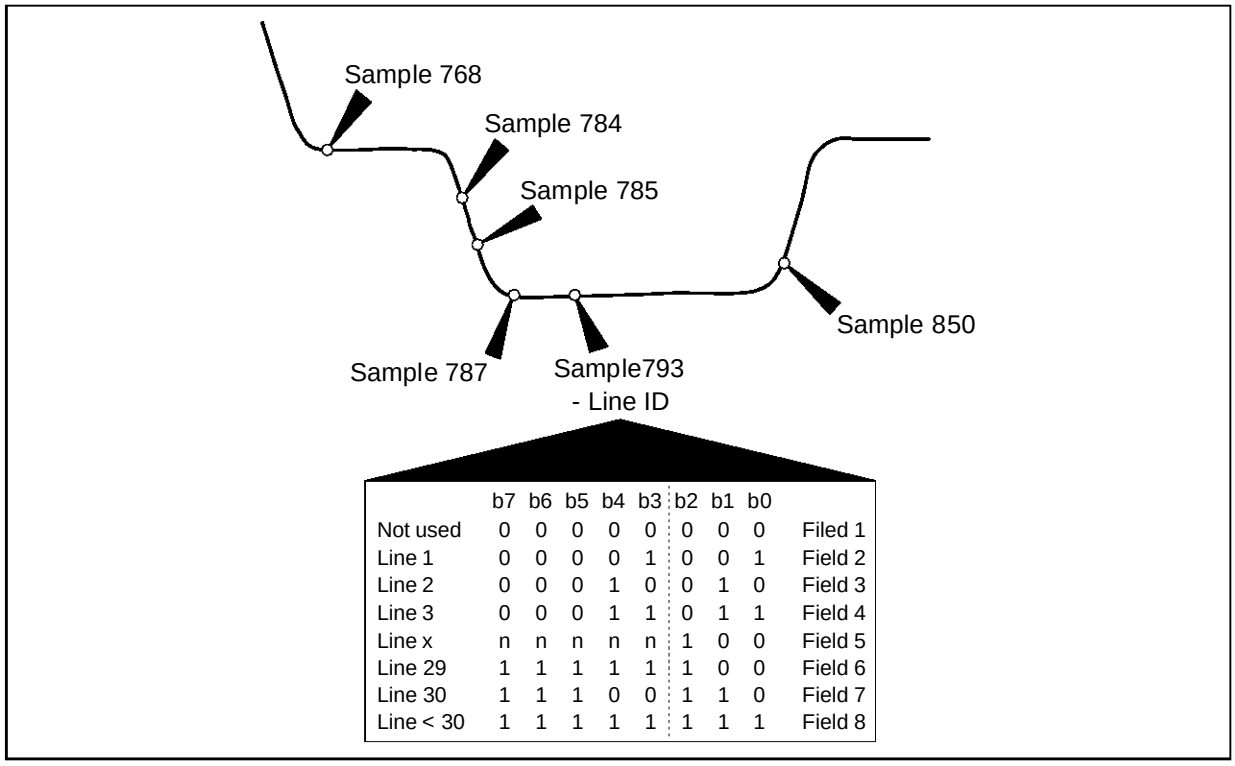
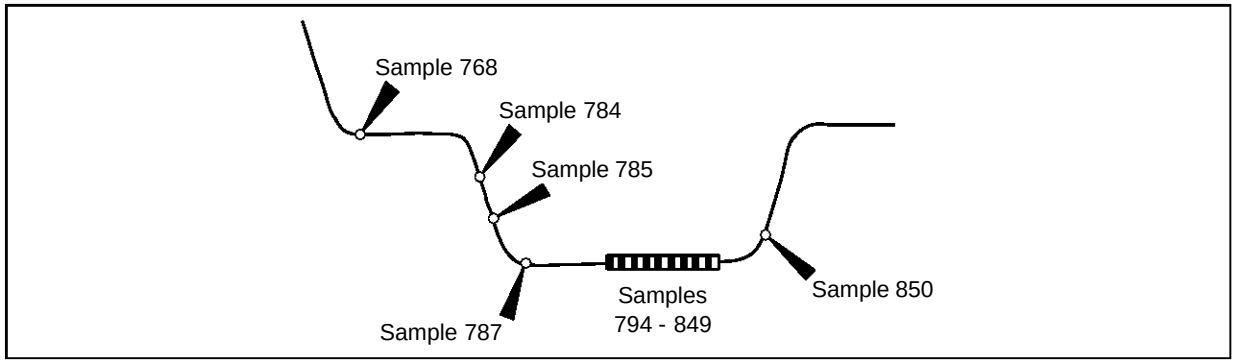


Figure 23 : Line ID



	b7	b6	b5	b4	b3	b2	b1	b0	
Not used	0	0	0	0	0	0	0	0	Filed 1
Line 1	0	0	0	0	1	0	0	1	Field 2
Line 2	0	0	0	1	0	0	1	0	Field 3
Line 3	0	0	0	1	1	0	1	1	Field 4
Line x	n	n	n	n	n	1	0	0	Field 5
Line 29	1	1	1	1	1	1	0	0	Field 6
Line 30	1	1	1	0	0	1	1	0	Field 7
Line < 30	1	1	1	1	1	1	1	1	Field 8

Figure 24 : Data Sample periods



1.3.2 - Auxiliary Data

Auxiliary Data is transmitted with the block format illustrated in Figure 25. The maximum length of a block is 260 words (10 bits), which corresponds to 255 words of actual data.

The breakdown of the block is as follows :

- Auxiliary Data Flag (l word)
  This flag must be present if the auxiliary data is to be recognized. The flag has the value $3FC_{HEX}$.
- Data ID
  This indicates the type of data, and could have one of 256 states, determined by the state of the 8 LSB's. The remaining bits are used for parity checking - b_8 is an even parity bit for b_7 - b_o, whilst by is the inverse of b_8.
  Although capable of distinguishing 256 different types of data only two are currently identified : AES/EBU format digital audio, and unformatted data. A data ID of FF_{HEX} indicates audio, 00_{HEX} unformatted.
- Data Block Number
  This is used to distinguish different blocks of data which all share the same Data ID. It increments by one over the range 1 to 255 as consecutive blocks are transmitted. In this way a string of data longer than 255 words can be transmitted and reassembled in the correct sequence. For data that is not continuous, or identified in some other way, the block number is set to zero. As with the Data ID the value is conveyed by the 8 LSB's, with the two MSB's providing a parity check.
- Data Count
  The Data Count word indicates the number of data words included in this block. It can have any value from 1 to 255, and is encoded in precisely the same way as the other control words.
- Checksum
  This is a single 9 bit word and is used to check the validity of the Data ID and user data. It is calculated as the sum, ignoring carry, of the 9 LSB's of the user data and Data ID words.

1.3.3 - AES/EBU Formatted Audio Data

When the Data ID is set to FF the auxiliary data is interpreted as audio signals corresponding to the AES/EBU format. Up to four channels of audio can be included, with a maximum of 27 bits per data sample, spread over three words. 20 of the 27 bits are used for the audio sample itself, the remainder are used for channel identification (2 bits) and the AES/EBU flags. Table 1 overleaf shows the distribution of data between the three words.

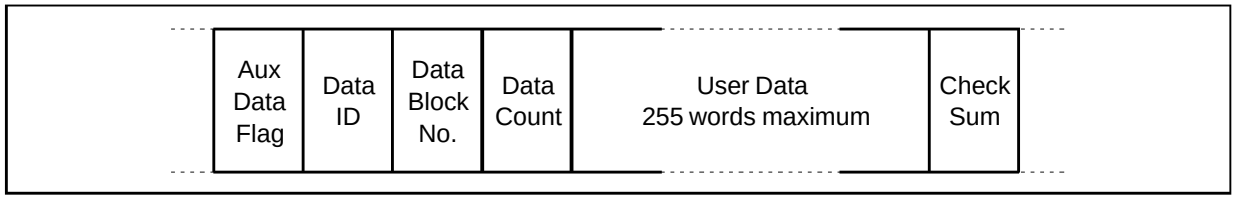
Table 1 : Distribution of AES/EBU Format Audio Data

bit	1st word	2nd word	3rd word
b9	b8	b8	b8
b8	d5	d14	P
b7	d4	d13	C
b6	d3	d12	U
b5	d2	d11	V
b4	d1	d10	d19
b3	d0	d9	d18
b2	CH msb	d8	d17
b1	ch 1sb	d7	d16
b0	Z	d6	d15

1.3.4 - Unformatted Data

Unformatted data can be transmitted in one of two forms, either 8-bit data with b_8 providing an even parity check, or-9 bit data with no parity checking. In either case b_9 is not used unless as the inverse of b_8.

Figure 25 : Auxiliary Data Format



2 - HARDWARE DETAILS

Sony has developed a number of VLSI chips with which the serial interface is implemented. Some, the co-processors, are concerned with combining parallel data such as audio and video and its subsequent separation. Other chips implement the serial link itself. A diagram showing the key features of the transmission system is shown in Figure 26.

2.1 - Transmitter Co-Processor

The transmitter (and receiver) co-processor is only available for composite signal processing at present.

Essentially the co-processor performs two tasks : it combines video and auxiliary data signals prior to serialisation, and it generates the TRS and line ID data. A block diagram of the co-processor will be found in Figure 27.

2.1.1 - Main Data Path

Composite digital data is first combined with the TRS and Line ID signals (see Section 1.3.1). The co-processor assumes 10-bit data throughout the main path, if the input is only 8-bit then the two input LSB's are merely dummies. The TRS and Line ID signals are produced by the main timing block, which is itself controlled by reference video timing information. The timing block ensures that the TRS and Line ID insertion timing is correct (see Figure 22). A sync phase adjustment allows timing discrepancies of up to 3 clock periods between the video data and sync reference to be accommodated.

Figure 26 : Transmitter Co-processor Bloc Diagram

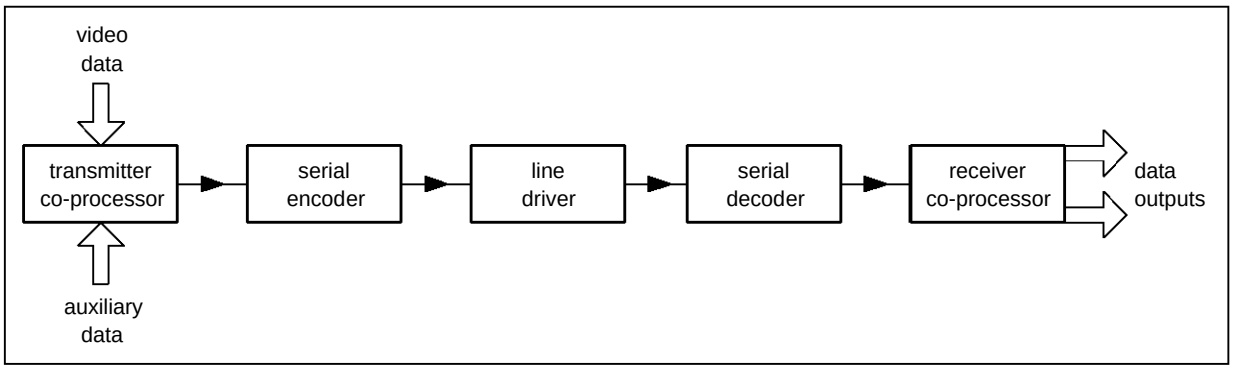
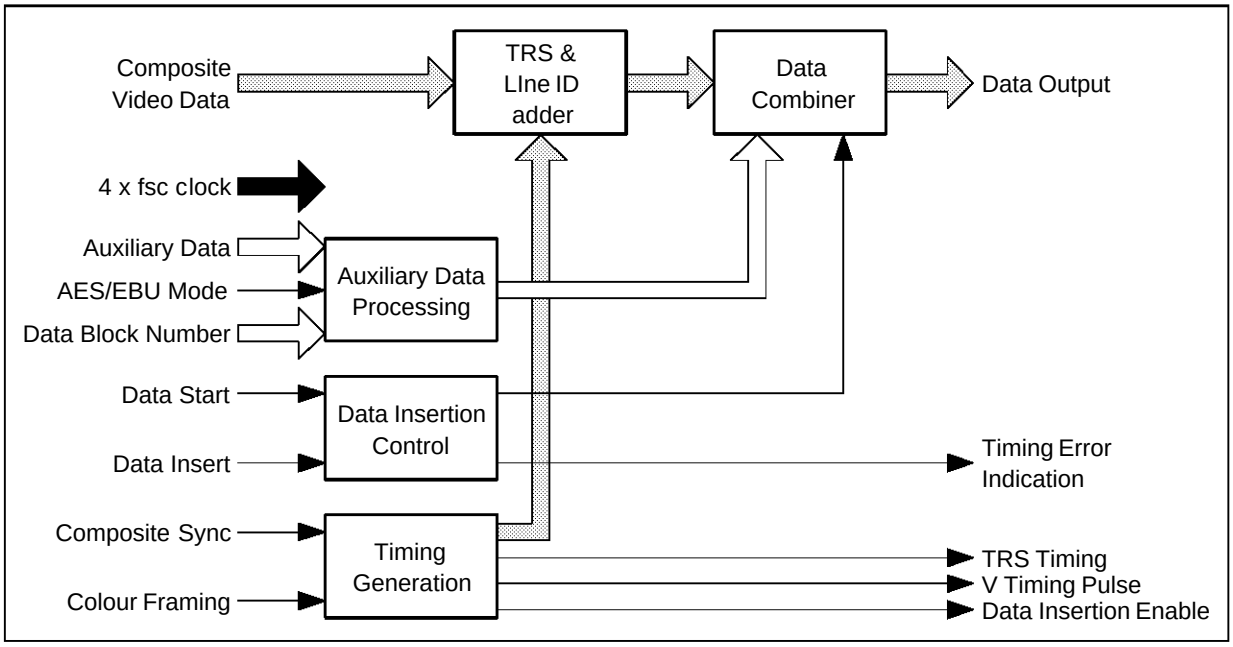


Figure 27 : Transmitter Co-processor Block Diagram



2.1.2 - Auxiliary Data Path

A number of functions are associated with the auxiliary path, principally mode control and insertion timing.

Mode control governs the way in which the auxiliary data is handled and formatted. With the AES/EBU mode input held high the processor assumes that the incoming data is formatted according to the AES/ EBU standard. The processor internally sets the Data ID word to FF_{HEX}, and the Data Block Number to the value present at the Data Block Number input (see 1.3.2 for description of Data ID etc.). The Auxiliary Data Flag is also internally set (to $3FC_{HEX}$) to indicate the presence of auxiliary data.

The information taken from the auxiliary data input is restricted to the Data Count and Audio Data words only. The checksum word is ignored and a new checksum calculated internally according to the rules outlined in Section 1.3.2.

When non AES/EBU formatted data is input (mode control held low) the Data ID and Day Block Number words are also derived from the incoming data. The Auxiliary Data Flag and Checksum continue to be generated internally Figure 29 shows the way in which such data fields are combined.

2.1.3 - Data Insertion Control

Insertion timing control ensures that the incoming auxiliary data is presented by the external device at the appropriate time. Figure 24 showed that 55 words were available during the NTSC horizontal sync period for the transmission of auxiliary data. Because of the different 4 times fsc clock rates the figure for PAL is somewhat higher at 61 words. To enable the external device to match its data to the transmission sequence the co-processor generates a Data Insertion Enable (DIE) pulse which can have three durations depending on the precise point in the field. There is, of course, less time for data during an equalizing pulse and rather more during a broad pulse. Figure 28 shows the timing of the three DIE pulse durations. The apparent one word discrepancy with Figure 24 is due to the fact that the DIE pulse does not include the checksum word.

In order to provide the external device with 'advance warning' of the different insertion periods two vertical rate output pulses are also provided. designated ENMOD0 and ENMODl respectively. Both are normally low, however ENMOD0 goes high during the vertical sync period (3 lines NTSC, 21/2 lines PAL) whilst ENMODI has two high periods, corresponding to the pre- and post- equalizing pulses.

Figure 28 : DIE Pulse Durations

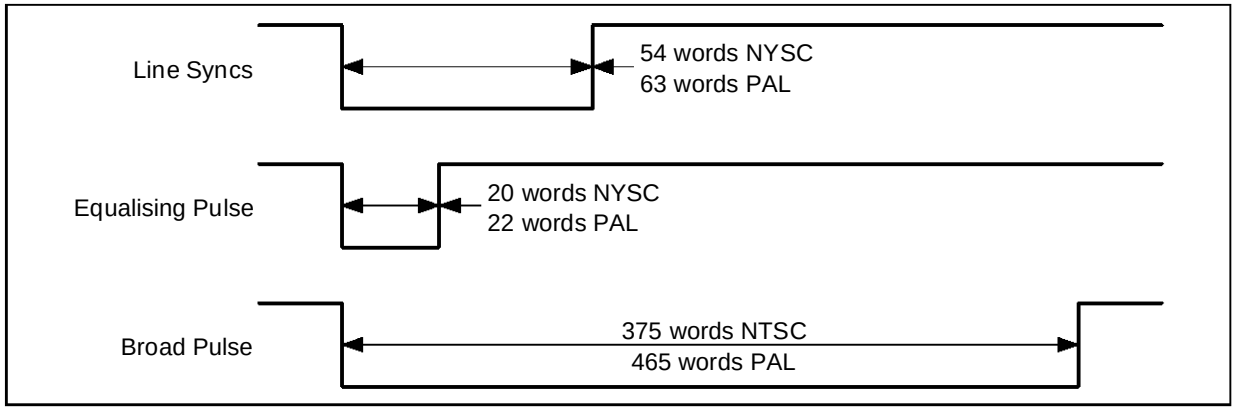
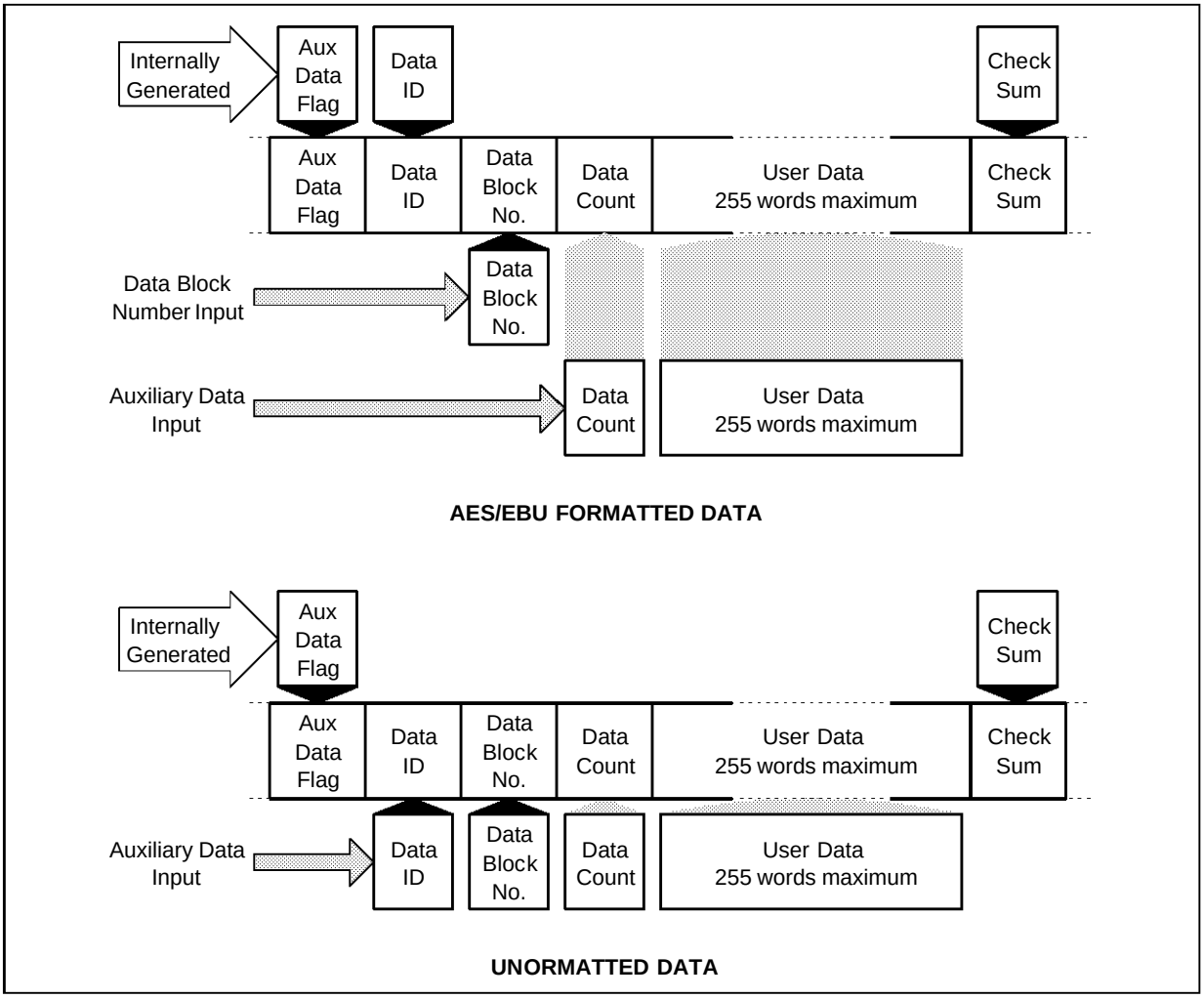


Figure 29 : Auxiliary Data Formatting



Once the external device has prepared its data it signals this by means of two control signals, Data Start Pulse (DSP) and Data Insert Pulse (DIP). These pulses indicate the start of the user data (defined as the timing of the Auxiliary Data Flag), and the period of actual data. which is, of course, different for formatted and unformatted data. Figure 30 overleaf shows the timing relationship of these signals for both cases.

Of the two pulses the timing of DIP is perhaps more important than that of DSP, the processor allows some leeway in the case of the latter. In terms of their relationship with the processor generated DIE signal both must start at least one clock period later. In terms of their relative timing DSP can occur up to 3 clock periods early, or 1 clock period late, with respect to DIP. This basically means that DSP can be permanently set for either formatted or unformatted data, and data of either type transmitted. The DIP timing is then changed to suit the data. An error indication is provided if these conditions are not met. Figure 31 illustrates these points.

2.1.4 - Timing Generator

The timing generator derives control timing information from the composite sync signal, in particular a vertical reference pulse which flags the start of each field. All insertion timings are derived from the sync signal, clocked by 4fsc as appropriate. One of the processor inputs is a PAL/NTSC mode controL which is used to modify the timings of signals such as ENMOD and DIE in order to reflect the different system parameters. Colour framing information (four field NTSC, eight field PAL) is also input. This is used to determine the state of the three LSB's of the Line ID, see Figure 23. A timing signal to indicate the position of the TRS information is also provided.

Figure 30 : Timing of DSP and DIP Signal

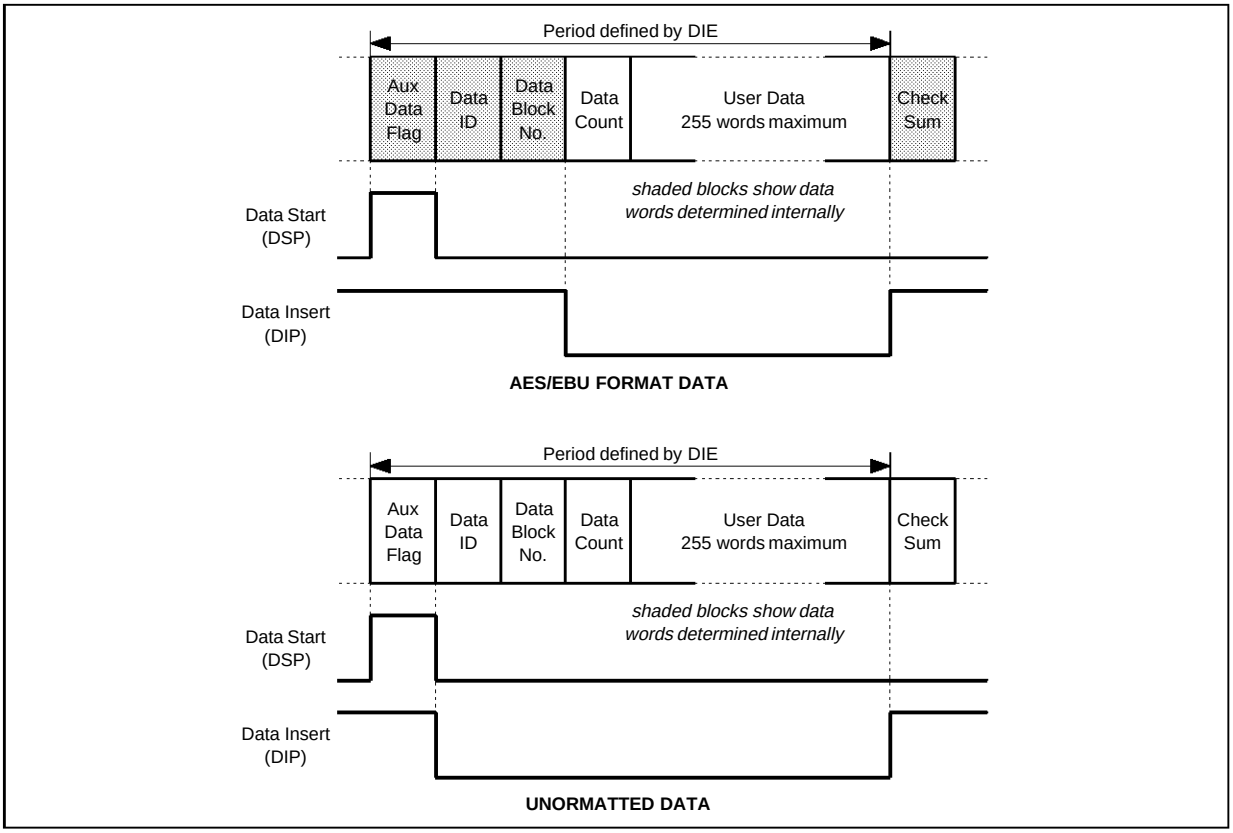
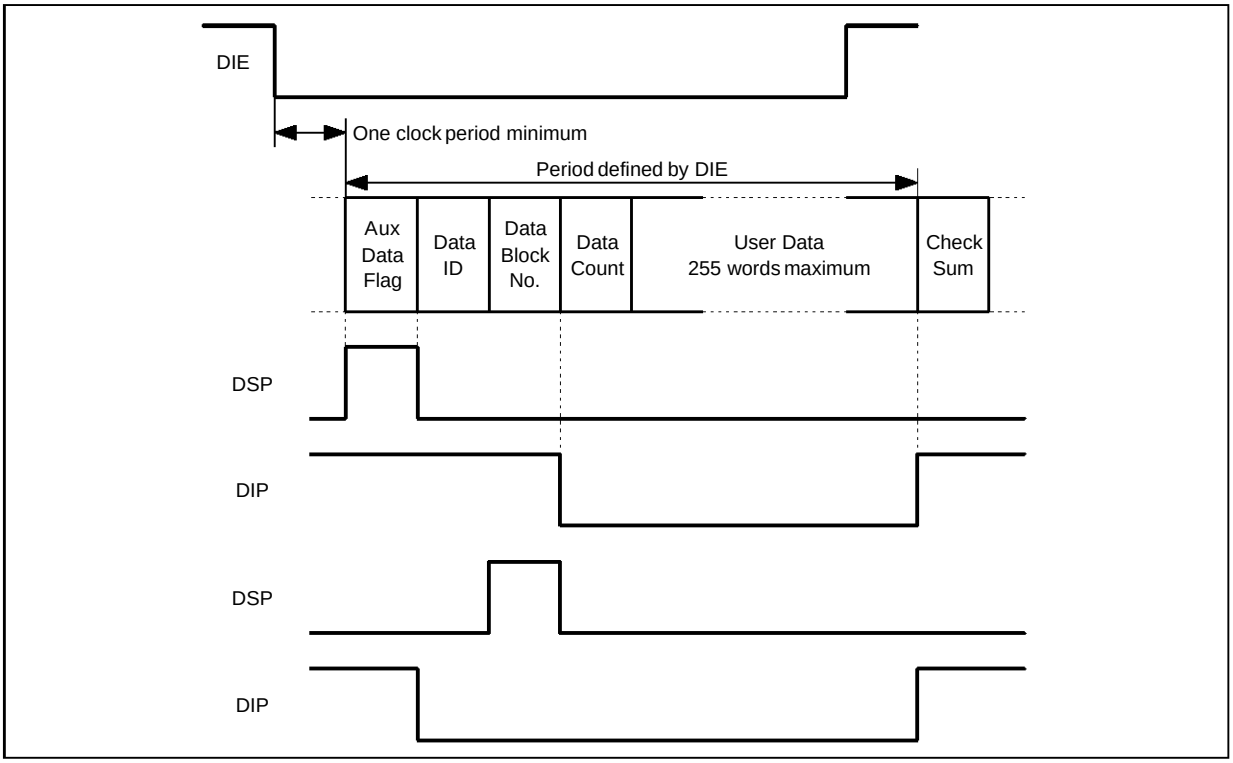


Figure 31 : Timing Requirements for DIP and DSP



2.2 - Serial Encoder

The encoder contains all the circuitry for parallel to serial conversion of the data signal, channel coding and transmission line driving. Whilst the internal transmission line driver is suitable for board to board interfacing the external cable driver will normally be used as well.
Whilst features such as parallel-to-serial conversion itself and clock generation are, perhaps, obvious two features are worthy of explanation.

2.2.1 - Sync Identification

Whilst the responsibility for generating the TRS ($3FF_{HEX}$. 000_{HEX}, 000_{HEX}) rests with the external parallel data processing the encoder includes circuitry to trap the presence of multiple all-zero words. If three consecutive all-zero words are detected at the input the two LSB's are set to one (003_{HEX}).

2.2..2 - Data Scrambling

The channel coding is described as scrambled NRZI (see Sections 4.4.2 and 4.4.6 in Appendix 1). In fact more accurate description would be (Scrambled NRZ), as the input NRZ data is first randomized, and the result converted to NRZI. Figure 32 shows the form of the so-called scrambler. Note that this as no effect on logic 0's, however the possibility of a long run occurring is limited because of the data restrictions associated with the sync word. In fact the maximum number of zeros recurring at the scrambler output has been calculated as 38 after the sync word, but only 20 with normal data.
Although a long run of ones could be produced these are converted to transitions by the NRZ to NRZI conversion, which helps to reduce the If content of the signal.

2.2.3 - Logic level

The parallel inputs are pin selectable for either balanced ECL levels, or single ended TTL drives.

2.3 - Decoder

The key features of the decoder are cable equalization, clock recovery, de-scrambling and serial to parallel conversion, as shown in the overview of Figure 33.

Figure 32 : Form of Data Scrambler

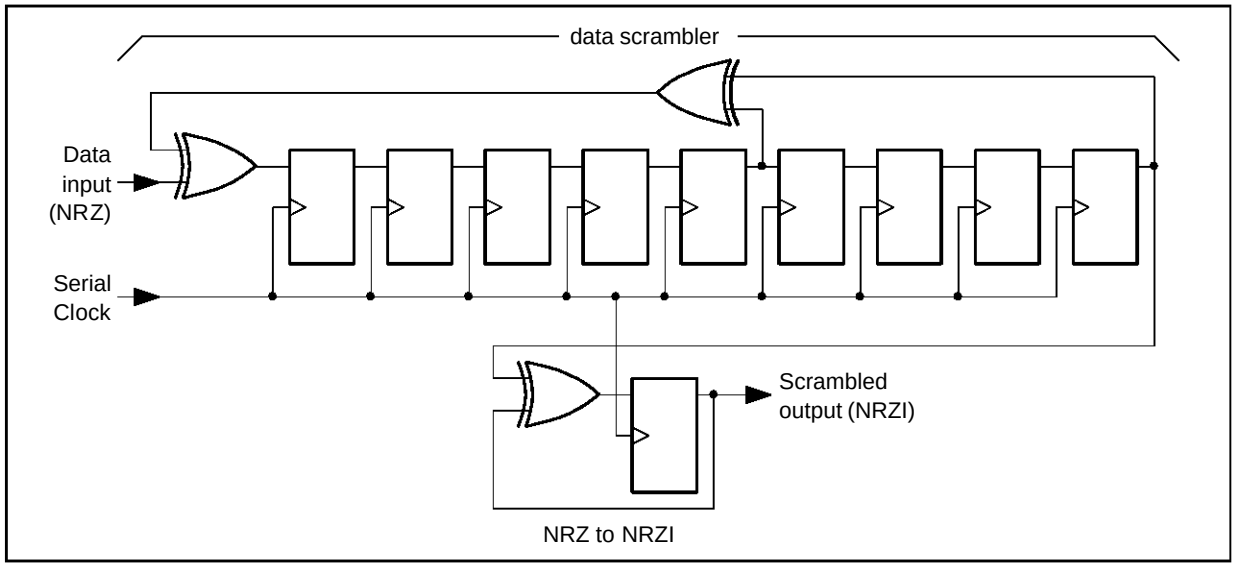
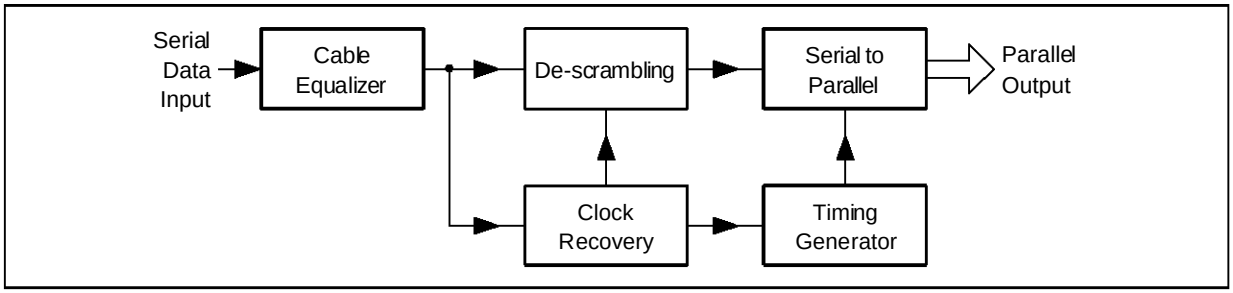


Figure 33 : Serial Decoder Overview



2.3.1 - Cable Equalization

Although line driving is carried out by a separate IC associated with the encoder cable equalization is an integral part of the decoder. In fact the decoder has two selectable data inputs, a 'digital' input intended for use with a balanced ECL transmission line, and an analogue input for single ended co-axial transmission.

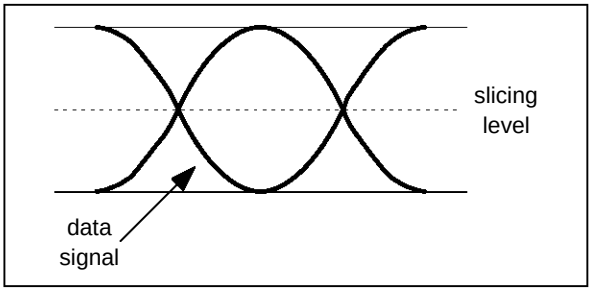
The analogue input includes a cable equalizer capable of correcting both the h.f. and l.f. responses of the circuit. The hf equalizer is designed to correct a loss of some 30dB at 135MHz, and is shown in Figure 35.

Rather than passing the input signal through some form of reactive correction circuit variable gain amplifiers in a side chain are used. With no hf loss, as determined by peak detection, the side chain amplifiers are just cut off. As the h.f. content falls the peak detector increases the gain of the amplifiers in order to boost the hf response. The equalizer is capable of maintaining a constant signal level, within +2dB, for cables of up to 300m.

Note that as the equalizer responds to the peak signal level it is vital that the signal level at the encoder output is set correctly.

The decoder also corrects the lf response, which in this case is taken as frequencies below 8MHz. Fundamentally the lf equalizer aims to keep the data signal at the dc level that results in half-amplitude sampling, see Figure 34.

Figure 34 : Optimum Slicing Level



"Basically the equalizer integrates the output of the data slicer, and uses the resulting dc to offset the slicer input bias. Figure 36 shows this.

2.3.2 - Clock Recovery

Clock recovery relies on the fact that the NRZI data stream contains a large number of transitions. An edge detector circuit identifies the transitions and passes them to a phase detector. This compares the local VCO output with the incoming data transitions and so regenerates the serial clock, see Figure 37.

2.3.3 - De-scrambling

De-scrambling follows the same procedure as scrambling in reverse. First the NRZI signal is converted back to NRZ, and then descrambled. Figure 38 shows this.

Figure 35 : Cable Equalizer

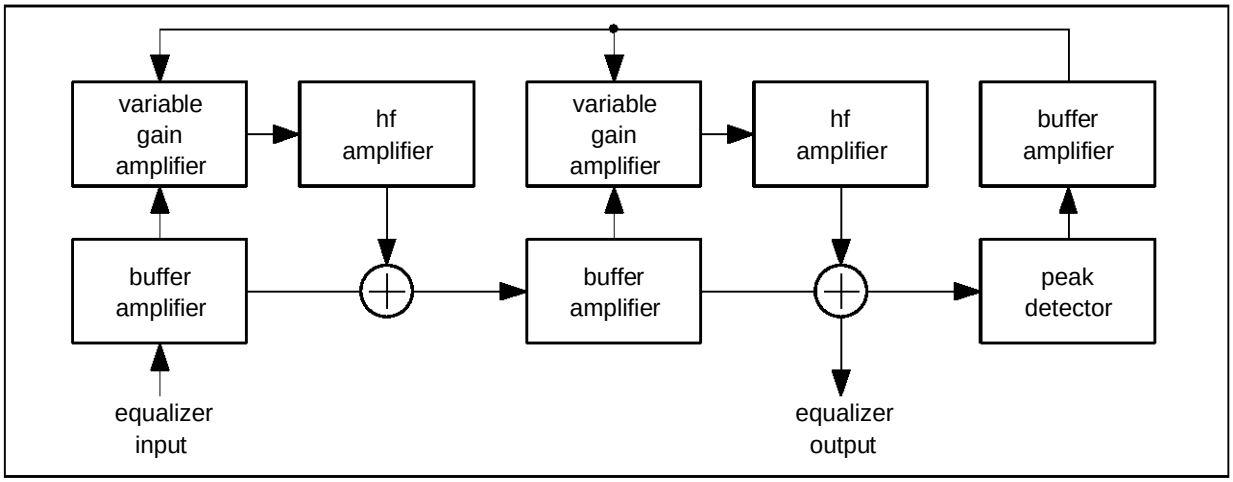


Figure 36 : DC Compensation

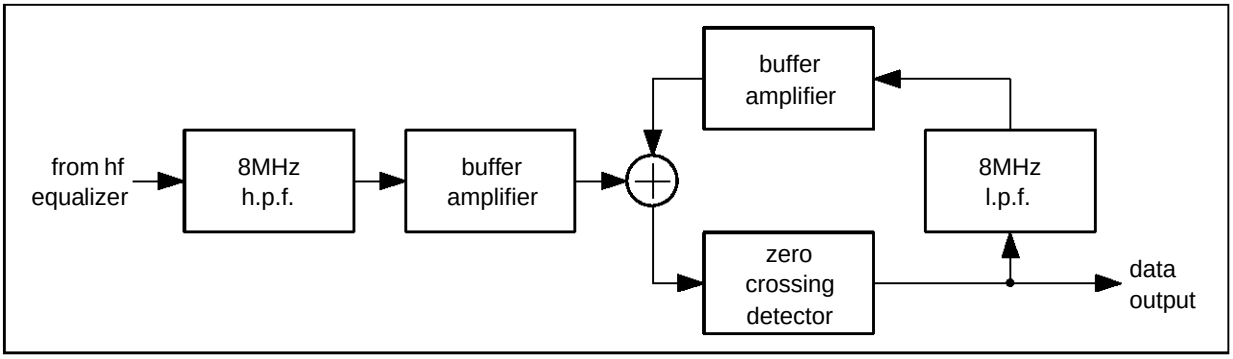


Figure 37 : Serial Clock Regeneration

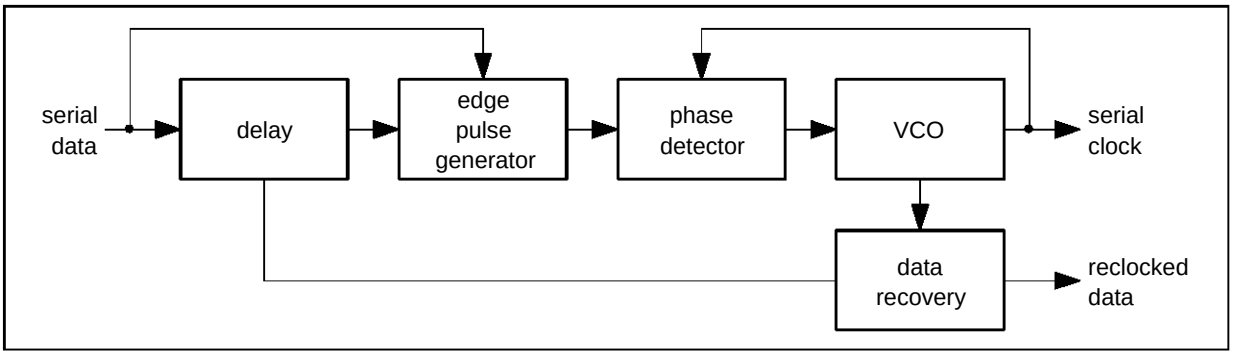
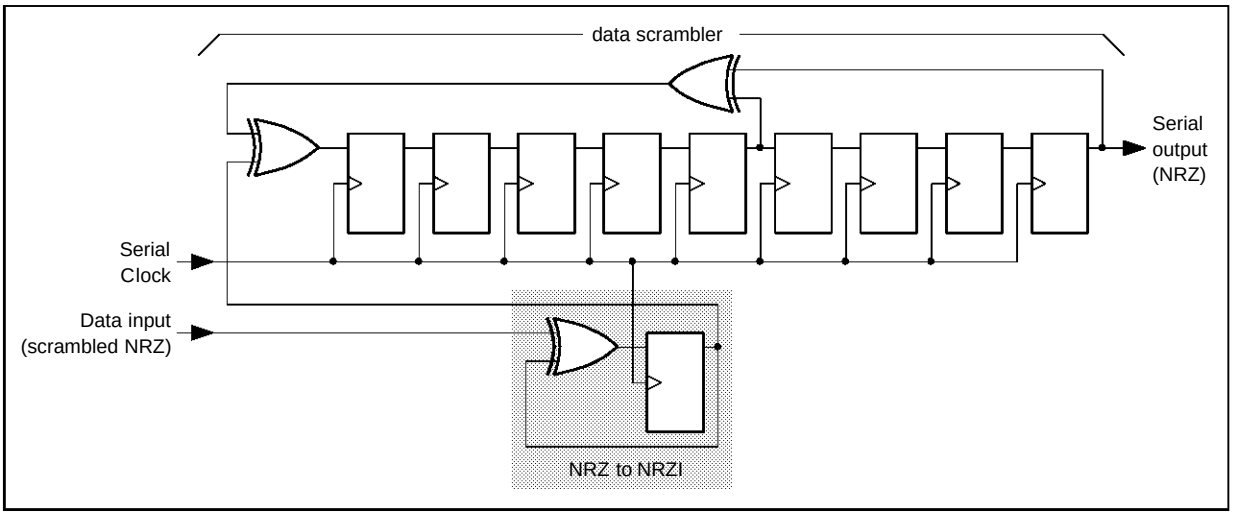


Figure 38 : Data De-Scrambling

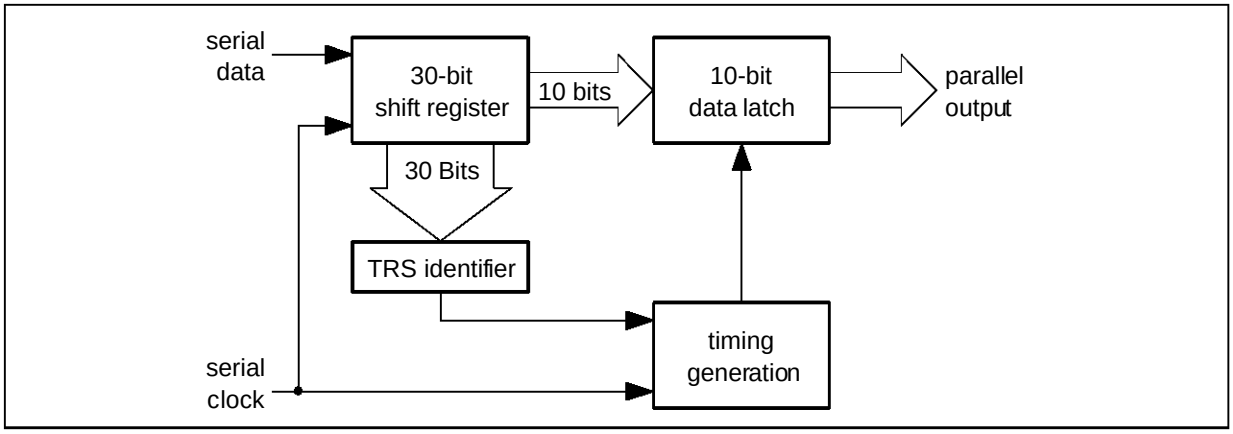


2.3.4 - Serial-to-parallel Conversion

10-bit parallel data words have to be re-assembled from the de-scrambled serial data. This means storing the data until a complete group is available and latching the result. How does the decoder identify the bit groups?

The answer is by looking for the TRS - the timing reference signal. As described in section 1.3.1 . the TRS is a reserved combination of three ten bit data words - $3FF_{HEX}$, 000_{HEX}, 000_{HEX}, the order of transmission being 3FF, followed by 000, 000, LSB's first. The decoder therefore stores a sequence of 30 bits in a shift register. The parallel outputs are continuously examined for the TRS data pattern, and, when identified, a sync signal is generated. The sync signal is used to control the timing generator used to produce the parallel clock, which in turn is used to latch the 10-bit data words to the decoder output. Figure 39 shows these functions.

Figure 39 : Serial-to-parallel Conversion



2.4 - Receiver Co-processor

The receiver co-processor caries out the reverse functions to those of the transmitter, at separates the video, audio and ancillary data for their respective destinations as well as providing control signals and error indication.

A block diagram of the co-processor is shown in Figure 40.

2.4.1 - Sync Data Replacement

This block is responsible for returning the sync intervals of the video data to normal. It therefore replaces all the non-video data words (TRS, Line ID, Aux Data, etc.) with the appropriate blanking or sync interval value. For D-2 PAL this is 004_{HEX}, 010_{HEX} for NTSC. The timing of this process is achieved from the Line ID information, which includes information about the line timing within the field and hence allows a replica of the transmitter DIE signal to be reconstructed. If a Line ID error is detected for three or more lines the ID Error output goes high to flag the condition.

2.4.2 - Auxiliary Data Recovery

The recovery of auxiliary data is initiated by correct identification of the Auxiliary Data Flag in the data stream. An output (User Data Exists) is provided as one of the control signals. The nature of data recovery depends on whether the data is formatted or not. This is achieved through detection of the Data ID and Data Count words. The auxiliary data block is, in fact, output in its entirety (Aux Data Flag to Checksum inclusive), however timing signals are provided to enable external equipment to identify the component parts of the data. Figure 41 shows this information

2.4.3 - Error Information

In addition to the Line ID error output described in Section 2.4.1 a composite error flag is provided. The signal is the result of parity checks performed on individual data words. By combining the error flag with the timing control signals shown in Figure 41 specific errors can be identified - e.g. Data ID error, Checksum, etc. An example error flag output is also shown in Figure 41.

Figure 40 : The receiver Co-Processor

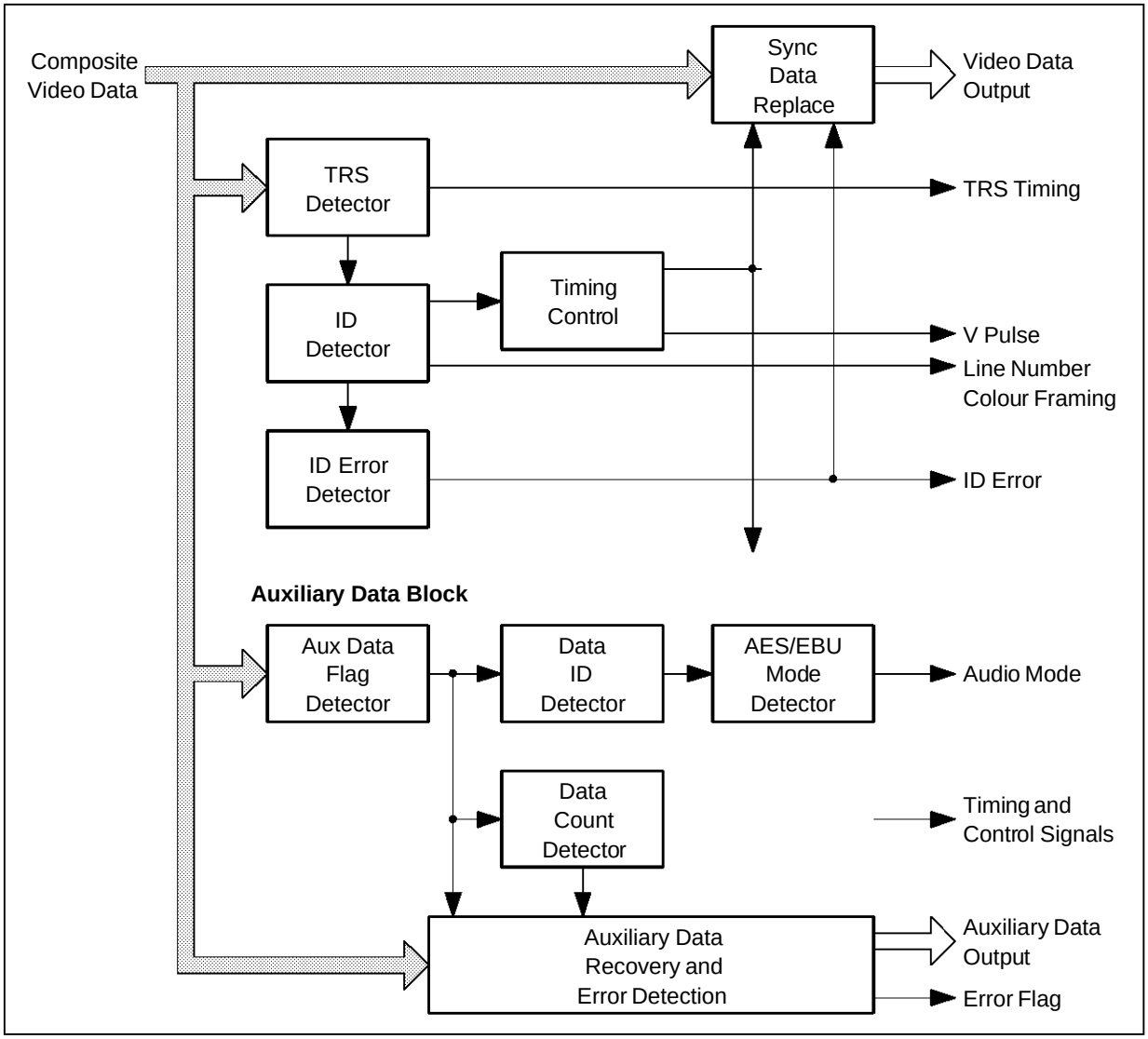


Figure 41 : Timing Control Signals