C6

Communications and Signal Processing

In this Lab, you will use the EMONA Telecoms-Trainer (ETT-101) in order to analyse pulse code modulation and experience real-time transmission of speech signals.

Then, in the second part of the lab you will model deterministic and random signals using Matlab and investigate the statistical characteristics of these signals.

Schedule

Preparation time	:	3 hours
Lab time	:	3 hours

Items provided

Tools	:	None
Components	:	*None*
Equipment	:	Oscilloscope, EMONA Telecoms-trainer 101 (ETT-101)
Software	:	Matlab, OpenChoice Desktop

Items to bring

Essentials. A full list is available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/databook/essentials/

Before you come to the lab, it is essential that you read through this document and complete ***all*** of the preparation work in section 2. If possible, prepare for the lab with your usual lab partner. Only preparation which is recorded in your laboratory logbook will contribute towards your mark for this exercise. There is no objection to several students working together on preparation, as long as all understand the results of that work. Before starting your preparation, read through all sections of these notes so that you are fully aware of what you will have to do in the lab.

Academic Integrity – *If you undertake the preparation jointly with other students, it is important that you acknowledge this fact in your logbook. Similarly, you may want to use sources from the internet or books to help answer some of the questions. Again, record any sources in your logbook.*

This exercise uses the standard **mark scheme** available on the Laboratory website at https://secure.ecs.soton.ac.uk/notes/ellabs/2/

1 Aims, Learning Outcomes and Outline

By the end of this lab you should be able to:

- Analyse pulse code modulation (PCM) and its transmission and reception issues;
- Analyse real-time transmission of speech signals using the EMONA Telecoms-Trainer 101 (ETT-101);
- Model deterministic and random signals using Matlab;
- Investigate the use of cross-correlation for estimating signal delay.

Section 2 of this document details the preparation that should be completed before attending the corresponding lab session. Then, in Section 3 you will use ETT-101 for implementing and analysing PCM. An appendix is provided at the end of this document, in order to explain the conventions adopted by the ETT-101.
In Section 4 and onwards, you will use Matlab to simulate and analyse deterministic and random signals.

2 Preparation

2.1 Matlab

Read the tutorials available at

`https://secure.ecs.soton.ac.uk/notes/elec6238/matlab/lab1.pdf`

Start Matlab and try out simple examples (*e.g.*, the ones in the tutorial). Use "`help fun`" to get online help of the usage of the function `fun`. Use an editor, *e.g.*, Matlab's editor (type `edit`), to write script files for examples, requiring more than a few commands.

2.2 Background reading

Revise the material on stochastic signals, probability, correlation, LTI systems, Pulse code modulation (PCM).
Read the theoretical background on PCM and the ETT-101 PCM encoder and decoders provided in **Section 3.1**.

2.3 Unit Sample and Unit Step Sequences

Two basic discrete-time sequences are the unit sample sequence $u[n]$ and the unit step sequence $s[n]$, which can be defined as:

$$u[n] = \begin{cases} 1, \text{ for } n = 0 \\ 0, \text{ for } n \neq 0. \end{cases} \tag{1}$$

$$\text{and } s[n] = \begin{cases} 1, \text{ for } n \geq 0 \\ 0, \text{ for } n < 0. \end{cases} \tag{2}$$

A unit sample sequence u[n] of length N can be generated using the MATLAB command $u = [1 \text{ zeros}(1,N-1)]$, while a unit sample sequence ud[n] of length N and delayed by M samples, where $M < N$, can be generated using the MATLAB command $ud = [\text{zeros}(1,M)\ 1\ \text{zeros}(1,N - M - 1)]$.

A unit step sequence s[n] of length N can be generated using the MATLAB command $s = [\text{ones}(1,N)]$.

```
% exercise1.m
% Generation of a Unit Sample Sequence
n = -20:20; % Generate a vector from -20 to 20
u = [zeros(1,20) 1 zeros(1,20)]; % Generate the unit sample sequence

% Plot the unit sample sequence
stem(n,u);
xlabel('n');
ylabel('u[n]');
axis([-20 20 0 1.2]);
```

2.3.1 Questions

1. exercise1.m can be used to generate and plot a unit sample sequence. Run exercise1.m to generate the unit sample sequence u[n] and display it.
2. Modify the program to generate a delayed unit sample sequence ud[n] with a delay of 10 samples. Run the modified program and display the sequence generated.
3. Modify the program to generate a unit step sequence s[n]. Run the modified program and display the sequence generated.
4. Modify the program to generate a delayed unit step sequence sd[n] with an delay of 10 samples. Run the modified program and display the sequence generated.

3 Pulse Code Modulation

3.1 Theoretical Background on PCM

3.1.1 PCM Encoder

PCM is a system for converting analogue message signals to a serial stream of '0's and '1's. The conversion process is called encoding. At its simplest, encoding involves:

- Sampling the analogue signal's voltage at regular intervals using a sample-and-hold scheme.
- Comparing each sample to a set of reference voltages called quantisation levels.
- Deciding which quantisation level the sampled voltage is closest to.
- Generating the binary number for that quantisation level.
- Outputting the binary number one bit at a time (that is, in serial form).
- Taking the next sample and repeating the process.

An issue that is crucial to the performance of the PCM system is the encoder's clock frequency. The clock tells the PCM encoder when to sample and this must be at least twice the message frequency to avoid aliasing (or, if the message contains more than one sine wave, at least twice its highest frequency). Another important PCM performance

issue relates to the difference between the sample voltage and the quantisation levels that it is compared to. To explain, most sampled voltages will not be the same as any of the quantisation levels. As mentioned above, the PCM Encoder assigns to the sample the quantisation level that is closest to it. However, in the process, the original sample's value is lost and the difference is known as quantisation error. More importantly, the error is reproduced when the PCM data is decoded by the receiver because there is no way for the receiver to know what the original sample voltage was. The size of the error is affected by the number of quantisation levels. The more quantisation levels there are (for a given range of sample voltages) the closer they are together and the smaller the difference between them and the samples.

A little information about the ETT-101 PCM Encoder module: The PCM Encoder module uses a PCM encoding and decoding chip (called a codec) to convert analog voltages between -2V and +2V to an 8-bit binary number. With eight bits, it's possible to produce 256 different numbers between 00000000 and 11111111 inclusive. This in turn means that there are 256 quantisation levels (one for each number). Each binary number is transmitted in serial form in frames. The number's most significant bit (called bit-7) is sent first, bit-6 is sent next and so on to the least significant bit (bit-0). The PCM Encoder module also outputs a separate Frame Synchronisation signal (FS) that goes high at the same time that bit-0 is outputted. The FS signal has been included to help with PCM decoding but it can also be used to help 'trigger' a scope when looking at the signals that the PCM Encoder module generates. Figure 1 shows an example of three frames of a PCM Encoder module's output data (each bit shown as both a 0 and a 1 because it could be either) together with its clock input and its FS output.

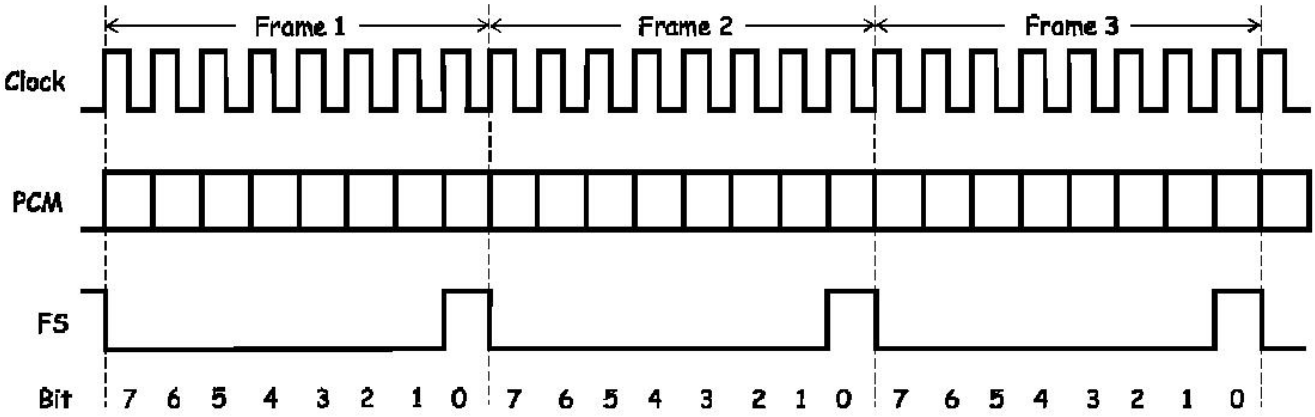

Figure 1:

3.1.2 PCM Decoder

The experiment in Section 3.2 would introduce you to the basics of Pulse Code Modulation (PCM) which is a system for converting message signals to a continuous serial stream of binary numbers (encoding). Recovering the message from the serial stream of binary numbers is called decoding.

At its simplest, decoding involves:

- Identifying each new frame in the data stream.
- Extracting the binary numbers from each frame.
- Generating a voltage that is proportional to the binary number.

- Holding the voltage on the output until the next frame has been decoded (forming a **Pulse Amplitude Modulation (PAM)** version of the original message signal).
- Reconstructing the message by passing the PAM signal through a low-pass filter.

The PCM decoder's clock frequency is crucial to the correct operation of simple decoding systems. If it's not the same frequency as the encoder's clock, some of the transmitted bits are read twice while others are completely missed. This results in some of the transmitted numbers being incorrectly interpreted, which in turn causes the PCM decoder to output an incorrect voltage. The error is audible if it occurs often enough. Some decoders manage this issue by being able to 'self-clock'.

There is another issue crucial to PCM decoding. The decoder must be able to detect the beginning of each frame. If this isn't done correctly, every number is incorrectly interpreted. The synchronising of the frames can be managed in one of two ways. The PCM encoder can generate a special frame synchronisation signal that can be used by the decoder though this has the disadvantage of sending two signals. Alternatively, a frame synchronisation code can be embedded in the serial data stream that is used by the decoder to work out when the frame starts.

A little information about the ETT-101 PCM Decoder module: Like the PCM Encoder module on the ETT-101, the PCM Decoder module works with 8-bit binary numbers. For 00000000 the PCM Decoder module outputs -2V and for 11111111 it outputs +2V. For numbers in between, the output is a proportional voltage between $\pm$2V. For example, the number 10000000 is half way between 00000000 and 11111111 and so for this input the module outputs 0V (which is half way between +2V and -2V). The PCM Decoder module is not self-clocking and so it needs a digital signal on the CLK input to operate. More importantly, for the PCM Decoder module to correctly decode PCM data generated by the PCM Encoder module, it must have the same clock signal. In other words, the decoder's clock must be 'stolen' from the encoder. Similarly, the PCM Decoder module cannot self-detect the beginning of each new frame and so it must have a frame synchronisation signal on its FS input to do this.

In this experiment, you'll use the the ETT-101 equipment to PCM-encode and PCM-decode a sinewave and a speech signal. For this to work correctly, the PCM decoder's clock and frame synchronisation signal are simply 'stolen' from the PCM Encoder module.

3.2 Setting up the PCM encoder

The first part of the experiment gets you to set up a PCM encoder.

1. Gather a set of the equipment: an ETT-101 trainer set (plus headphone) and an oscilloscope.
2. Set up the scope. Ensure that: the Trigger Source control is set to the CH1 position. the Mode control is set to the CH1 position. You may use the AUTOSET function of a digital scope.
3. Locate the PCM Encoder module and set its Mode switch to the PCM position.
4. Connect the set-up shown in Figure 2 below.
   Note: Insert the black plugs of the oscilloscope leads into ground (GND) sockets.

   This set-up can be represented by the block diagram in Figure 3. The PCM Encoder module is clocked by the Master Signals module's 100kHz DIGITAL output. Its analog input is the Variable DC module's VDC output.
5. Adjust the scope's Timebase control to view one pulse of the PCM Encoder module's FS output.
   Tips: The 10 μs/div setting is the best to use. Set the Variable DCV module's Variable DC control to about the middle of its travel.

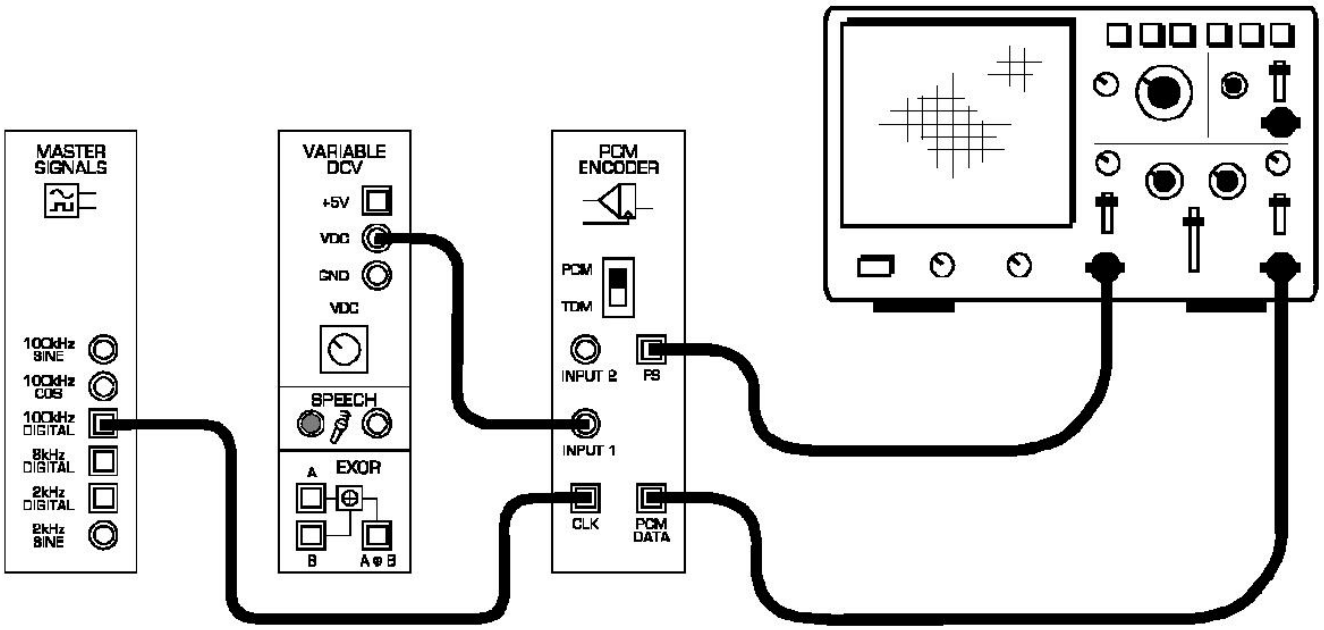

Figure 2:

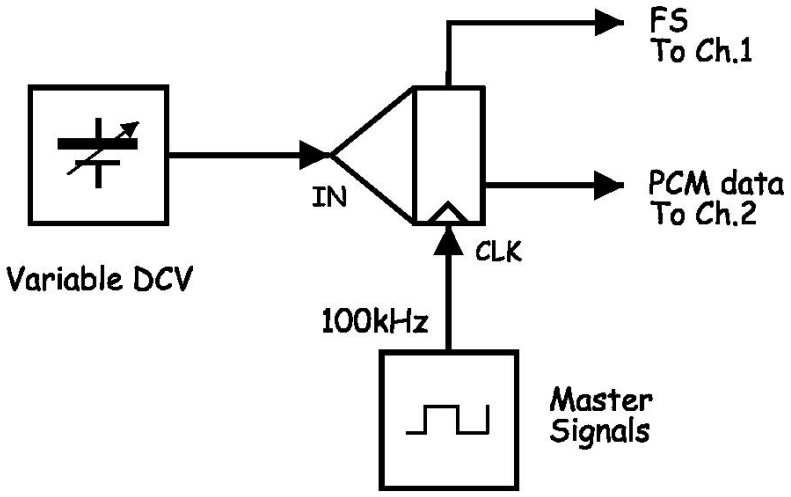

Figure 3:

6. Turn on ‘CH 2’ to view the PCM Encoder module’s PCM DATA output as well as its FS output.

7. Vary the Variable DCV module’s Variable DC control left and right.
   If your set-up is working correctly, this last step should cause the number on PCM Encoder module’s PCM DATA output to go down and up. If it does, carry on to the next step. If not, check your wiring or ask the instructor for help.

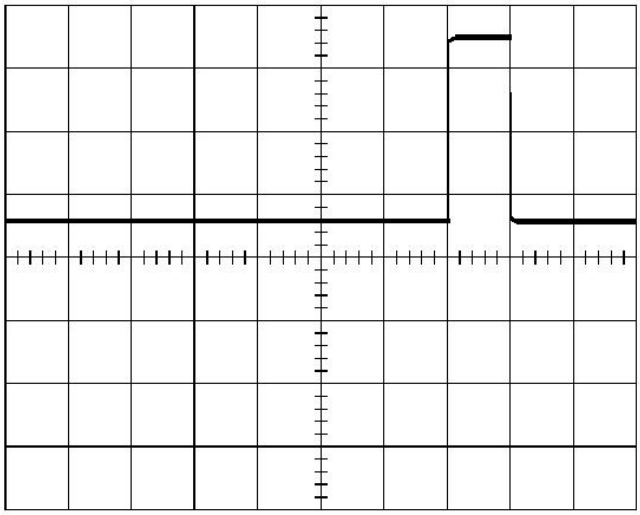

Figure 4:

Display on the scope the FS output (on CH. 1), the PCM data (on CH. 2) and the CLK input (on CH. 3). Adjust the Variable DCV module’s Variable DC control to its middle. Print and stick (or sketch) the waveforms onto your logbook. Indicate on the printout/sketch i) the start and end of the frame, ii) the start and end of each bit, iii) which bit is bit-0 and which is bit-7.

Question 1
What is the binary number that the PCM Encoder module is outputting?

---

8. See if you can vary the Variable DC control left and right without causing the output code to change.
   The sampled voltage can be changed without causing the output code to change because it is compared to a set of quantisation levels but there are a finite number of them. This means that, in practice, there’s a range of sample voltages for each quantisation level.

Question 2
What’s the name for the difference between a sampled voltage and its closest quantisation level?

Question 3
It’s possible to work out how far apart a PCM encoder’s quantisation levels are, by using the information you’ve gathered so far. Given that the PCM Encoder convert analogue voltages between -2V and +2V to an 8-bit binary

number. Calculate the difference between the quantisation levels.

Question 4
What is the maximum quantisation noise in this case?

Question 5
To reduce quantisation error is it better to have fewer or more quantisation levels between ± 2V? How do you change the number of quantisation levels?

---

9. Disconnect the plug to the Variable DCV module's VDC output.
10. Locate the VCO module and turn its Frequency Adjust control fully anti-clockwise.
11. Set the VCO module's Range control to the LO position.
12. Modify the set-up as shown in Figure 5.
    Remember: Dotted lines show leads already in place.

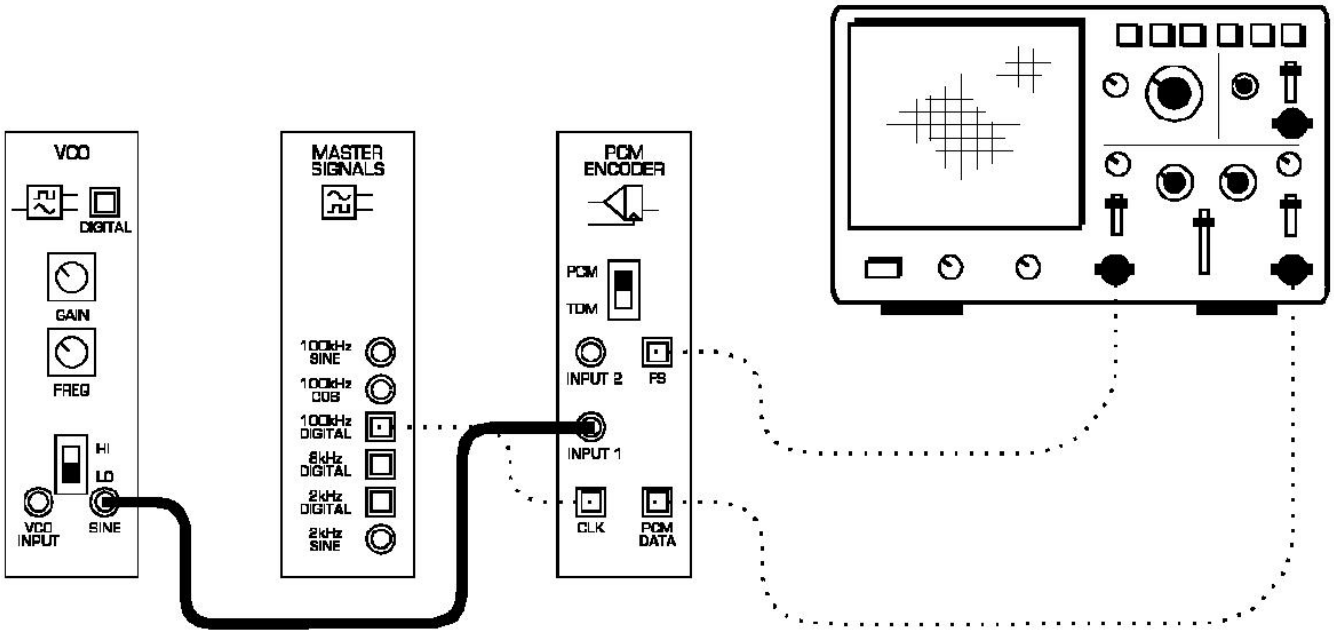

Figure 5:

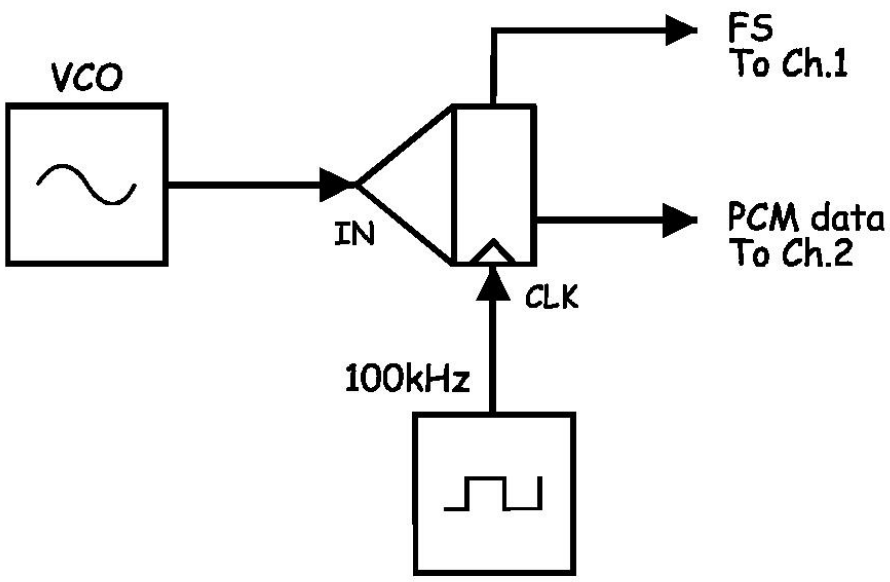

Figure 6:

This set-up can be represented by the block diagram in Figure 6. Notice that the PCM Encoder module's input is now the VCO module's SINE output.

As the PCM Encoder module's input is a sinewave, the module's input voltage is continuously changing. This means that you should notice the PCM DATA output changing continuously also.

3.3 Decoding the PCM data

1. Set the scope's Mode control to CH1 position.
2. Modify the set-up as shown in Figure 7.
   Note: set the VCO module's Range control to the LO position.

   The entire set-up can be represented by the block diagram in Figure 8. Notice that the decoder's clock and frame synchronisation information are 'stolen' from the encoder.
3. Adjust the scope's Timebase control to view two or so cycles of the message.
4. Display the PCM Decoder module's output as well as the message signal on the scope. Press the 'RUN/STOP' button on the scope to freeze the display if needed.

   Question 6
   What does the PCM Decoder's 'stepped' output tell you about the type of signal that it is? (**Tips:** If you're not sure, see the preliminary discussion at Section 3.1.2.)

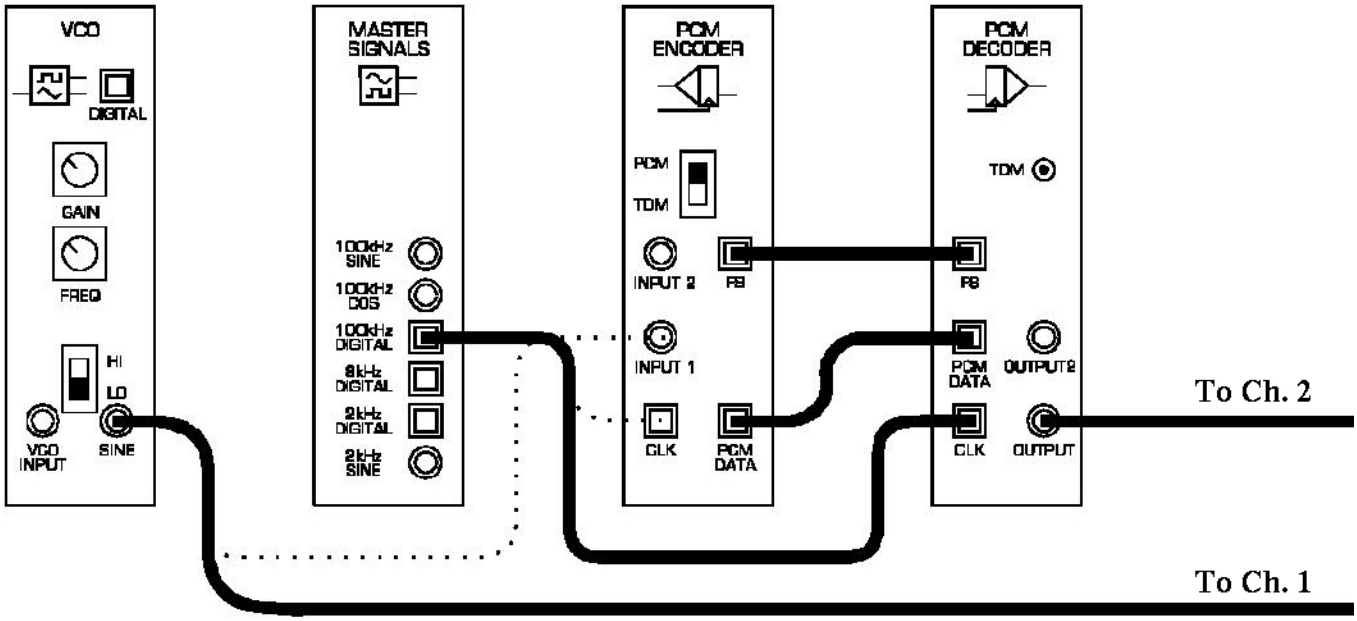

Figure 7:

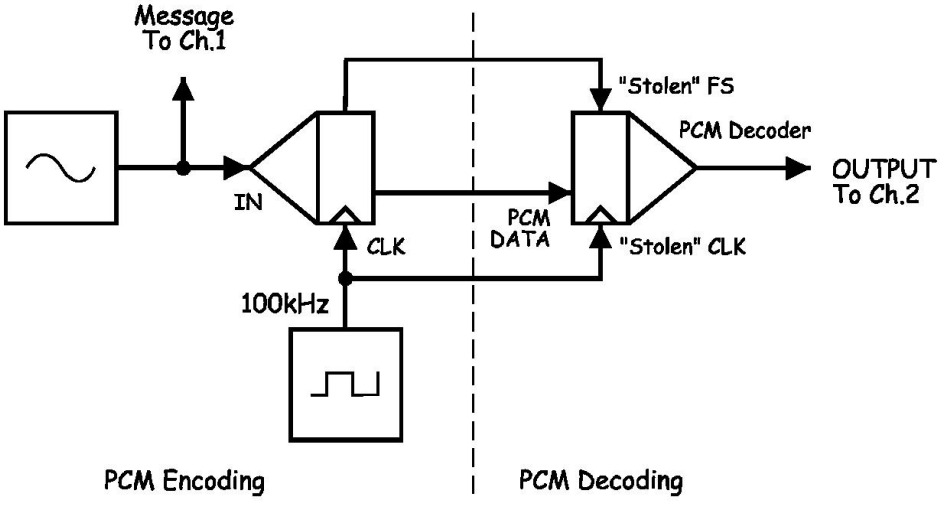

Figure 8:

The PCM Decoder module's output signal looks very similar to the message. However, they're not the same. A sampled message contains many sinewaves in addition to the message. This can be better appreciated if you compare the message and the PCM Decoder module's output by listening to them.

5. Add the Buffer module to the set-up as shown in Figure 9 leaving the scope's connections as they are.

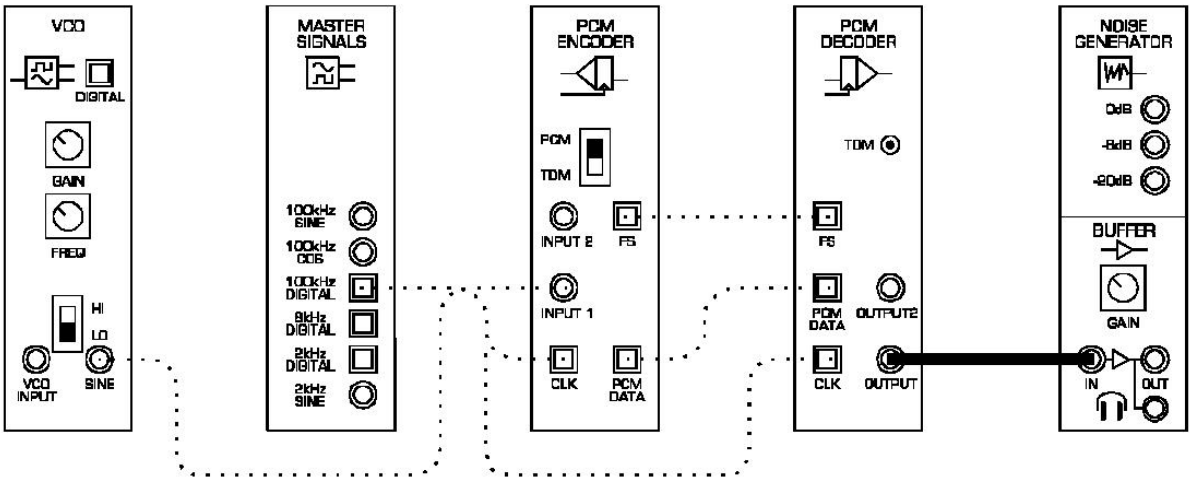

Figure 9:

6. Turn the Buffer module's Gain control fully anti-clockwise.
7. Without wearing the headphones, plug them into the Buffer module's headphone socket.
8. Put the headphones on.
9. Turn the Buffer module's Gain control clockwise until you can comfortably hear the PCM Decoder module's output.
10. Disconnect the Buffer module's lead where it plugs to the PCM Decoder module's output.
11. Modify the set-up as shown in Figure 10, again leaving the scope's connections as they are. Compare the sound of the two signals. You should notice that they're similar but clearly different.

    Question 7
    What must be done to the PCM Decoder module's output in order to reconstruct the message appropriately? (**Tips:** If you're not sure, see the preliminary discussion at Section 3.1.2.)

3.4 Encoding and decoding speech

So far, this experiment has encoded and decoded a sine wave for the message. The next part of the experiment lets you do the same with speech.

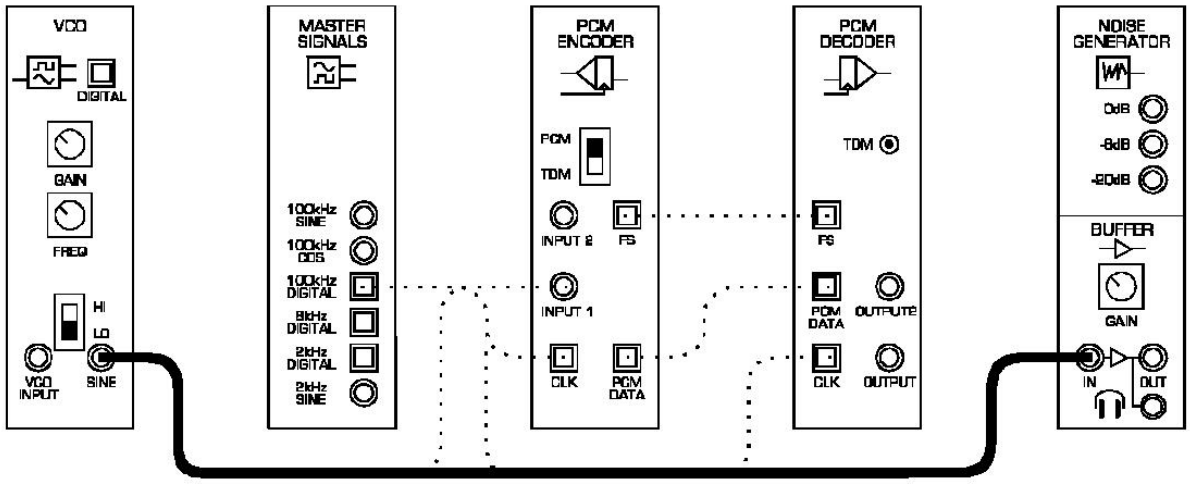

Figure 10:

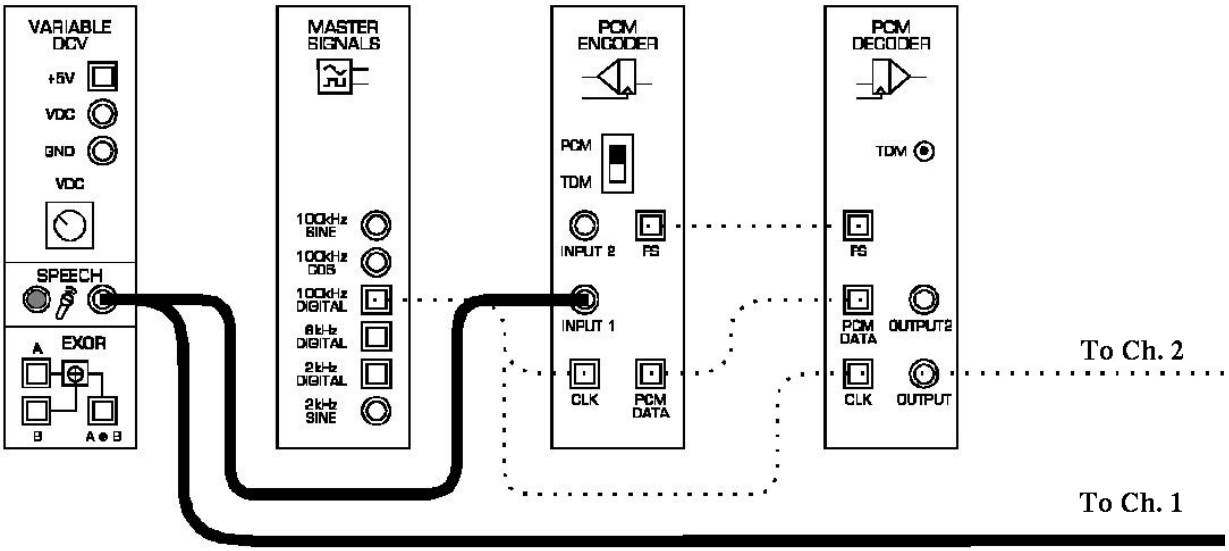

Figure 11:

1. Completely remove the Buffer module from the set-up while leaving the rest of the leads in place.
2. Disconnect the plugs to the VCO module's SINE output.
3. Modify the set-up as shown in Figure 11.
4. Talk, sing or hum while watching the scope's display.

3.5 Recovering the message

The message can be reconstructed from the PCM Decoder module's output signal using a low-pass filter. This part of the experiment lets you do this.

1. Locate the Tuneable Low-pass Filter module and set its Gain control to about the middle of its travel.
2. Turn the Tuneable Low-pass Filter module's Cut-off Frequency Adjust control fully anti-clockwise.
3. Disconnect the plugs to the Speech module's output.
4. Modify the set-up as shown in Figure 12.

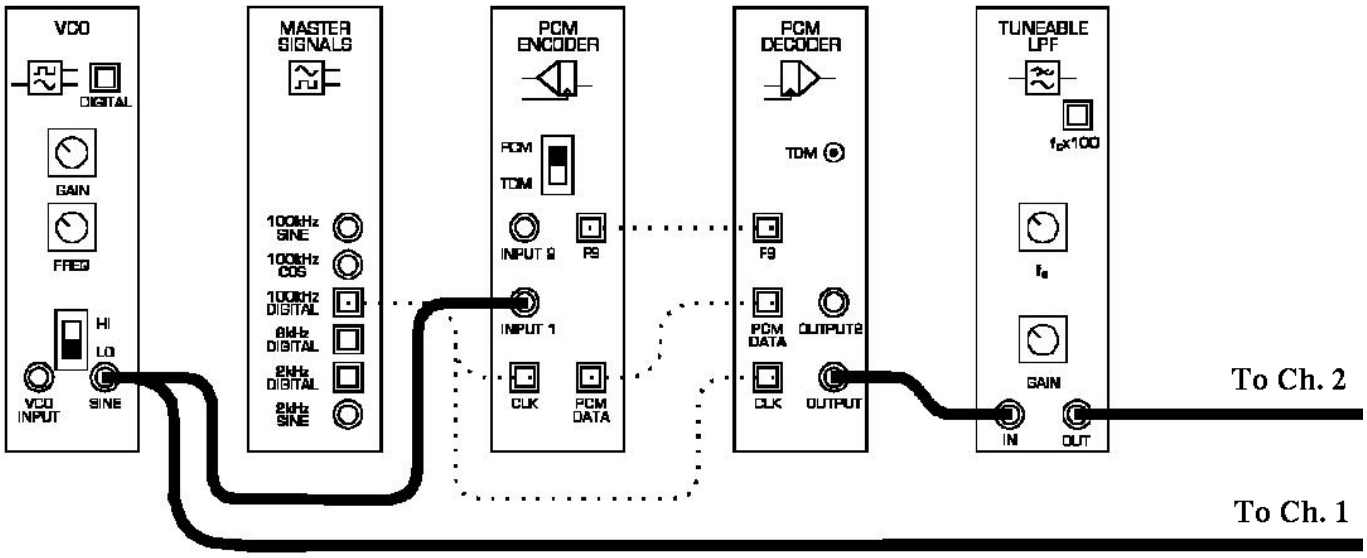

Figure 12:

The entire set-up can be represented by the block diagram in Figure 13. The Tuneable Low-pass Filter module is used to reconstruct the original message from the PCM Decoder module's PAM output.

5. Slowly turn the Tuneable Low-pass Filter module's Cut-off Frequency control clockwise and stop the moment the message signal has been reconstructed (ignoring phase shift). The two signals are clearly the same so let's see what your hearing tells you.
6. Add the Buffer module to the set-up as shown in Figure 14 leaving the scope's connections as they are.

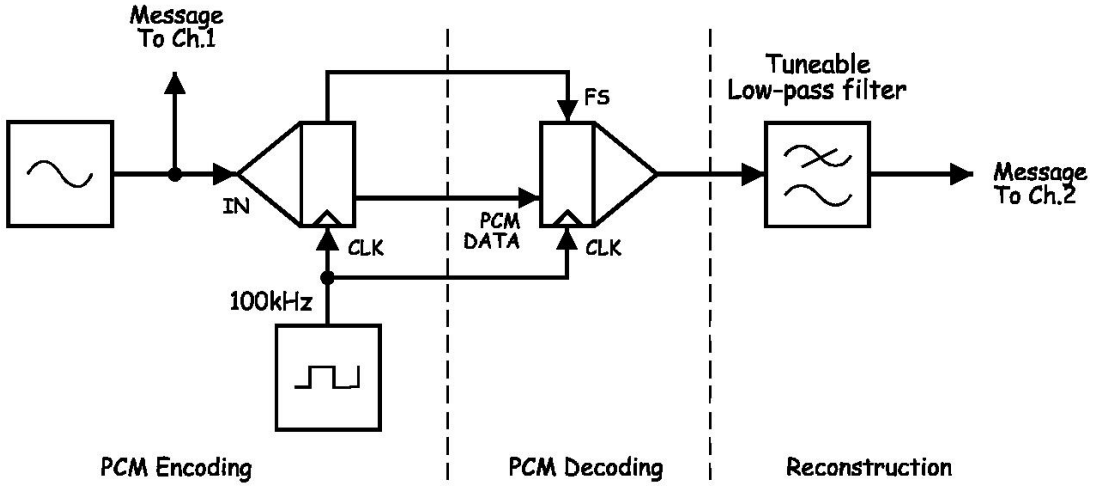

Figure 13:

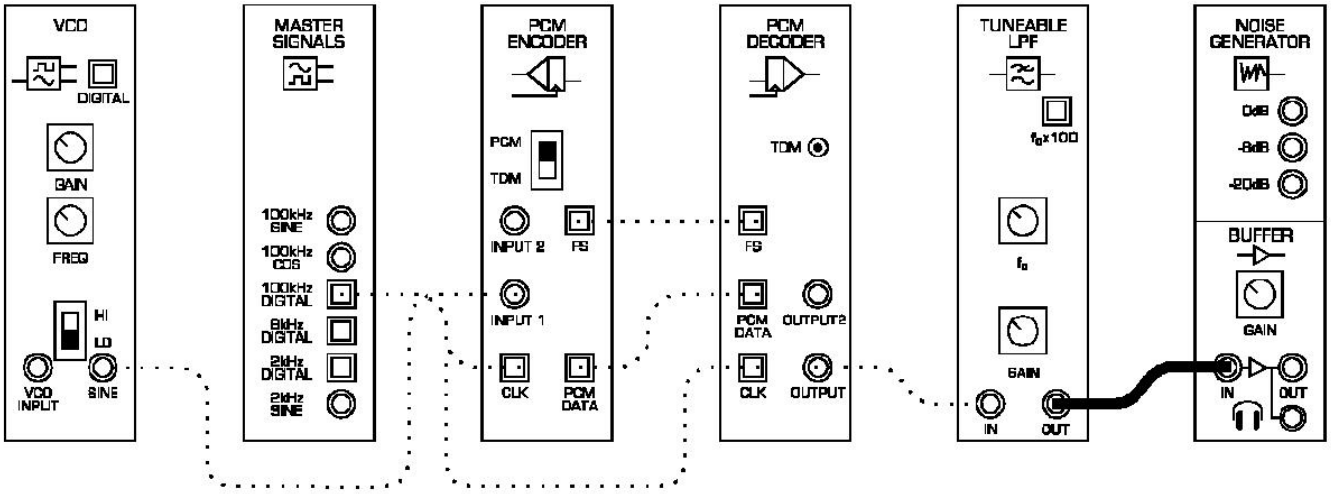

Figure 14:

7. Turn the Buffer module's Gain control fully anti-clockwise.
8. Put the headphones on.
9. Turn the Buffer module's Gain control clockwise until you can comfortably hear the Tuneable Low-pass Filter module's output.
10. Disconnect the Buffer module's lead where it plugs to the PCM Decoder module's output and connect it to the VCO module's output (like you did when wiring Figure 10).
11. Compare the sound of the two signals. You should find that they're very similar.

Question 8
Even though the two signals look and sound the same, why isn't the reconstructed message a perfect copy of the original message?

_____

4 Sinusoidal Sequences

Another very useful class of discrete-time signals is the real sinusoidal sequence of the form $x[n] = A \cdot sin(\omega_0 n + \phi)$. Such sinusoidal sequences can be generated in MATLAB using the trigonometric operators cos and sin.

4.1 Questions

1. Consider the following code:

```
clear all, close all
T = 1; ts = 0.01; % time horizon and discretization time
t = 0:ts:T;       % time vector
phi = 0;
u = sin(2*pi*t + phi);  % signal

figure, hold on, stem(t,u,'k')
title('Signal'), xlabel('time, t'), ylabel('u(t)')
```

   (a) Explain what the above code does.
   Execute the above code and observe the effect of modifying the `T` and `ts` parameters.

   (b) Save the above code into a file. Modify the code to generate a sinusoidal sequence of frequency 0.9 and display it. Compare this new sequence with the one generated in Question (a).

   (c) Now, modify the program to generate a sinusoidal sequence of frequency 1.1 and display it. Compare this new sequence with the one generated in Question (a). Comment on your results.

2. *Time and phase shift:*
   Assuming that the code above is executed first, explain what the following code does:

```
u1 = sin(2*pi*(t+1/4)); % positive time shift
u2 = sin(2*pi*(t-1/4)); % negative time shift
u3 = sin(2*pi*t+pi/2);  % positive phase shift
u4 = sin(2*pi*t-pi/2);  % negative phase shift

figure, hold on
stem(t,u,'k'), stem(t,u1,'r'), stem(t,u2,'b')
title('Original signal and its time shifts')
xlabel('time, t'), ylabel('u(t), u1(t), u2(t)')

figure, hold on
stem(t,u,'k'), stem(t,u3,'r'), stem(t,u4,'b')
title('Original signal and its phase shifts')
xlabel('time, t'), ylabel('u(t), u3(t), u4(t)')
```

   Comment on the results.

3. *Time scaling:*
   Assuming that the code above is executed first, explain what the following code does:

```
u5 = sin(2*pi*(2*t));   % time scale by constant > 1
u6 = sin(2*pi*(1/2*t)); % time scale by constant < 1

figure, hold on
stem(t,u,'k'), stem(t,u5,'r'), stem(t,u6,'b')
title('Original signal and its time scaled versions')
xlabel('time, t'), ylabel('u(t), u5(t), u6(t)')
```

   Comment on the results.

5 Random Signals

A random signal of length N with samples uniformly distributed in the interval (0,1) can be generated by using the MATLAB command x = rand(1,N);

Likewise, a random signal x[n] of length N with samples normally distributed with zero mean and unity variance can be generated by using the following MATLAB command x = randn(1,N);

5.1 Questions

1. Write a MATLAB program to generate and display a random signal of length 100000 whose elements are uniformly distributed in the range $[-4,4]$. Display the probability density function (PDF) of the signal using the Matlab 'hist' command. Also display the cumulative distribution function (CDF) of the signal.

2. Write a MATLAB program to generate and display a Gaussian random signal of length 100000 whose elements are normally distributed with zero mean and a variance of 2. Display the PDF and CDF of the signal.

3. Using the functions `randn` and `hist`, generate a sequence of N independent realizations of a zero mean, unit variance, Gaussian random variable, and plot the histogram. Repeat the experiment with $N = 100, 500, 2000, 10000$ realizations and comment on the results. Plot on top of the histogram the theoretical probability density function.

6 Delay estimation by cross-correlation

A signal u is sent out, takes time τ to reach an object and is reflected back and received. In total, the signal is delayed by a time 2τ, attenuated by an amount a and is subject to noise n. Thus, the received signal is modelled by

$$y(t) = au(t-2\tau) + n(t). \tag{3}$$

You know u and y, but because of the noise you are uncertain about τ, which is what you want to know, to determine the distance of the object from the sender/receiver. You also don't know the noise, but you can assume that u and n are uncorrelated. Explain how to find τ. Apply your method on the data available from

`https://secure.ecs.soton.ac.uk/notes/ellabs/2/c6/delay.mat`

which is the received signal $y(t)$ in (3).

Hint: you can use the Matlab function `load('delay.mat')` to load the data, where you will get two vectors `y` and `u`.

7 Filtering

Consider a received signal

$$y(t) = u(t) + n(t), \tag{4}$$

where u is the transmitted signal and n is a zero mean white Gaussian noise. The transmitted signal is smooth compared with the noise. Using this knowledge and the fact that the noise is zero mean white Gaussian, explain how to improve the signal-to-noise ratio of the received signal. Apply your method on the data available from

`https://secure.ecs.soton.ac.uk/notes/ellabs/2/c6/filtering.mat`

Hint: You can use the following code to simulate the filter coefficients a and b:

```
M=250;
b = ones(1,M)/M;
a = 1;
```

You can use these filter coefficients using the Matlab function "filter(b,a,y)".

8 Appendix: ETT-101 System Conventions

The front panel of the EMONA Telecoms-Trainer 101 (ETT-101) has been laid out following a series of front panel conventions. All ETT-101 modules, for example the module shown in Figure 15, conform to the following mechanical and electrical conventions:

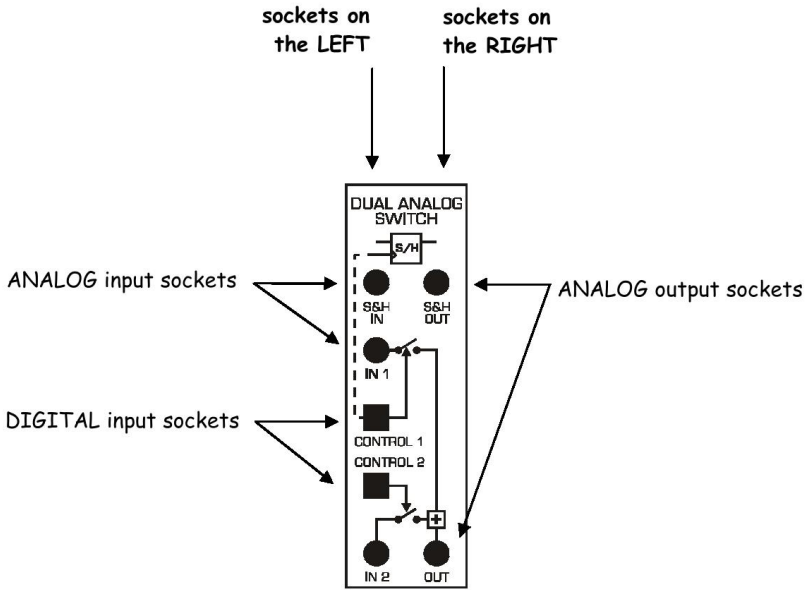

Figure 15:

- Signal interconnections are made via front panel, 2mm sockets.
- Sockets on the **LEFT HAND SIDE** are for signal **INPUTS**. All inputs are high impedance, either 10k ohms or 56k ohms depending on the module, in order to reduce effects when connections are made and broken.
- Sockets on the **RIGHT HAND SIDE** are for signal **OUTPUTS**. All analog outputs are low impedance, typically 330 ohms. Again, this is to reduce effects when connections are made and broken. Digital outputs are typically 47 ohms.
- ROUND sockets, "○", are only for ANALOG signals. ANALOG signals are typically held near the ETT-101 standard reference level of 4V pk-pk.
- SQUARE sockets, "□", are only for DIGITAL signals. DIGITAL level signals are TTL level, 0 to 5 V.
- ROUND sockets labelled GND, "○", are common, or system GROUND.

⇨ **Warning:** Each of the scope probe has a red lead and a black lead. Plug the **black lead** to the ground (GND) socket. It will not work if you plug the red lead to the GND.

⇨ **Note:** There are two GND sockets at the middle of the trainer and one GND socket at the 'Variable DCV' module. It is possible to create more than 3 GNDs using the jumpers provided.

If you are using a digital scope, you may use the 'AUTOSET' button for most of the scope settings or use the 'AUTOSET' button for attaining the initial scope setting.