Department of Electronics and
Electrical Communications Engineering
Faculty of Engineering – Cairo University

Fourth Year Communications Lab

Volume 1 Experiment 21

Volume 2 Experiment 14

Name: _____

Class: _____

21 - DSSS modulation and demodulation

Experiment 21 – DSSS modulation and demodulation

Preliminary discussion

Recall that when a sinusoidal carrier is DSBSC modulated by a message, the two signals are multiplied together. Recall also that the resulting DSBSC signal consists of two sets of sidebands but no carrier (refer to the preliminary discussion of Experiment 6 for a discussion of this).

When the DSBSC signal is demodulated using product detection, both sidebands are multiplied with a local carrier that must be synchronised to the transmitter's carrier (that is, it has the same frequency and phase). Doing so produces two messages that are in-phase with each other and so add to form a single bigger message (refer to the preliminary discussion of Experiment 9 for a discussion of this).

Direct sequence spread spectrum (DSSS or often just "spread spectrum") is a variation of the DSBSC modulation scheme with a pulse train (called a *pseudo-noise* sequence or just PN sequence) for the carrier instead of a simple sinewave. This may sound radical until you remember that pulse trains are actually made up of a theoretically infinite number of sinewaves (the *fundamental* and *harmonics*). That being the case, spread spectrum is really the DSBSC modulation of a theoretically infinite number of sinusoidal carrier signals. The result is a theoretically infinite number of pairs of tiny sidebands about a suppressed carrier.

In practice, not all of these sidebands have any energy of significance. However, the fact that the message information is distributed across so many of them makes spread spectrum signals difficult to deliberately interfere with or "jam". To do so, you would have to upset a significant number of the sidebands which is difficult considering their number.

Spread spectrum signals are demodulated in the same way as DSBSC signals using a product detector. Importantly, the product detector's local carrier signal must contain all the sinewaves that make up transmitter's pulse train at the same frequency and phase. If this is not done, the tiny demodulated signals will be at the wrong frequency and phase and so they won't add up to reproduce the original message. Instead, they'll produce a garbage signal that looks like noise.

The only way for the receiver to generate the right number of sinewaves at the right frequency is to use a pulse train with an identical sequence to that used by the transmitter. Moreover, it must be synchronised. This issue gives spread spectrum another of its advantages over other modulation schemes. The transmitted signal is effectively encrypted.

Of course, with trial and error it's possible for an unauthorised person to guess the correct PN sequence to use for their receiver. However, this can be made difficult by making the sequence longer before it repeats itself (that is, by making it consist of more bits or *chips*). Longer sequences can produce more combinations of unique codes which would take longer to guess using a trial and error approach. To illustrate this point, an 8-bit code has 256 combinations while a 20-bit code has 1,048,575 combinations. A 256-bit code has 1.1579×10^{77} combinations. That's 11579 with 73 zeros after it!

Increasing the sequence's chip-length has another advantage. To explain, the total energy in a spread spectrum signal is distributed between all of the tiny DSBSC that make it up (though not evenly because not all of the sinewaves that make up the carrier's pulse train are the same amplitude). A mathematical technique called *Fourier Analysis* shows that the greater the number of chips in a sequence before repeating, the greater the number of sinewaves of significance needed to make it.

That being the case, using more chips in the transmitter's PN sequence produces more DSBSC signals and so the signal's total energy is distributed more thinly between them. This in turn means that the individual signals are many and extremely small. In fact, if the PN sequence is long enough, all of these DSBSC signals are smaller than the background electrical noise that's always present in free-space. This fact gives spread spectrum yet another important advantage. The signal is difficult to detect.

Spread spectrum finds use in several digital applications including: CDMA mobile phone technology, cordless phones, the global positioning system (GPS) and two of the 802.11 wi-fi standards.

The experiment

For this experiment you'll use the Emona DATEx to generate a DSSS signal by implementing its mathematical model. You'll then use a product detector (with a stolen carrier) to reproduce the message. Once done, you'll examine the importance of using the correct PN sequence for the local carrier and the difficulty of jamming DSSS signals.

It should take you about 50 minutes to complete this experiment.

Equipment

- Personal computer with appropriate software installed
- NI ELVIS II plus USB cable and power pack
- Emona DATEx experimental add-in module
- Two BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads

Procedure

Part A – Generating a DSSS signal using a simple message

As DSSS is basically just DSBSC with a pulse train for the carrier instead of a simple sinusoid, it can be generated by implementing the mathematical model for DSBSC.

1. Ensure that the NI ELVIS II power switch at the back of the unit is off.

2. Carefully plug the Emona DATEx experimental add-in module into the NI ELVIS II.

3. Set the *Control Mode* switch on the DATEx module (top right corner) to *PC Control*.

4. Connect the NI ELVIS II to the PC using the USB cable.

   Note: This may already have been done for you.

5. Turn on the NI ELVIS II power switch at the rear of the unit then turn on its *Prototyping Board Power* switch at the top right corner near the power indicator.

6. Turn on the PC and let it boot-up.

7. Launch the NI ELVISmx software.

8. Launch the DATEx soft front-panel (SFP) and check that you have soft control over the DATEx board.

9. Locate the Sequence Generator module on the DATEx SFP and set its soft dip-switches to 00.

10. Connect the set-up shown in Figure 1 below.

Note: Insert the black plugs of the oscilloscope leads into a ground (*GND*) socket.

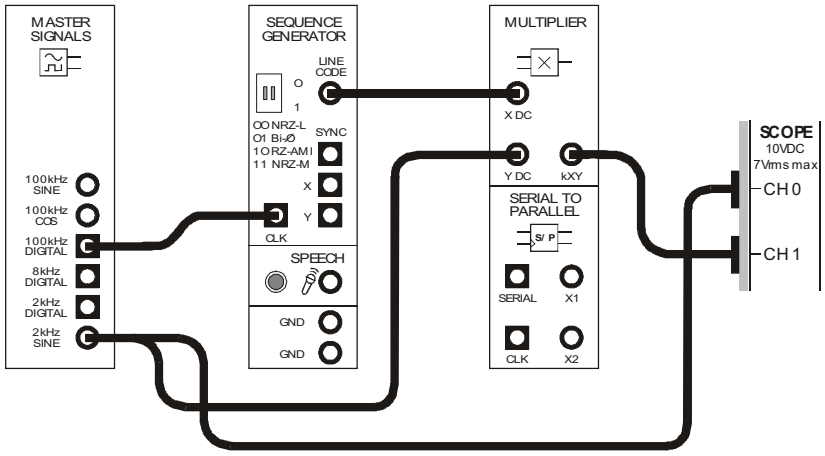


Figure 1

This set-up can be represented by the block diagram in Figure 2 below. It multiplies the 2kHz sinewave message with a PN sequence modelled by the Sequence Generator's 32-bit pulse train output.

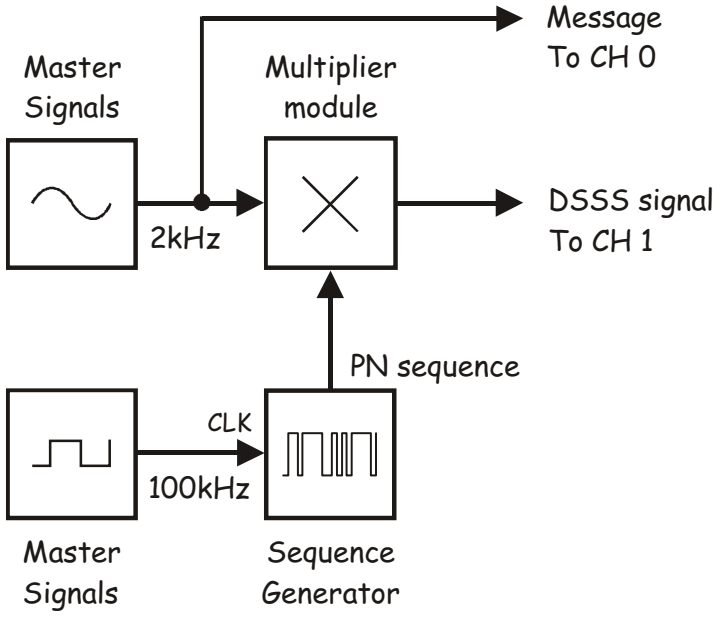


Figure 2

11. Launch and run the NI ELVIS II Oscilloscope VI.

12. Set up the scope per the instructions in Experiment 1 with the following changes:

    - *Timebase* control to *100µs/div* instead of *500µs/div*
    - Channel 1 *Scale* control to *2V/div* instead of *1V/div*

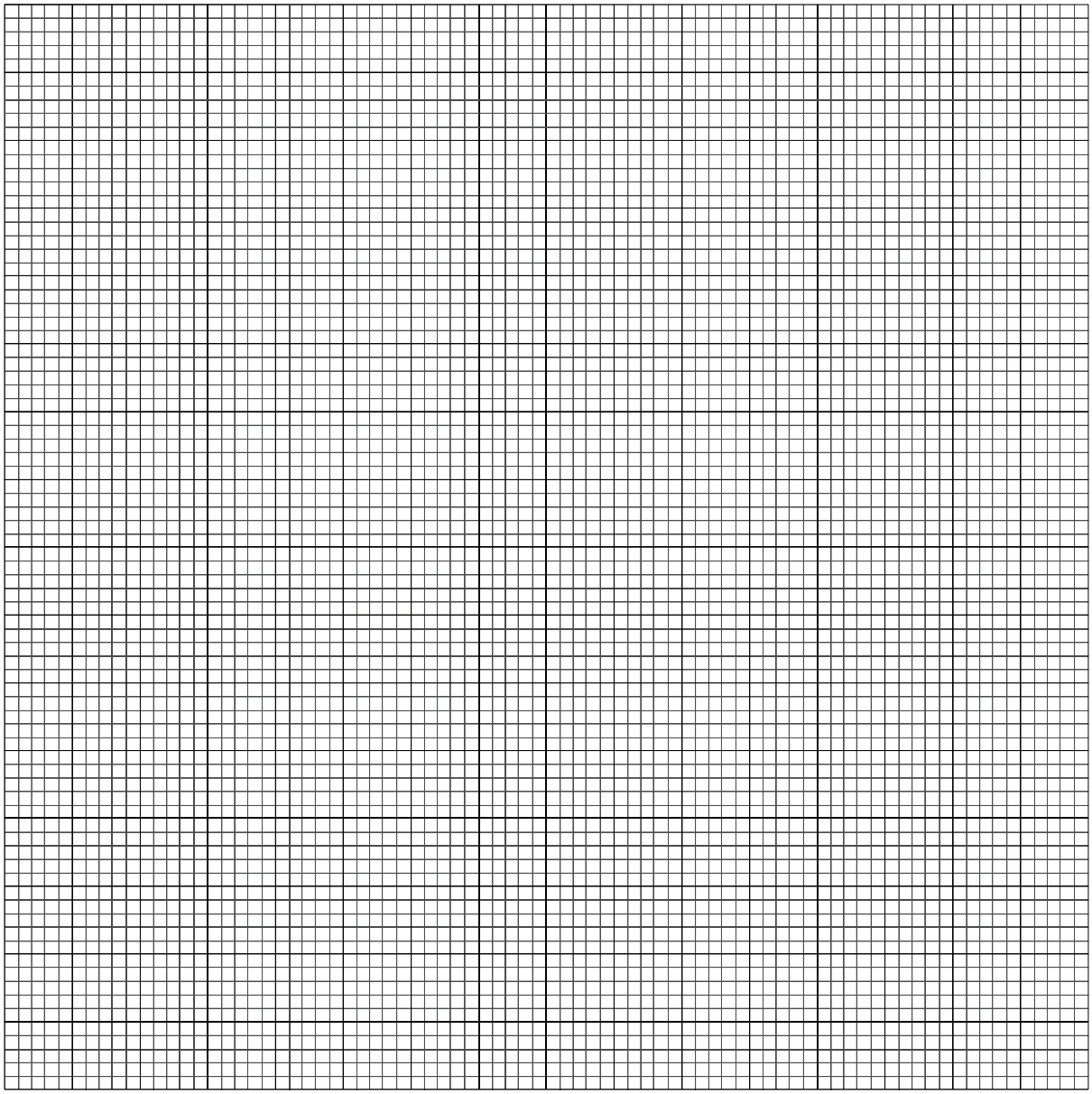
13. Activate the scope's Channel 1 input to observe the DSSS signal out of the Multiplier module as well as the message signal.

14. Draw the two waveforms to scale in the space provided on the next page leaving room to draw a third waveform.

    Tip: Draw the message signal in the upper third of the graph and the DSSS signal in the middle third.

Question 1
What feature of the Multiplier module's output suggests that it's basically a DSBSC signal? **Tip:** If you're not sure, read the preliminary discussion for Experiment 6.

Question 2
Why is the DSSS signal so large when it's supposed to be small and indistinguishable from noise? **Tip:** If you're not sure, see the preliminary discussion for this experiment.



Ask the instructor to check your work before continuing.

Part B – Observations of DSSS signals in the frequency domain

One of the features of DSSS is that it produces a theoretically infinite number of pairs of tiny sidebands with each pair straddling a suppressed carrier. This part of the experiment lets you examine this.

15. Launch and run the NI ELVIS II Function Generator VI.

16. Adjust the function generator using its soft controls for an output with the following specifications:

    - Waveshape: Square
    - Frequency: 30kHz
    - Amplitude: 4Vpp
    - DC Offset: 0V

17. Disconnect the plug to the Sequence Generator module's *LINE CODE* output and modify the set-up as shown in Figure 3 below.

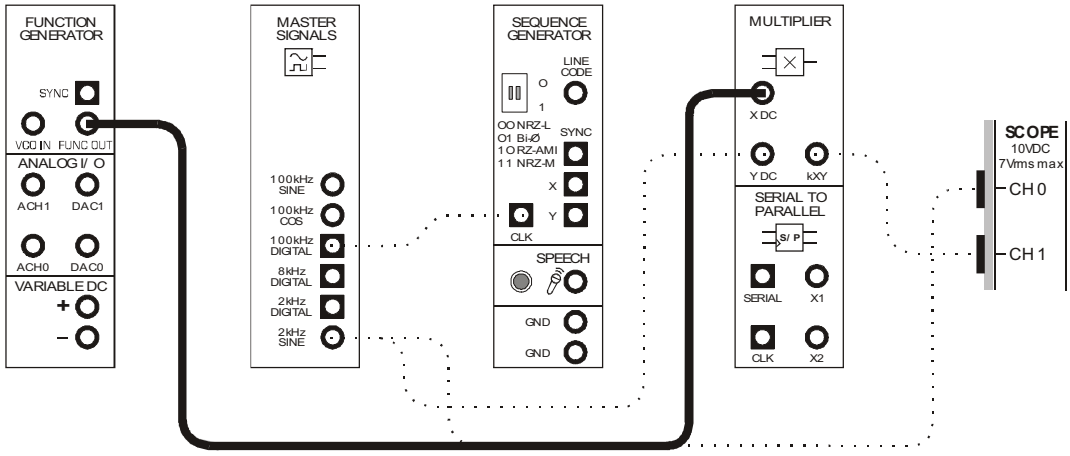


Figure 3

18. Examine the new DSSS signal on the scope.

    Note: You should notice that it looks similar to the DSSS signal you obtained earlier. That said, it'll be different in that the spacing between the carrier's transitions are regular.

The set-up in Figure 3 can be represented by the block diagram in Figure 4 below. Notice that the carrier signal is a 30kHz squarewave.

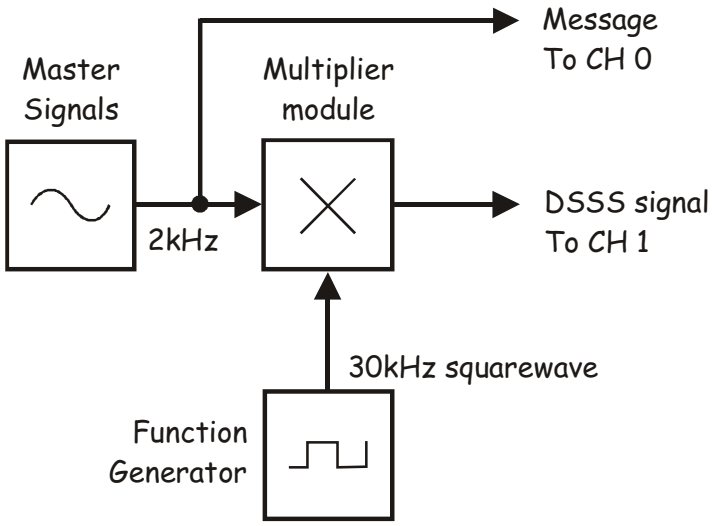


Figure 4

Recall that a squarewave consists of a fundamental at the same frequency as the squarewave itself and a theoretically infinite number of odd harmonics (each with proportionally smaller amplitude to the amplitude of the frequency before it). So, our 30kHz squarewave carrier consists of sinewaves at 30kHz, 90kHz, 150kHz, 210kHz and so on.

Theoretically then, the DSSS signal consists of a 30kHz suppressed carrier with 28kHz and 32kHz lower and upper sidebands, a 90kHz suppressed carrier with 88kHz and 92kHz lower and upper sidebands, a 150kHz suppressed carrier with 148kHz and 152kHz lower and upper sidebands, and so on. Let's examine these using the NI ELVIS II Dynamic Signal Analyzer virtual instrument.

19. Suspend the scope VI's operation by clicking on its *Stop* control once.

    Note: The scope's display should freeze.

20. Launch and run the NI ELVIS II Dynamic Signal Analyzer VI.

21. Adjust the signal analyzer's controls as follows:

    Input Settings

    - *Source Channel* to *SCOPE CH 1*
    - Voltage Range to ±10V

    FFT Settings

    - *Frequency Span* to *200,000*
    - *Resolution* to *400*
    - *Window* to *7 Term B-Harris*

    Averaging

    - *Mode* to *RMS*
    - *Weighting* to *Exponential*
    - *# of Averages* to *3*

    Trigger Settings

    - *Type* to *Edge*

    Frequency Display

    - *Units* to *dB*
    - *Mode* to *RMS*
    - *Scale* to *Auto*
    - *Cursors On* box unchecked (for now)

The display should now be showing about ten pairs of what appear to be significant sinewaves. This is deceptive as you'll see.

22. Activate the signal analyzer's cursors by checking the *Cursors On* box.

23. Use the signal analyzer's *C1* cursor to measure the frequency in the middle of each pair of the sinewaves.

    Note: You'll find that the signal consists of pairs of sidebands about a suppressed carrier at frequencies listed in the second last paragraph of the previous page.

    You'll also find that it consists of sidebands about suppressed carriers at other frequencies. However, although these signals are present, the display is a little misleading because the vertical axis is logarithmic (i.e. non-linear).

24. Change the signal analyzer's *Units* control (under the *Frequency Display* heading) from *dB* to *Linear*.

    Note: This display shows you the linear relationship between the sinewaves' amplitude.

25. Use the signal analyzer's *C1* cursor to measure the frequency of these significant sinewaves.

    Note: The frequencies should be identical to those listed on the bottom of page 21-9.

26. Return the signal analyzer's *Units* control to the *dB* position.

27. Disconnect the patch lead from the function generator's output and return it to the Sequence Generator module's *LINE Code* output.

    Note: This returns the set-up to that shown in Figures 1 and 2 with a PN Sequence for the carrier instead of a squarewave.

28. Examine the spectral composition of the original DSSS signal with the Signal Analyzer's *Units* control in both the *dB* and *Linear* positions.

Question 3
Why is the spectral composition of the DSSS signal much more complex when the carrier is a PN Sequence instead of a squarewave?

Ask the instructor to check your work before continuing.

Part C – Using the product detector to recover the message

29. Close the signal analyzer's VI.

30. Restart the scope's VI by clicking its *RUN* control once.

    Note: If you closed the scope's VI instead of suspending it, set it up per the instructions in Experiment 1 with the following changes:

    - *Timebase* control to *100µs/div* instead of *500µs/div*
    - Channel 1 *Scale* control to *2V/div* instead of *1V/div*
    - *Activate* the scope's Channel 1 input

31. Locate the Tuneable Low-pass Filter module on the DATEx SFP and set its soft *Gain* control to about a quarter of its travel.

32. Turn the Tuneable Low-pass Filter module's soft *Cut-off Frequency Adjust* control fully anti-clockwise.

33. Modify the set-up as shown in Figure 5 below.

    Note: Notice that the leads connect to the Multiplier module's *AC* inputs and not its *DC* inputs.

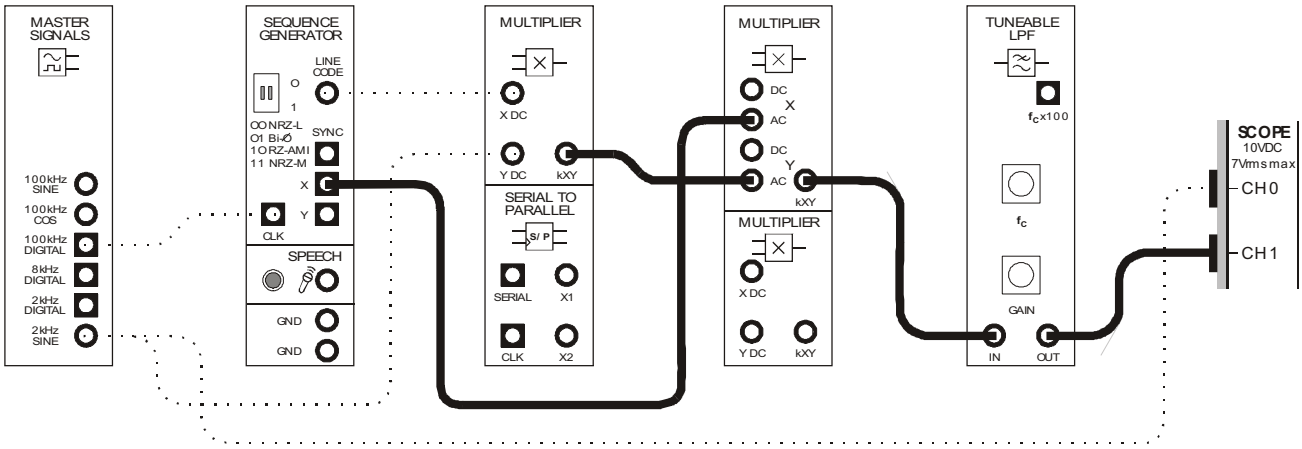


Figure 5

The additions to the set-up in Figure 5 can be represented by the block diagram in Figure 6 below. The Multiplier module and the Tuneable Low-pass Filter module implement a product detector which recovers the original message from the DSSS signal. To facilitate this, the PN sequence used for the modulator's carrier is "stolen" for the product detector's local carrier (though it's stolen from the module's *X* output but the bit pattern is the same).

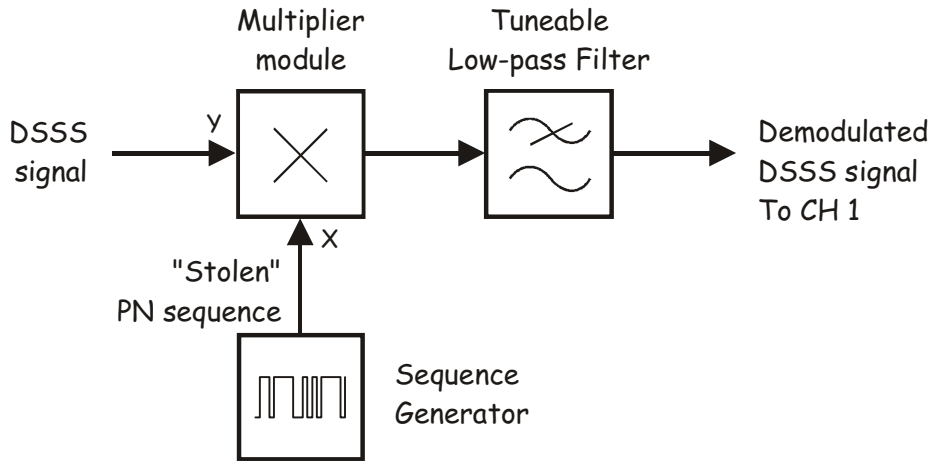


Figure 6

The entire set-up can be represented by the block diagram in Figure 7 below.

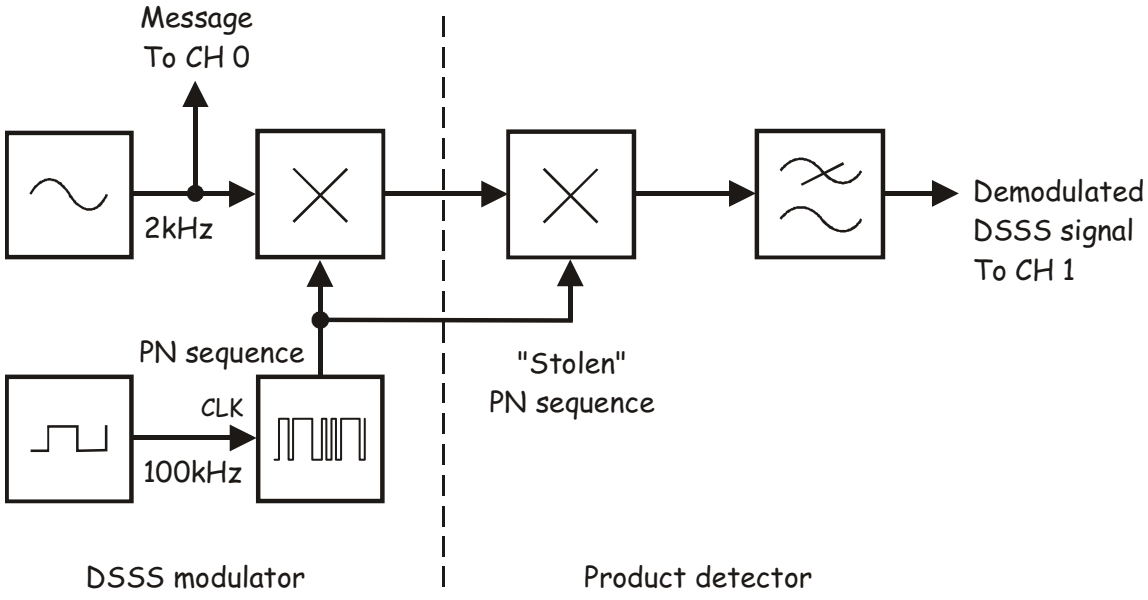


Figure 7

34. Slowly turn the Tuneable Low-pass Filter module's soft *Cut-off Frequency* control clockwise while watching the scope's display.

    Remember: You can use the keyboard's TAB and arrow keys for fine adjustments of DATEx controls.

35. Stop when the message signal has been recovered and is about in phase with the original.

36. Draw the demodulated DSSS signal to scale in the space that you left on the graph paper.

Ask the instructor to check your work before continuing.

Recall that the message can only be recovered by the product detector if an identical PN sequence to the DSSS modulator's carrier is used. The next part of the experiment demonstrates this.

37. Modify the set-up as shown in Figure 8 below to make the demodulator's local carrier a different PN sequence to the transmitter's carrier.

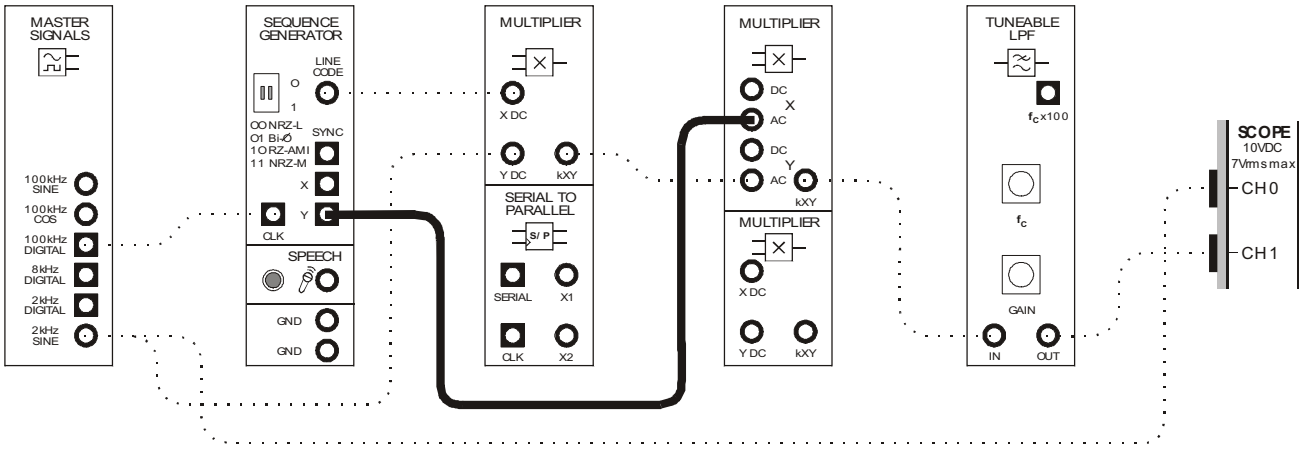


Figure 8

38. Compare the message with the product detector's new output.

Question 4
What does the signal out of the low-pass filter look like?

Question 5
Why does using the wrong PN sequence for the local carrier cause the product detector's output to look like this?

Ask the instructor to check your work before continuing.

Part D - DSSS and deliberate interference (jamming)
Interference occurs when an unwanted electrical signal gets added to the transmitted signal (typically in the channel) and changes it enough to change the recovered message. Electrical noise is a significant source of unintentional interference.

However, sometimes noise is deliberately added to the transmitted signal for the purpose of interfering or "jamming" it. The next part of the experiment models deliberate interference to show how spread spectrum signals are highly resistant to it.

39. Move the patch lead from the Sequence Generator's Y output back to its X output.

Note: The product detector should now be recovering the message again.

40. Adjust the function generator using its soft controls for an output with the following specifications:

    - Waveshape: Sine
    - Frequency: 50kHz
    - Amplitude: 4Vpp
    - DC Offset: 0V

41. Set the scope's *Trigger Source* control to the *CH 1* position.

42. Locate the Adder module on the DATEx SFP and turn its soft *g* control fully anti-clockwise.

43. Set the Adder module's soft *G* control to about the middle of its travel.

44. Modify the set-up as shown in Figure 9 below.

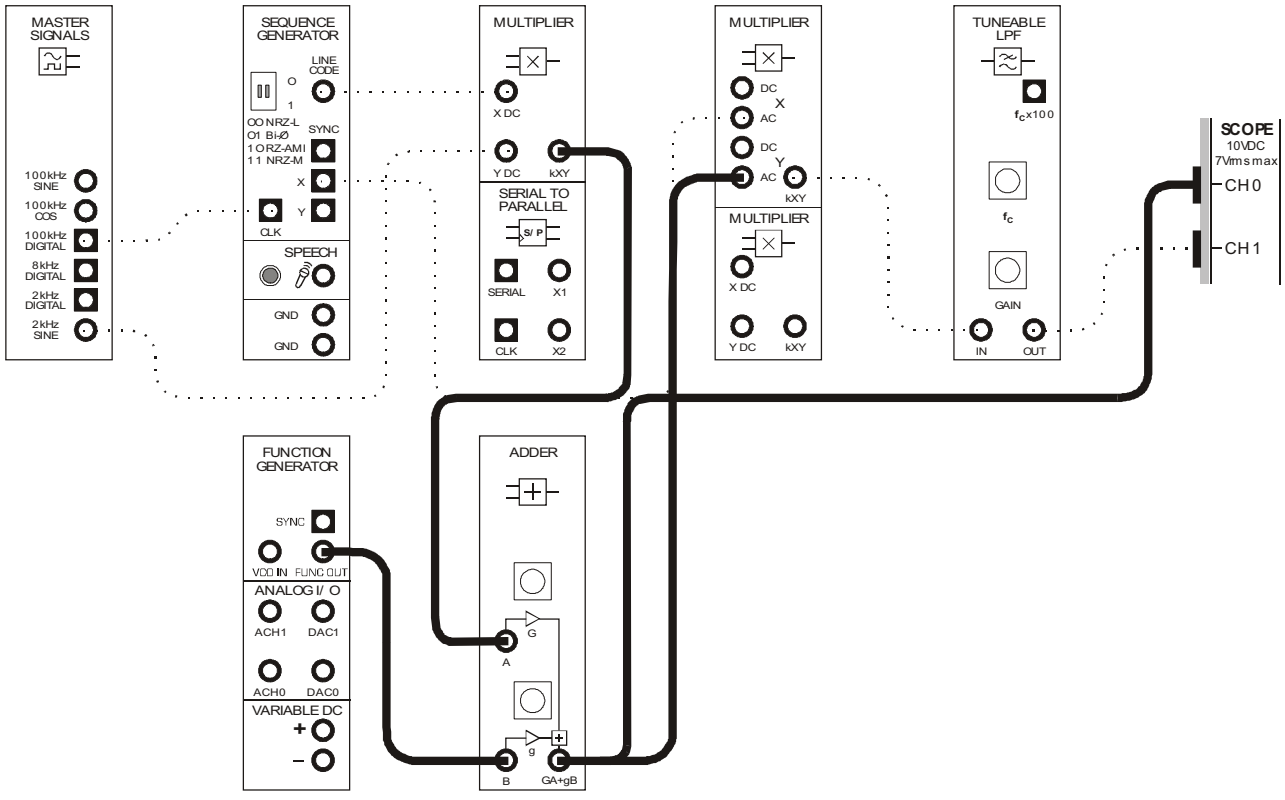


Figure 9

The set-up in Figure 9 can be represented by the block diagram in Figure 10 below. The function generator is used to generate a variable frequency jamming signal that is added to the DSSS signal in the channel using the Adder module.

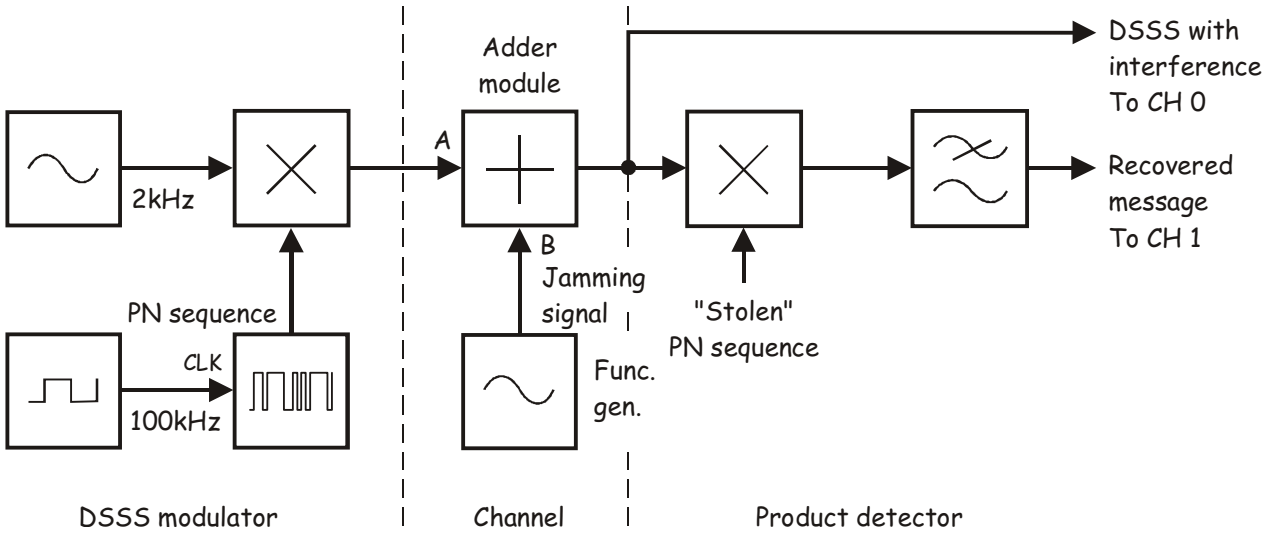


Figure 10

45. Add the jamming signal to the DSSS signal by slowly turning the Adder module's *g* control clockwise. Stop when it's at about half its travel.

46. As you increase the amplitude of the jamming signal note the effect it has on the DSSS signal and the recovered message.

47. Vary the jamming signal's frequency by varying the function generator's output frequency.

48. Note the effect this has on the DSSS signal and on the recovered message.

49. Increase the size of the jamming signal to maximum by turning the Adder module's *g* control fully clockwise.

50. Note the effect this has on the DSSS signal and on the recovered message.

Question 6

Why doesn't the jamming signal interfere with the recovery of the message?

Ask the instructor to check your work before continuing.

A more sophisticated approach to jamming involves automatically sweeping the jamming signal through a wide range of frequencies to increase the chances of upsetting the transmitted signal. The next part of the experiment let's you see how spread spectrum handles this.

51. Return the Adder module's *g* control to about the middle of its travel.

52. Adjust the function generator by setting its *Modulation Type* to *FM*.

53. Modify the set-up as shown in Figure 11 below.

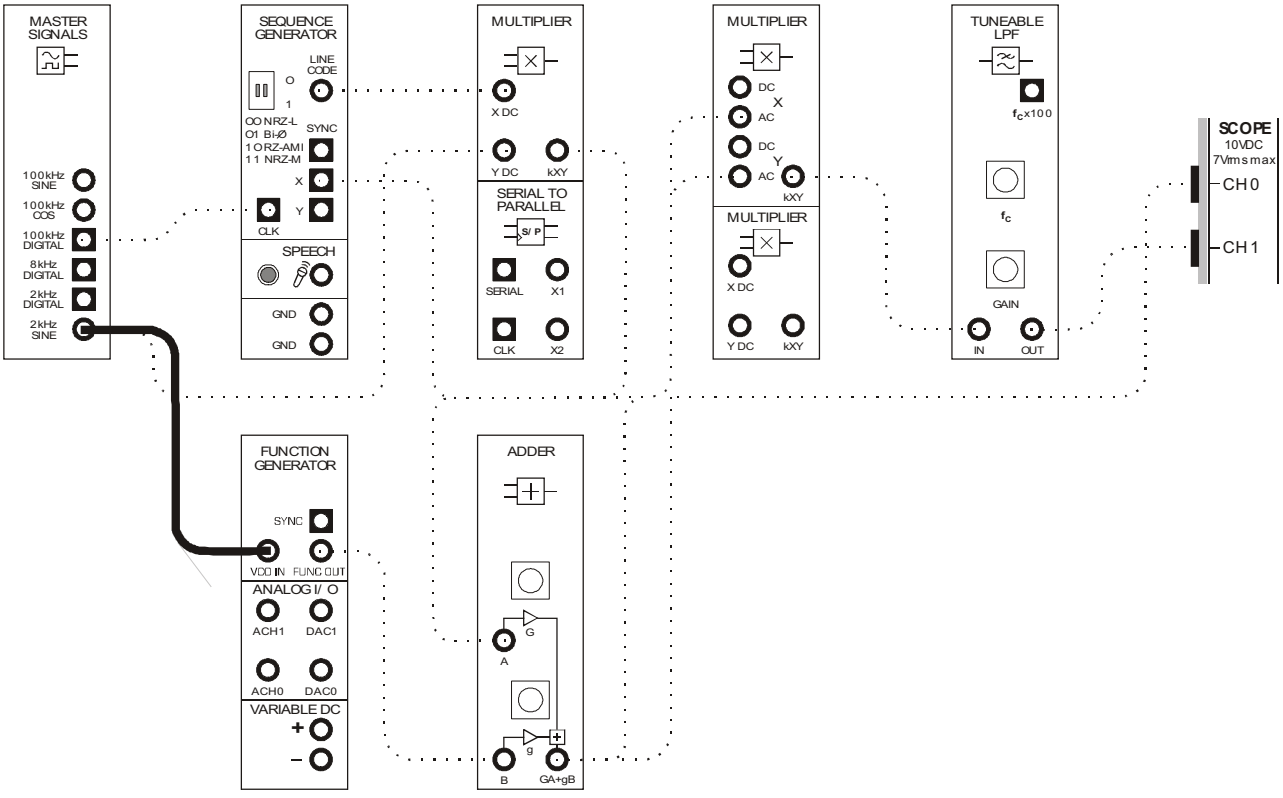


Figure 11

This modification forces the function generator's output to sweep continuously through a wide range of frequencies.

54. Note the effect this has on the DSSS signal and on the recovered message.

55. Increase the size of the jamming signal to maximum by turning the Adder module's *g* control fully clockwise.

56. Note the effect this has on the DSSS signal and on the recovered message.

Question 7
Why doesn't the sweeping jamming signal interfere with the recovery of the message?

Ask the instructor to check your work before continuing.

An even more sophisticated approach to jamming involves using many jamming signals at once (broadband jamming) to increase the chances of upsetting the transmitted signal. The next part of the experiment let's you see how spread spectrum handles this.

57. Return the Adder module's soft *g* control to about the middle of its travel.

58. Disconnect the lead to the function generator and modify the set-up as shown in Figure 12 below.

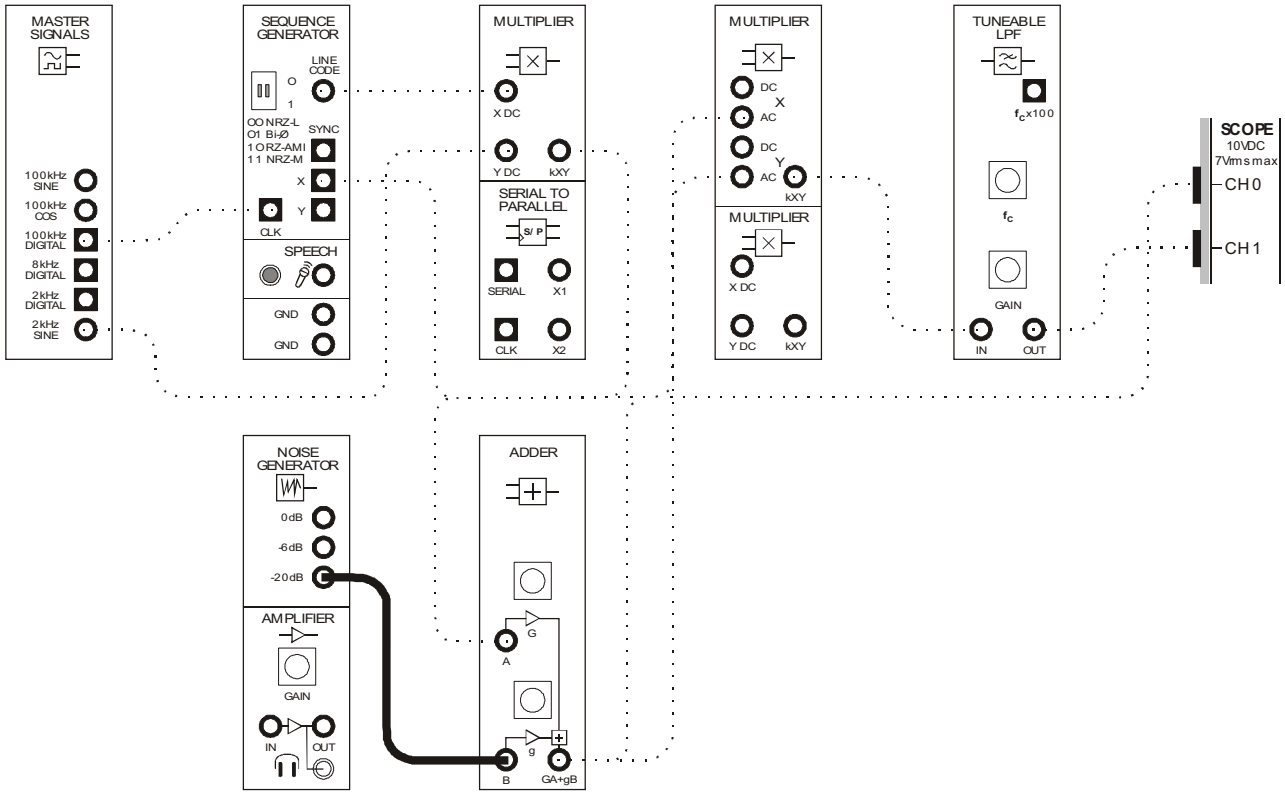


Figure 12

This modification uses the Noise Generator module to model a jamming signal that consists of thousands of frequencies.

59. Note the effect this has on the DSSS signal and on the recovered message.

60. Increase the strength of the broadband jamming signal by connecting the Adder module's B input to the Noise Generator module's *-6dB* output.

61. Note the effect this has on the DSSS signal and on the recovered message.

62. Increase the strength of the broadband jamming signal even more by connecting the Adder module's B input to the Noise Generator module's *0dB* output.

63. Note the effect this has on the DSSS signal and on the recovered message.

Question 8
Why doesn't this broadband jamming signal interfere with the recovery of the message?

Ask the instructor to check your work before finishing.

If time permits…

If the instructor allows, listen to how DSSS performs when being jammed by following these instructions. You'll need a set of stereo headphones for this activity.

1. Remove the jamming signal by disconnecting the Adder module's B input from the Noise Generator module's *0dB* output.

2. Connect the Tuneable Low-pass Filter module's output to the Amplifier module's input.

3. Locate the Amplifier module on the DATEx SFP and turn its soft *Gain* control fully anti-clockwise.

4. Without wearing the headphones, plug them into the Amplifier module's headphone socket.

5. Put the headphones on.

6. Adjust the Amplifier module's soft *Gain* control until the 2kHz tone is a comfortable sound level.

7. Investigate what happens when the wrong PN sequence is used to demodulate the DSSS signal (like you did in Part C) by moving the patch lead from the Sequence Generator's X output to its Y output.

8. Return the patch lead from the Sequence Generator's Y output back to its X output.

9. Investigate what happens when a single sinewave is used to jam the DSSS signal (like you did in Part D) by connecting the function generator's output to the Adder module's B input.

10. Investigate what happens when a broad-band signal is used to jam the DSSS signal (like you did in Part D) by connecting the Noise Generator module's *-20dB* output to the Adder module's B input.

11. Repeat the step above for higher levels of jamming/noise by connecting the Noise Generator module's *-6dB* output to the Adder module's B input then the *0dB* output.

Name: _____

Class: _____

14 - PN sequence spectra and noise generation

Experiment 14 – PN sequence spectra and noise generation

Preliminary discussion

Pseudo-noise sequences (or just PN sequences) are very useful signals in communications and telecommunications, especially for implementing modulation schemes such as DSSS and CDMA (among others). They can also be used to generate noise for experimental purposes when modelling real world communications systems. But what exactly is a PN sequence?

To understand the answer to this question, you must return to the spectral composition of pulse trains. Recall that pulse trains are made up of a theoretically infinite number of sinewaves – the fundamental and its harmonics. Recall also that the frequency and amplitude of a pulse train's sinusoidal components affects its frequency and mark-space ratio (or duty cycle). Despite this, the spectral composition of all pulse trains follows the pattern of the (truncated) *Sinc Function* shown in Figure 1 below.

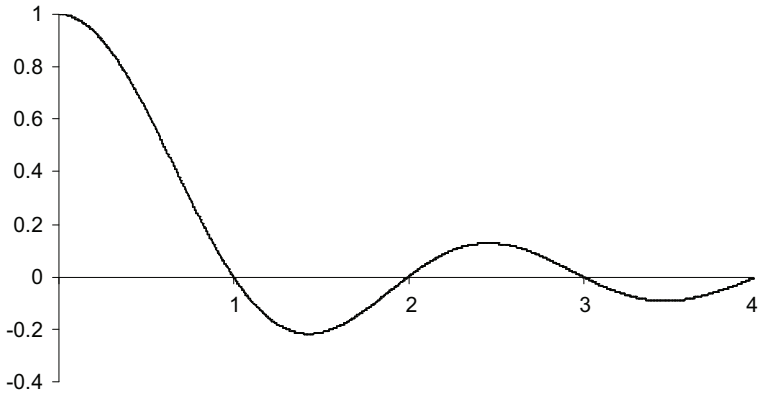


Figure 1

Figure 2 below illustrates this with an example of a 1kHz squarewave (a pulse train with a mark-space ratio of 1:1 or a duty cycle 50%). This is a spectrum that would be familiar to you.

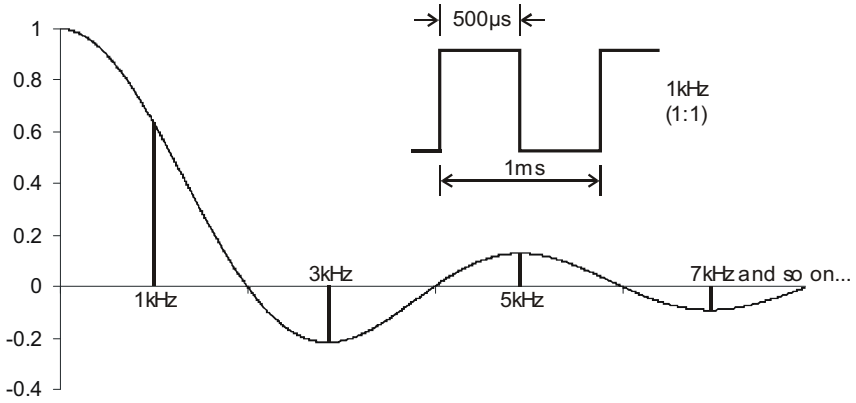


Figure 2

Figure 3 below shows the spectral composition of a 1kHz pulse train pulse having a mark-space ratio of 1:3 (or a duty cycle 25%). Notice that it too follows the pattern of the Sinc Function.

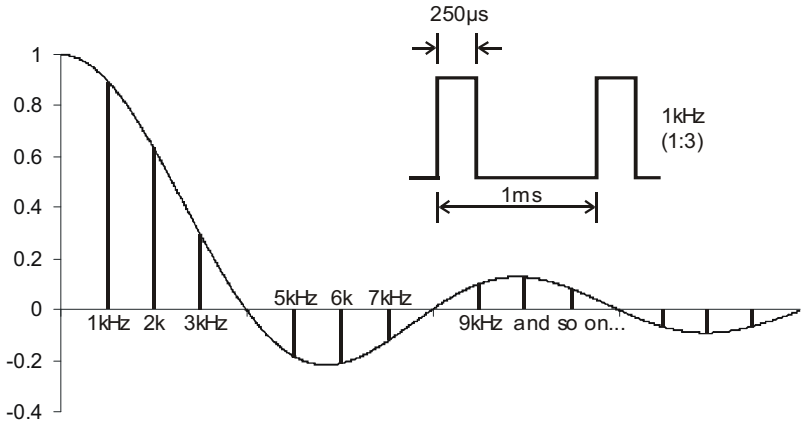


Figure 3

The examples in Figures 2 and 3 are instructive. Together, they show us that some harmonics of the pulse trains have an amplitude of zero (or are "nulled") and this is true of all pulse trains. Second, a comparison of Figures 2 and 3 shows us that, as the pulse train's mark-space ratio decreases, the number of significant harmonics that make it up increases. Or, put another way, as the mark-space ratio decreases, the number of harmonics that are present in each of the Sinc Function's lobes increases.

Now, suppose a sequence generator continuously outputs the sequential 4-bit binary number 1000 with each bit being 250µs wide (requiring a bit-clock of 4kHz). In the time domain, the resulting digital data signal is identical to the pulse train in Figure 3. This means that the sequence's spectral composition must be identical to the spectrum in Figure 3 also.

This fact has a couple of important implications. First, we can establish a general rule for determining the nulled harmonics in repeated sequential binary number sequences. They correspond with whole number multiples of the digital signal's bit-clock (that is, f_{bit}, $2f_{bit}$, $3f_{bit}$ and so on). In the case of our repeated sequential 4-bit binary number 1000 generated using a 4kHz bit-clock, the nulls occur at 4kHz, 8kHz, 12kHz and so on to infinity (theoretically).

Second, if the sequence generator's continuously repeated output is changed to the 5-bit binary number sequence 10000, the mark-space ratio of the resulting digital data signal decreases and so more harmonics are present between the nulls. Importantly though, if a 4kHz bit-clock is used to generate the 5-bit sequence, the nulls occur at the same frequencies as our example in Figure 3. So, with the nulls occurring at the same frequencies but with more harmonics between them, the spectral composition of the 5-bit sequence must be richer than that of its 4-bit counterpart. This gives us a second general rule. The greater the number of bits in a repeated sequence for a given bit-clock, the greater the sequence's spectral composition (though this doesn't apply to PN sequences with internally repeated sequences like 101010… and 11001100…).

Using the Sinc Function to analyse the spectral composition of several binary number sequences like 1000, 10000, 100000 and so on would quickly show that the number of harmonics in each lobe is the same number as the sequence's length (though the last one is nulled).

Finally, we can now return to the question of what is a pseudo-noise sequence. If the length of certain binary number sequences is long enough, their spectral composition becomes so dense that it can be used to model bandwidth limited white noise. That said, there would still be a repetitive element to the "noise signal" and so they're called pseudo (or "apparent") noise sequences.

The experiment

For this experiment you'll use the Emona DATEx to consider a 31-bit and 255-bit binary number sequence in the time domain. You'll then look at the data signals' spectra in the frequency domain to confirm their spectral composition. Finally, you'll use the sequences to generate electrical noise and compare their effectiveness.

It should take you about 50 minutes to complete this experiment.

Pre-requisites:

Experiments 1, 2 & 3 (Vol. 1): Intros to the NI ELVIS II, the Emona DATEx and SFP control

Equipment

- Personal computer with appropriate software installed
- NI ELVIS II plus USB cable and power pack
- Emona DATEx experimental add-in module
- Three BNC to 2mm banana-plug leads
- Assorted 2mm banana-plug patch leads

Procedure

Part A – Observations of PN sequences in the time domain

The next part of this experiment gets you to set up a 31-bit and a 255-bit binary number sequence and consider them in the time domain as preparation for looking at their spectra.

1. Ensure that the NI ELVIS II power switch at the back of the unit is off.

2. Carefully plug the Emona DATEx experimental add-in module into the NI ELVIS II.

3. Set the *Control Mode* switch on the DATEx module (top right corner) to *PC Control*.

4. Connect the NI ELVIS II to the PC using the USB cable.

   Note: This may already have been done for you.

5. Turn on the NI ELVIS II power switch at the rear of the unit then turn on its *Prototyping Board Power* switch at the top right corner near the power indicator.

6. Turn on the PC and let it boot-up.

7. Launch the NI ELVISmx software.

8. Connect the set-up shown in Figure 4 below.

   Note: Insert the black plugs of the oscilloscope leads into a ground (*GND*) socket.

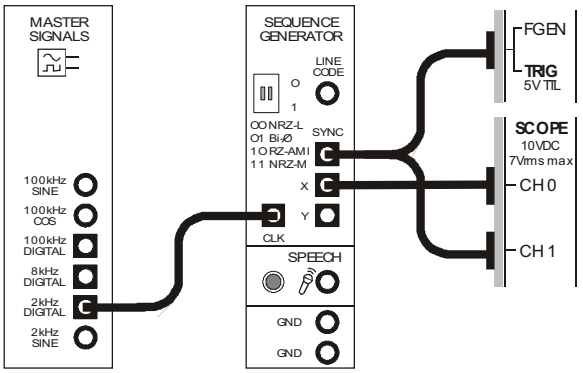


Figure 4

The set-up in Figure 4 can be represented by the block diagram in Figure 5 below. The Master Signals module's *2kHz DIGITAL* output is used to provide the Sequence Generator module's bit-clock. The Sequence Generator module's *X* output is a continuous 31-bit sequential binary number. The module's *SYNC* output is a pulse that corresponds with the sequence's first output bit on every repetition.

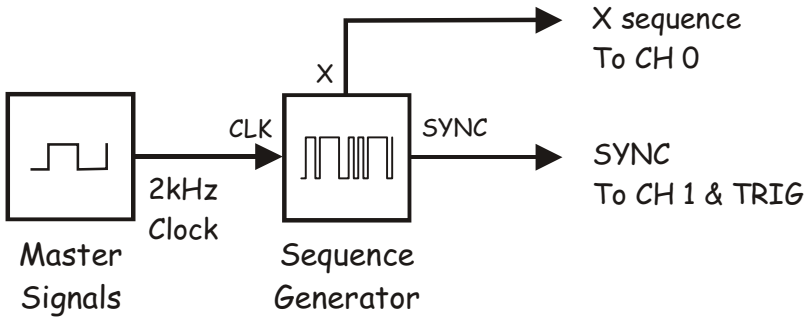


Figure 5

9. Launch and run the NI ELVIS II Oscilloscope VI.

10. Adjust the scope to view the Sequence Generator module's *X* output as a stable display. Essential scope settings include:

    - *Timebase* to *2ms/div*
    - *Trigger Type* to *Digital*
    - CH 1 *Vertical Position* to *-5V*

11. Activate the scope's Channel 1 input (by checking the Channel 1 *Enabled* box) to view both the Sequence Generator module's *X* and *SYNC* outputs.

Ask the instructor to check your work before continuing.

Question 1
Calculate the width of each bit in the Sequence Generator module's *X* sequence for the bit-clock used. **Note:** For accuracy here, you need to be aware that the Master Signals module's 2kHz outputs are actually 2.083kHz.

Question 2
Calculate the duration of the entire 31-bit sequence.

The next part of the experiment gets you to verify your answer to Question 2 using the scope's cursors.

12. Activate the scope's cursors by checking (that is, ticking) the scope's *Cursors On* box.

The NI ELVIS II Oscilloscope has two cursors (*C1* and *C2*) that default to the left most side of the display when the scope's VI is first launched. They're repositioned by "grabbing" their vertical lines with the mouse and moving the mouse left or right.

13. Use the mouse to grab and move the vertical line of cursor *C1*.

14. Repeat Step 13 for cursor *C2*.

The NI ELVIS II Oscilloscope cursors are actually measurement points. They measure the absolute instantaneous voltage of the signal on either Channel 0 or Channel 1 (the default is set to Channel 0 for both cursors). And, they measure the time difference between them. This information is displayed just below the signal display using brown text on the line labelled "Cursors:" and is highlighted in Figure 6 below.

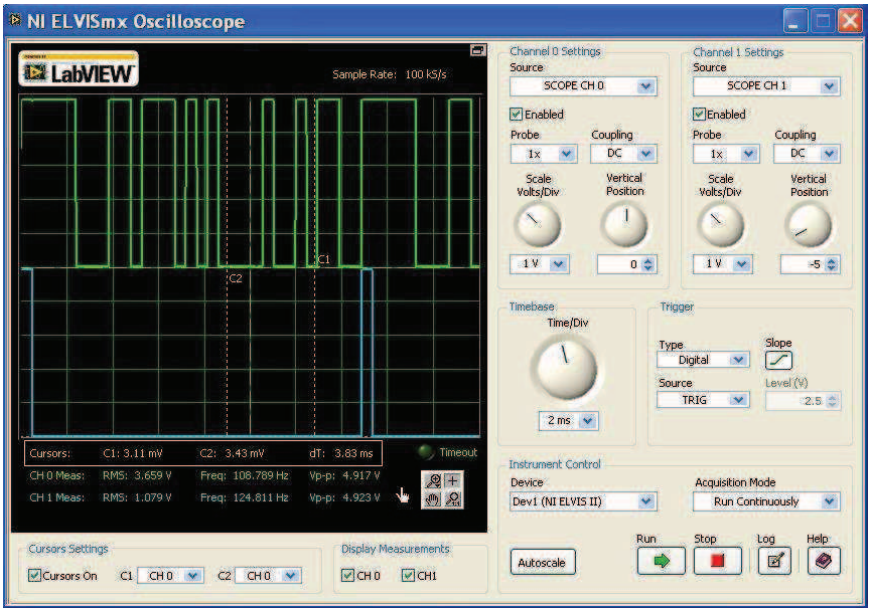


Figure 6

Notice that the absolute voltage of the CH 0 signal at *C1* in Figure 6 is 3.11mv and the absolute voltage of the CH 0 signal at *C2* is 3.43mV. Also notice that the time difference between the cursors is 3.83ms.

Now, to verify your answer to Question 2…

15. Move *C1* to the extreme left of the scope's display.

    Note: This aligns *C1* with the beginning of the sequence on the Sequence Generator module's *X* output.

16. Align *C2* with the next positive edge of the Sequence Generator module's *SYNC* signals.

    Note: This aligns *C2* with the beginning of the next sequence on the Sequence Generator module's *X* output.

17. Note the time difference between the cursors.

    Note: It should be the same as your answer to Question 2. If not, work out which one is wrong.

Ask the instructor to check your work before continuing.

18. Deactivate the scope's cursors.

19. Modify the set-up as shown in Figure 7 below.

Note: Remember that the dotted lines show leads already in place.

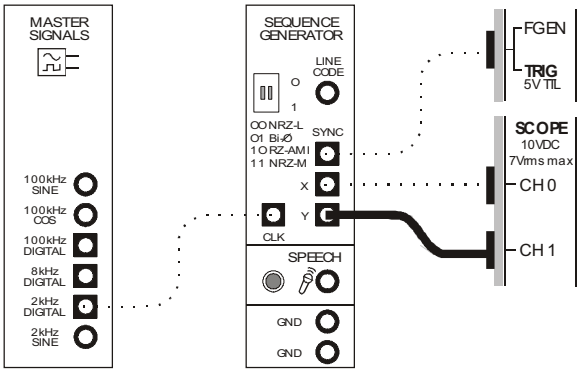


Figure 7

This set-up can be represented by the block diagram in Figure 8 below.

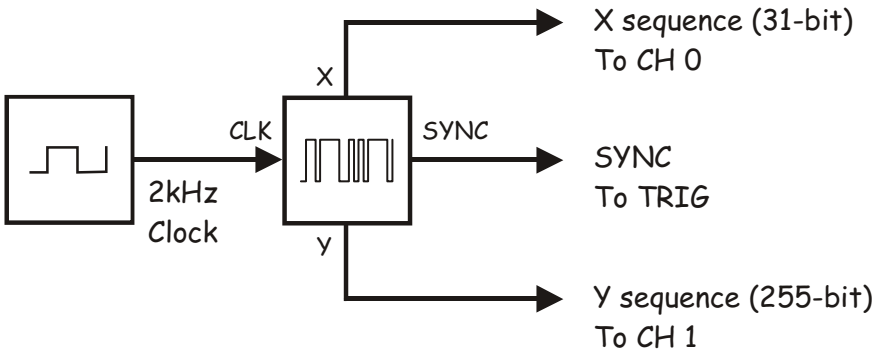


Figure 8

The scope will now be showing the Sequence Generator module's two sequences. However, you'll notice that the scope cannot trigger on the *Y* sequence. This is because the scope triggers on the first bit of the *X* sequence on every repetition using the Sequence Generator module's *SYNC* signal. As the *Y* sequence is 255 bits long, and as 255 ÷ 31 is not a whole number, every sweep of the scope's display starts at a different point in the *Y* sequence resulting in a different pattern for it on every sweep.

Ask the instructor to check your work before continuing.

Part B – Observations of PN sequences in the frequency domain

The next part of the experiment gets you to examine the spectral composition of the 31-bit and a 255-bit Sequence Generator module's *X* and *Y* outputs. But first, a little preparation is necessary.

Question 3

Calculate the frequency of the first four nulled harmonics in the Sequence Generator module's *X* sequence? **Tip:** If you're not sure how to calculate this, re-read the preliminary discussion.

Let's verify your answer using the NI ELVIS II Dynamic Signal Analyzer.

20. Suspend the scope's operation by clicking on its *Stop* control once.

    Note: The scope's display should freeze and its hardware has been deactivated. This is a necessary step as the scope and signal analyzer share hardware resources and so they cannot be operated simultaneously.

21. Minimise the scope's VI.

22. Launch and run the NI ELVIS II Dynamic Signal Analyzer VI.

Note: If the Dynamic Signal Analyzer VI has launched successfully, the instrument's window will be visible (see Figure 9).

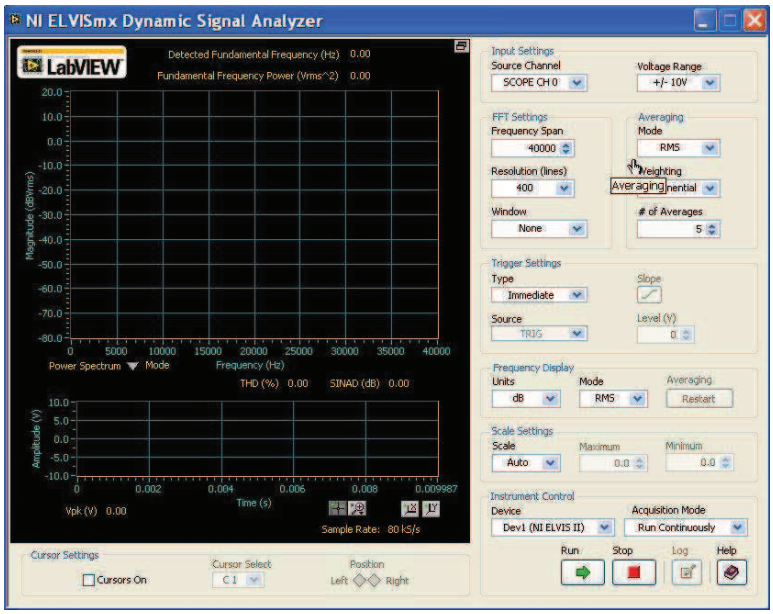


Figure 9

23. Adjust the signal analyzer's controls as follows:

Input Settings

- *Source Channel* to *SCOPE CH 0*
- Voltage Range to ±10V

FFT Settings

- *Frequency Span* to *40,000*
- *Resolution* to *400*
- *Window* to *7 Term B-Harris*

Averaging

- *Mode* to *RMS*
- *Weighting* to *Exponential*
- *# of Averages* to *3*

Trigger Settings

- *Type* to *Digital*

Frequency Display

- *Units* to *dB*
- *Mode* to *RMS*
- *Scale* to *Auto*
- *Cursors On* box unchecked (for now)

Once adjusted correctly, the signal analyzer's display should look like Figure 10 below.

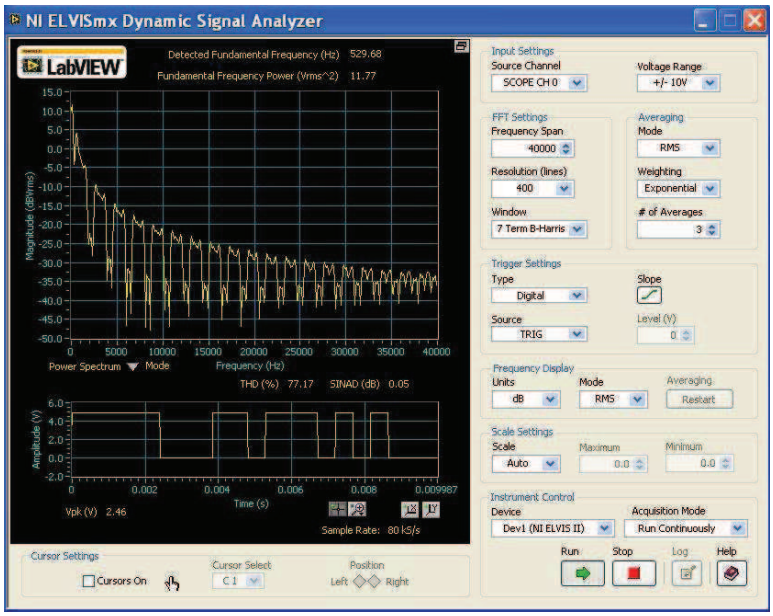


Figure 10

If you've not used the signal analyzer before, its display may need a little explaining here. There are actually two displays, a large one on top and a much smaller one underneath. The smaller one is a time domain representation of the input (in other words, the display is a scope).

The larger of the two displays is the frequency domain representation of the 31-bit sequence on the Sequence Generator module's *X* output. The humps or lobes represent groups of the sinewaves and, as you can see, they follow the general pattern of the Sinc Function.

Question 4

Why are there many more lobes in the *X* sequence's spectrum than suggested in Figures 1, 2 and 3 of the preliminary discussion?

If you have used the signal analzyer before and have also used its cursors, go directly to Step 31 on the next page.

24. Activate the signal analyzer's cursors by checking (that is, ticking) *Cursors On* box.

    Note: When you do, green horizontal and vertical lines should appear on the signal analyzer's frequency domain display.

The NI ELVIS II Dynamic Signal Analyzer has two cursors *C1* and *C2* that default to the left most side of the display when the signal analyzer's VI is launched. Like the scope's cursors, they're repositioned by "grabbing" their vertical lines with the mouse and moving the mouse left or right.

25. Use the mouse to grab and move the vertical line of cursor *C1*.

    Note: As you do, notice that cursor *C1* moves along the signal analyzer's trace and that the vertical and horizontal lines move so that they always intersect at *C1*.

26. Repeat Step 25 for cursor *C2*.

    Note: Fine control over the cursors' position is obtained by using the cursor's *Position* control in the *Cursor Settings area (below the display*).

The NI ELVIS II Dynamic Signal Analyzer includes a tool that measures the difference in magnitude and frequency between the two cursors. This information is displayed in green between the frequency and time domain displays.

27. Move the cursors while watching the measurement readout to observe the effect.

28. Position the cursors so that they're on top of each other and note the measurement.

    Note: When you do, the measurement of difference in magnitude and frequency should both be zero.

Usefully, when one of the cursors is moved to the extreme left of the display, its position on the X-axis is zero. This means that the cursor is sitting on 0Hz. It also means that the measurement readout gives an absolute value of frequency for the other cursor. This makes sense when you think about it because the readout gives the difference in frequency between the two cursors but one of them is zero.

29. Move *C2* to the extreme left of the display.

30. Move *C1* to any point on any of the spectrum's lobes.

    Note: The readout will now be showing you the frequency of a sinewave at that point in the lobe.

31. Use the signal analyzer's *C1* cursor to check that the frequencies you listed in your answer to Question 3 are indeed nulled as you predicted.

Ask the instructor to check your work before continuing.

Question 5
Theoretically, how many harmonics make up each lobe in the spectrum of the Sequence Generator module's *X* output?

Question 6
Why can't you see each of the harmonics individually?

32. To verify the answer to your questions, set the signal analyzer's *Frequency Span* to 2,500Hz.

    Note: Once the signal analyzer's display has updated, you should see a number of discrete sinewaves representing all of the harmonics in the signal's first lobe.

33. Use the signal analyzer's *C1* cursor to locate the null at 2.083kHz.

34. Count the number of significant harmonics in the signal's first lobe.

    Note: Your count should match your answer to Question 5. If not, investigate which one is wrong.

Ask the instructor to check your work before continuing.

Now let's consider a longer sequence. But first, a little more preparation.

Question 7
Calculate the frequency of the first four nulled harmonics in the Sequence Generator module's *Y* sequence?

35. Return the signal analyzer's *Frequency Span* to 40,00Hz.

36. Set the signal analyzer's *Source Channel* to *SCOPE CH 1*.

37. Use the signal analyzer's *C1* cursor to verify your answer to Question 7.

Question 8
Theoretically, how many harmonics make up each lobe in the spectrum of the Sequence Generator module's *Y* output?

38. See if you can verify your answer to Question 8 by setting the signal analyzer's *Frequency Span* to 2,500Hz to examine the spectral composition of the signal's first lobe.

Question 9
Why can't you accurately count the harmonics in the signal's first lobe?

Question 10
Which of the Sequence Generator module's two sequences has the greater harmonic content?

39. As there are 255 harmonics in the each of the signal's lobes, they should be about 8.16Hz apart (2.083kHz ÷ 255). Reduce the signal analyzer's *Frequency Span* to see if you can measure this separation between them using the cursors.

    Note: You may need the instructor's help with adjusting some of the signal analyzer's other controls.

Ask the instructor to check your work before continuing.

Part C – Using PN sequences to generate noise

Generating the theoretical proposition of white noise is impossible. To explain, an infinite number of sinewaves (with or without equal power density) would require an infinite amount of power! However, as you have just seen, long PN sequences are rich in harmonics. Moreover, although the spectrum of PN sequences have lobes of changing amplitude, small portions of its spectrum are relatively flat (a point you may have noticed at Step 38). That being the case, it's possible to isolate a small portion of a PN sequence's spectrum using a filter to model band-limited white noise. The next part of this experiment demonstrates this.

40. Suspend the signal analyzer's VI.

41. Completely dismantle the current set-up.

42. Launch and run the NI ELVIS II Function Generator VI.

43. Adjust the function generator for a 150kHz output.

    Note: It's not necessary to adjust any other controls as the function generator's *SYNC* output will be used and this is a digital signal.

44. Connect the set-up shown in Figure 11 below.

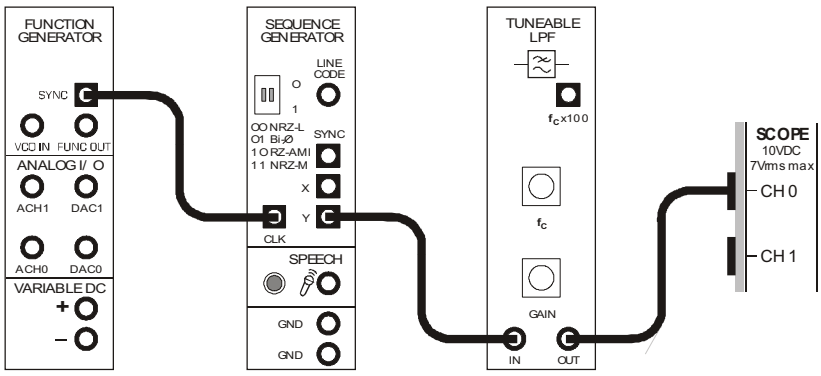


Figure 11

The set-up in Figure 11 can be represented by the block diagram in Figure 12 below. A quick mathematical analysis tells us that, with a 150kHz bit-clock, the spectral composition of the Sequence Generator module's *Y* output includes 255 sinewaves per lobe and so they are separated by 588Hz.

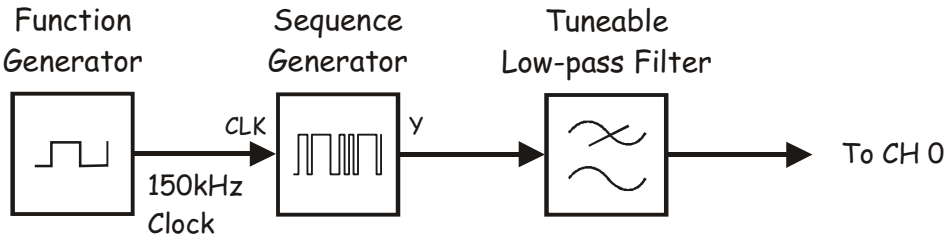


Figure 12

45. Launch the DATEx soft front-panel (SFP) and check that you have soft control over the DATEx board.

46. Locate the Tuneable Low-pass Filter module on the DATEx SFP and set its soft *Gain* to about the middle of its travel.

47. Turn the Tuneable Low-pass Filter module's soft *Cut-off Frequency Adjust* control fully clockwise.

    Note: This sets the Tuneable Low-pass Filter module's cut-off frequency to 15kHz.

48. Restart the scope's VI.

49. Adjust the scope to view the Tuneable Low-pass Filter module's output. Essential scope settings include:

    - *Timebase* to *1ms/div*
    - *Trigger Type* to *Immediate*
    - CH 1 deactivated

50. Observe the signal.

Question 11
What does the signal on the Tuneable Low-pass Filter module's output look like?

Ask the instructor to check your work before continuing.

The signal on the Tuneable Low-pass Filter module's output isn't "white" noise because it is bandwidth limited. Nor is the signal truly "noise". This can be demonstrated using the scope. True noise is non-repetitive. However, the signal on the Tuneable Low-pass Filter module's output repeats itself every 1.7ms. [This figure is calculated using the bit-clock's period (1÷150,000Hz) and multiplying it by the PN sequence's number of bits (255).] The repetitive nature of the "noise" you have modelled can be observed using the scope.

51. Suspend the scope's VI to stop the signal from jumping around on the screen.

52. Look closely at the signal - You should see it repeat itself about 5 times.

53. Activate the scope's cursors and use them to measure the signal's period.

    Note: You should find it is close to the figure quoted above.

Question 12
Given the signal on the Tuneable Low-pass Filter module's output is repetitive, what's a better name for it than "noise"?

Ask the instructor to check your work before continuing.

Question 13
How many sinewaves fall inside the Tuneable Low-pass filter's 15kHz pass-band?

---

54. Restart the signal analyzer's VI and make the following adjustments:

    - *Source Channel* to *SCOPE CH 0*
    - *Frequency Span* to *20,000Hz*
    - *Trigger Type* to *Immediate*

55. Use the signal analyzer's *C1* cursor to indicate on the screen the Tuneable Low-pass Filter's cut-off frequency.

56. Count the number of significant harmonics between 0Hz and the filter's cut-off frequency.

    Note: Your count should match your answer to 13. If not, work out which one is wrong.

57. Suspend the signal analyzer's VI.

58. Restart the scope's VI.

59. Modify the set-up as shown in Figure 13 below.

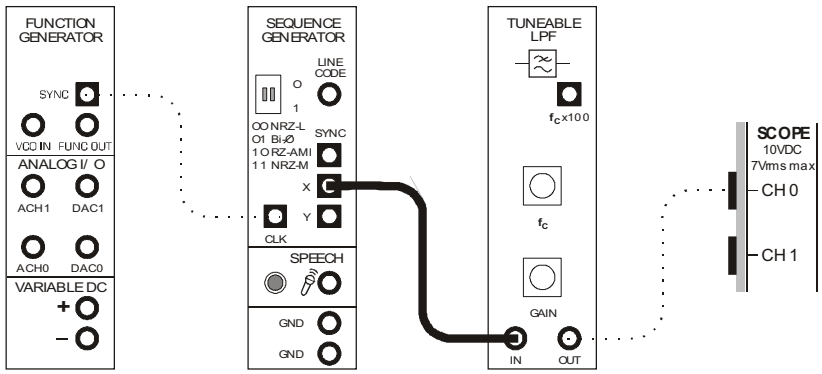


Figure 13

60. Observe the Tuneable Low-pass filter module's new output signal.

Question 14
Why doesn't the signal look like electrical noise any more?

61. To verify your answer to Question 14, suspend the scope's VI.

62. Restart the signal analyzer's VI.

63. Count the number of significant harmonics between 0Hz and the filter's cut-off frequency.

Ask the instructor to check your work before finishing.

Why are the DATEx Sequence Generator module's outputs 31 and 255 bits long?
You may be wondering why the length of the sequences for the DATEx Sequence Generator module's outputs are 31 and 255 bits long. To explain, shift register circuits known as *linear feedback shift registers* have been used to generate them. These circuits are similar to ring counters in that they recycle data through the shift register. However, they also include feedback via exclusive OR gates at key points to change the data word as it cycles through. That said, this doesn't generate purely random numbers as the data pattern must repeat. However, certain feedback connections create particularly long sequences and these are known as *maximal length sequences* (MLS).

Maximal length sequences have several interesting properties:

i) They have almost equal 1s and 0s…(actually 1 less 0 than 1s)

ii) They have equal number of "runs" of 1s and 0s (you can readily see this using the Sequence Generator module's 31-bit sequence

iii) They create a spectrum with no missing harmonics

iv) They never repeat within themselves (unlike the sequences mentioned at bottom of page 14-3)

These properties make maximal length sequences ideal for PN sequences used for applications like encryption, encoding etc. Naturally then, linear feedback shift registers generating maximal length sequences are used for the Emona DATEx Sequence Generator module.

For the *X* output, a 5-bit shift register is used making it 31 bits long (that is, 2^5-1) and for the *Y* output an 8-bit bit shifter register is used giving 255 bits (that is, 2^8-1).