Amateur-radio Applications of the Fast Fourier Transform

1. Introduction

The Fast Fourier Transform (FFT) is a very efficient numerical algorithm to compute a discrete Fourier transform. The FFT algorithm has found many different applications and it is not limited only to digital computers and digital signal processing. For example, a completely analog implementation of the FFT algorithm can be made with a number of 3dB directional couplers and delay lines to feed an antenna array.

Obviously the FFT algorithm is widely used in digital signal processing too. In this article a particular DSP application of the FFT algorithm is described and discussed in detail: FFT spectrum analysis. Since the FFT algorithm can be implemented on almost any digital computer and a FFT spectrum analyzer only requires a little additional hardware, FFT spectrum analysis is very convenient for many practical applications.

In order to understand the operation of a FFT spectrum analyzer and its advantages and drawbacks when compared with other spectrum analysis techniques, a short description of the various Fourier transforms and the operation of the FFT algorithm is included. Building a practically working spectrum analyzer around the bare FFT algorithm is also discussed. No mathematical proofs are given just to keep the discussion as simple as possible. The interested reader can find the former in almost any book describing digital signal processing.

Although FFT spectrum analysis is presently limited to the audio frequency range, at least for amateur resources, it has many amateur-radio applications ranging from weak signal detection to modulation analysis. A few typical amateur applications are discussed later in this article, including spectrum function plots and intensity spectrograms obtained from real signals using a FFT spectrum analyzer.

Finally, a FFT spectrum-analyzer program for the DSP computer described in UKW-Berichte/VHF-Communications is presented, including a short description of its commands and performance limits imposed by the hardware. All the practical examples shown in this article were obtained with this software: most figures are just hard-copy prints of the computer screen on a laser printer.

2. The Fourier Transform and the Discrete Fourier Transform

The Fourier transform is a mathematical operation that computes a new function F(w) from an original function f(t), as shown on Fig. 2.1. Both functions have real arguments t and w while their values f(t) and F(w) are in general complex numbers. F(w) is called the Fourier transform of a given

function f(t). The Fourier transform has some interesting properties including a relatively simple inverse operation: computing f(t) back from a given F(w) is very similar to the Fourier transform itself.

The Fourier transform is frequently used in physics to compute the frequency spectrum of a time-dependent physical quantity. In a physical problem t represents time and f(t) represents a physical quantity (force, displacement, pressure, electrical current or voltage) as a function of time. In all physical problems f(t) is a real function. The new variable w represents the frequency and the function F(w) is the frequency spectrum of the physical quantity observed. The frequency spectrum is a complex function of a real variable w. The absolute value of F(w) represents the magnitude of a given spectral component and the argument of F(w) represents the phase relative to the (chosen) time origin. Since f(t) is always a real function one does not really need to compute F(w) for negative frequencies: F(-w) is simply the complex-conjugate of F(w) for a real f(t).

While the Fourier transform is a powerful analytical tool to perform theoretical computations it has at least two constraints which could never be fulfilled in practice: infinite bandwidth and infinite frequency resolution (which implies an infinite observation time), regardless of the method, analog or digital, used to perform the Fourier transform.

A real-world, finite-bandwidth signal can be sampled without loosing any information provided that the sampling frequency is high enough, at least twice the signal bandwidth. The Fourier transform of a sampled signal is shown on Fig. 2.2. The integral is replaced with a sum, the bounds of the sum are however still infinite. An important difference should be noticed in plot of F(w): the spectrum of a sampled signal is a periodic function and its period is inversely proportional to the sampling period. It is therefore sufficient to compute (either in an analog or in a digital way) just one period of F(w), usually between -PI/deltat and +PI/deltat.

The (original) Fourier transform integral (or sum) has infinite bounds: the integration (or summation) should be performed from minus infinity to plus infinity. Of course no real-world signal will ever last that long! It is therefore completely sufficient to compute the integral or the sum only over the time interval when the signal exists. Integrating over a finite amount of time limits the frequency resolution, which is inversely proportional to the integration time. This implies that the frequency spectrum could also be represented by discrete samples in place of a continuous function.

The procedure that computes a finite number of spectral lines from a finite number of signal samples is called the Discrete Fourier Transform (DFT) and is shown on Fig. 2.3. To properly describe the signal spectrum one needs at least the same number of frequency samples as there are input signal samples. Of course, to obtain meaningful results, the frequency interval (spectral line spacing) has to be chosen in a close relationship to the time interval (signal sampling step). Similar constraints also apply to a completely analog Fourier

transform (analog spectrum analyzer).

In order to simplify computations, both time and frequency units are usually normalized to 1 as shown on Fig. 2.4. Time t now only takes the values of 0, 1, 2, ... (N-1) and frequency w also takes the values of 0, 1, 2, ... (N-1), if the discrete Fourier transform produces N spectral lines from N signal samples. As a result of this normalization the constant 2*PI/N appears in the complex exponent function. This constant is chosen such that the resulting spectral lines cover exactly one period of the periodic signal spectrum.

The DFT can be computed on any general-purpose computer. Its operation is similar to N bandpass FIR filters, each filter having N stages and tuned to its own frequency. The DFT is not computationally efficient, since the number of computations required increases with N^2. If the DFT is computed on a real signal (real f(t)) with N signal samples, then the result only includes N/2 spectral lines ranging from zero to half the sampling frequency. The remaining N/2 spectral lines are simply complex conjugates that can be computed in a much simpler way once the first N/2 spectral lines are known, reducing the total number of computations required to about N^2/2.

3. The Fast Fourier Transform (FFT) algorithm

The Discrete Fourier Transform supplies a very useful result, but unfortunately requires a very large number of computations on a digital computer or a very large number of components if performed by an analog circuit. A more efficient algorithm that provides exactly the same result as DFT with a considerably smaller effort required is called the Fast Fourier Transform. As an example, to compute a 1024 data point DFT the FFT algorithm requires about 200 times less computations (or analog components) than a straightforward DFT. Further, the number of computations required in the FFT algorithm is only proportional to N/2*log2(N). The FFT algorithm can therefore be performed on a very large number of data samples without significantly increasing the number of computations per sample.

The basic building block of the FFT algorithm is called a "butterfly" operation. A "butterfly" operation consists of a phase-shift operation and a sum/difference operation. It operates on two input variables and produces two results. A single "butterfly" operation can already compute a 2-point FFT, as shown on Fig. 3.1. In the case of a 2-point FFT, the two input variables are the input data to the FFT algorithm and the two results are already the result of the algorithm. The phase shift is equal to PI in this case.

Fig. 3.2. shows how a 4-point FFT works. A 4-point FFT is computed in two stages. Each stage includes two "butterfly" operations. In the first stage, all phase shifts are equal to PI. In the second stage, the phase shifts are PI/2 and PI. Note that each output of the first stage "butterflies" is fed to exactly one input of the second stage "butterflies". Considering the periodicity of the complex exponent function

the four outputs correspond exactly to the result obtained with a straightforward DFT, the latter however requires 16 phase shifts (8 if one does not consider zero phase shifts) compared to the 4 phase shifts of the FFT algorithm.

Similarly, a 8-point FFT is computed in three stages as shown on Fig. 3.3. Each stage includes 4 "butterfly" operations. Again, in the first stage all phase shifts are equal to PI. In the second stage the phase shifts are PI/2 and PI. In the third stage the phase shifts are PI/4, PI/2, 3*PI/4 and PI. The algorithm block diagram follows a regular pattern too, suggesting that a FFT algorithm working on an arbitrary large number of samples could be designed in a similar way. In the case of an 8-point FFT only 12 "butterflies" are required compared to the 64 shift/add operations of a straightforward DFT algorithm to get the same result.

To design FFT algorithms operating on an even larger number of data samples one should therefore investigate the possibility of combining several FFTs computed on a smaller number of samples. Fig. 3.4. shows how to combine two N-point transforms (not necessarily computed using the FFT algorithm) into one single transform on 2N (twice the number of) points. In addition to the two N-point transforms, N "butterflies" are required. These "butterflies" require N different phase shifts ranging from PI/N, 2*PI/N, 3*PI/N ... (N-1)*PI/N, PI in steps of PI/N.

The principle shown on Fig. 3.4. is in fact used to design a FFT algorithm operating on any data length N that is a power of 2. A FFT algorithm can thus operate on 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024... data points. The 4-point FFT can be derived from the 2-point FFT, the 8-point FFT can be derived from the 4-point FFT, the 16-point FFT can be derived from the 8-point FFT etc... Each doubling of the data points only requires an additional stage so the total number of stages is equal to log2(N). Each stage requires N/2 "butterflies". The total number of "butterfly" operations is therefore equal to N/2*log2(N).

A quick look at Fig. 3.3. shows that the results do not appear in any reasonable order at the output of the FFT algorithm: 1, 5, 3, 7, 2, 6, 4, 0. Considering the construction principle shown on Fig. 3.4. a very simple rule to find the desired output can be found. This rule is called bit-reversed addressing and it is shown on Fig. 3.5. for the 8-point FFT example. To find an output number one has to take the corresponding input number, write this number down in binary format, reverse the order of bits, convert the number back to decimal and add 1. This rule can be easily implemented in digital hardware (dedicated DSP microprocessors), where data is stored in consecutive memory locations.

To implement the FFT algorithm a number of "butterfly" operations have to be performed. An analog implementation called the "Butler matrix" uses delay lines to obtain phase shifts and "rat-race" hybrids or 3dB directional couplers for the add/subtract operations. The inputs of the FFT circuit are connected to the elements of a linear antenna array and the outputs of the FFT circuit to receivers and/or transmitters. Since the required phases and magnitudes of the signals

feeding an antenna array are the Fourier transform of the desired radiation pattern, the outputs of the FFT circuit correspond directly to beams in the various directions.

Implementing the FFT algorithm on a digital computer most operations are performed with complex numbers. Although phase shifts are easier to perform if complex numbers are held in a magnitude/phase format, sums and differences require a real/imaginary-component number format. Since conversions from one number format to another are very time consuming all computations are usually done in the real/imaginary-component format. In this case a phase shift operation requires four real multiplications with coefficients from a precomputed table of phase shifts and two additions. Each summation of two complex numbers requires two real summations and each complex difference requires two real differences.

The phase-shift coefficients are precomputed and stored in memory, since only N different coefficients are required in all stages of a N-point FFT. The same coefficients can be used in the following FFT, if a number of FFTs have to be computed on changing data. Finally, the phase-coefficient table includes sines and cosines which are much more time-consuming to compute than the multiplications and additions required for the FFT itself.

If a FFT is computed on real data, then only half of the outputs contain interesting data, the other half are just complex conjugates. To use the algorithm more efficiently another set of input data can be fitted into the imaginary part of the input variables. After the FFT algorithm is performed, the two results can be separated by simple additions and subtractions using the symmetry laws of the Fourier transform. If desired, the two results can be further combined into a single, double length FFT.

The inverse DFT can easily be performed using the FFT algorithm in the reverse direction. The number of mathematical operations is identical except for an additional division by N for each data point to obtain the original magnitude back.

There are even more efficient algorithms to compute a DFT or its inverse. All of them are however based on the FFT principle described above and require a little more programming efforts to further reduce the number of computations required.

4. Spectrum analysis using the FFT algorithm

One of the most obvious applications of the FFT algorithm is a FFT spectrum analyzer. The FFT algorithm itself is performed on a digital computer, usually a DSP microprocessor. The input signal is provided in a digital format from an A/D converter. The microprocessor itself can display the result in a variety of formats.

However, to build a digital FFT spectrum analyzer some additional functions are required. A block diagram of a DSP microprocessor-based spectrum analyzer is shown on Fig. 4.1. The analog input signal is first sent to an analog low-pass (band-pass) filter to prevent aliasing like in any DSP

application. The analog filter is followed by a sample-and-hold and an A/D converter. The A/D converter feeds a buffer memory, since the FFT algorithm operates on blocks of data. Before the FFT the signal samples may be weighted optionally. The complex spectrum provided by the FFT algorithm is used to compute the magnitudes of the spectral components, the phase information is discarded. Averaging is used to improve the signal-to-noise ratio in some measurements. An optional linear-to-logarithmic conversion is standard for all spectrum analyzers. Finally, the result can be displayed in different formats and a conventional computer printer can be used to obtain a hard-copy.

The performance of a DSP FFT spectrum analyzer is mainly limited by the performance of the A/D converter used. The conversion speed of the A/D converter defines the maximum bandwidth and the resolution of the A/D converter defines the available dynamic range. After the A/D converter the digitized input signal can be conveniently stored in memory if the microprocessor is unable to process the data in real time. Also all suitable microprocessors offer a computational accuracy of at least 16-24bits allowing a much wider dynamic range than any A/D converter.

Even if the DSP microprocessor is fast enough to process all of the data in real time, the input data stream has to be sent to a buffer memory first, since the FFT algorithm operates on blocks of data, not on single samples. Without weighting FFTs are usually performed on contiguous, but non-overlapping blocks of data to obtain almost all of the spectral information contained in the input signal.

In the case of input signal weighting, the contribution of the samples at the beginning or at the end of a block is very limited. To use all of the information contained in the input signal overlapped FFTs have to be performed. Both cases are shown on Fig. 4.2. In the second case the overlap is set to about 50% for a raised-cosine weighting function.

If the FFT algorithm is performed on a raw, unweighted block of data, the corresponding spectrum is distorted. Since the FFT does not consider any data samples before the block nor any data samples after the processed block of data, the actual input signal to the FFT algorithm corresponds to the real input signal modulated with a rectangular impulse of the length of the data block. A rectangular impulse with steep leading and trailing edges has a very wide frequency spectrum of the form sin(X)/X. The resulting output will be the real signal spectrum convoluted with the sin(X)/X function.

Although the real signal spectrum can not be obtained since it requires an infinite amount of time, a much more accurate spectrum can be obtained by weighting the signal samples as shown on Fig. 4.3. Weighting means multiplying each signal sample with a constant whose value depends on the position of the sample inside the input data block. The practical effect of weighting is to replace the abrupt ON/OFF transitions with smooth transitions at the beginning and at the end of the data block. Weighting functions are selected to minimize the distortion of the signal spectrum. Raised-cosine and Gaussian functions are usual choices, since they have a very narrow own spectrum with low side lobes.

The effects of no weighting versus weighting are shown on Fig. 4.4. The two plots show the spectrum of the same signal obtained in two different ways. A linear 512-point frequency scale is used on the horizontal axis and a logarithmic amplitude scale (15bits or 90dB/full scale) is used on the vertical axis. All the parameters, including the input signal, are the same for both plots except for weighting: the plot above was obtained without any weighting while the plot below was obtained with a raised-cosine weighting. Raised-cosine weighting clearly provides much more clear spectral lines, with no INEXSISTENT sidebands. On the other hand, weighting slightly reduces the frequency resolution making the peaks broader. Selecting weighting or not is therefore a trade-off between dynamic range and frequency resolution.

The Fourier transform is computed on real data only, therefore the FFT algorithm can be used in a more efficient way: either to produce two independent transforms at the same time or a double length transform. The Fourier transform and the FFT algorithm provide a complex signal spectrum: each spectral line is represented by a complex number in the real/imaginary-component format. The phase information represented by the argument of the complex numbers is usually discarded, since the timing relationship between the spectrum analyzer time base and the signal examined is usually not known. To compute the magnitude from the real and imaginary components, two multiplications, one addition and one square root are required for each sample. Unfortunately the square root is a very time-consuming operation on most computers.

If a measurement is affected by random noise, averaging among a number of otherwise identical measurements improves the accuracy of the result. Automatic averaging is very easy to implement on any microprocessor-controlled test equipment. Averaging is used in spectrum analyzers to improve the signal-to-noise ratio of the displayed data. The improvement that can be obtained by averaging is shown on Fig. 4.5. and Fig. 4.6. All four plots were obtained from the same input signal in an identical way except for different amounts of averaging.

The linear-to-logarithmic conversion includes the computation of a logarithm for each spectral component. Since the logarithm is a rather "slow" function on digital computers, a lookup-table algorithm or a similar approach has to be used to avoid unnecessary loading of the computer. The same constraint applies to the display procedure: most computers require more time to draw a high-resolution plot than to compute the FFT algorithm. The display routine and/or dedicated hardware has to be quick enough to avoid slowing-down the spectrum analyzer. On the other hand, a hard-copy of the video display is usually very easy to obtain on any computer using a standard printer or plotter, at least when compared to analog instrumentation with CRT displays.

Besides the conventional frequency/amplitude function plot other types of display are possible on digital computers. A practically very useful type of display is the intensity spectrogram. In the latter frequency is still plotted on the horizontal axis. Each FFT result is however represented by a

single image line and the pixel brightness is used to represent the magnitude of a spectral component. The results of successive FFTs are plotted on successive lines, showing the results of a large number of consecutive measurements on just one computer screen. Intensity spectrograms are useful when analyzing continuously changing signals.

The display of a FFT spectrum analyzer usually includes the complete frequency range covered by the FFT algorithm: from zero to half the signal sampling frequency. The frequency resolution is then simply equal to the frequency span divided by the number of lines displayed. The frequency resolution may be slightly worse if weighting is used in front of the FFT algorithm. Of course partial displays are possible too, showing just a subset of spectral lines computed by the FFT algorithm. The sample-and-hold circuit in front of the A/D converter is an excellent harmonic mixer. It is therefore possible to observe a different frequency band just by replacing the input low-pass filter with a bandpass filter for the selected frequency range.

Finally, a comparison has to be made between a FFT-based spectrum analyzer and a scanning-receiver type spectrum analyzer (conventional analog RF spectrum analyzer). Of course a scanning-receiver type spectrum analyzer could be implemented on a digital computer as well. A FFT-based spectrum analyzer has however a very important advantage over a scanning-receiver spectrum analyzer: regardless of the hardware used the FFT spectrum analyzer uses the available spectral information in a much more efficient way resulting in a much quicker operation.

As an example, consider that a 5kHz wide frequency band has to be analyzed to a resolution of 10Hz. A scanning receiver with a 10Hz bandwidth has to dwell on each 10Hz frequency step for about 0.1 seconds, resulting in a total sweep time of about 50 seconds! On the other hand, a FFT-based spectrum analyzer needs to sample a 5kHz wide signal with a sampling frequency of 10kHz. A 1024-point FFT has to be used to obtain a 512-point display, so the total "scanning" time is 0.1024 seconds!

In the above real-world example the FFT spectrum analyzer is about 500 times faster! The reason for this is that a conventional scanning-receiver spectrum analyzer only uses the information contained in its receiver bandwidth, all the other information contained in the signal is simply rejected! On the other hand, the FFT-based spectrum analyzer uses all of the information contained in the signal since the FFT algorithm corresponds to a bank of 512 parallel bandpass filters in the above example. Such a bank of filters would be prohibitively expensive and difficult to make using conventional analog technology.

A FFT-based spectrum analyzer can therefore be used in applications where a conventional scanning-receiver type spectrum analyzer is not practical due to the too long scanning time or completely useless since the signal is not available for the scanning time period required. Even in the case when the scanning-receiver type can be used, the FFT-type can provide a much more accurate result in the same time, averaging among a large number of measurements.

Unfortunately the bandwidth and dynamic range of digital FFT spectrum analyzers are severely limited by the available hardware, mainly A/D converters. It is therefore necessary to understand the advantages and disadvantages both techniques to select the most suitable one for a particular problem, since the two techniques are complementing each other rather than competing at the present state of technology.

5. Amateur applications of a FFT spectrum analyzer

Although FFT spectrum analysis is presently limited to the audio-frequency range or slightly above, it has many interesting and very useful amateur-radio applications. In the following section a few typical amateur-radio applications will be presented including the spectrum plots and intensity spectrograms obtained. All of the latter were obtained by connecting the output of an amateur SSB or FM receiver to the MC68010 based DSP computer described in a series of articles in UKW-Berichte/VHF-Communications. All of the plots were obtained with a FFT spectrum analyzer program including a 1024-point FFT algorithm and all the other features described in the previous section. Unfortunately the intensity spectrograms could only be printed in black-and-white with no gray levels, so they can not represent all of the information that was visible on the computer screen.

A FFT spectrum analyzer is a useful tool when building a SSB receiver or transceiver. One of the most difficult tasks when building a SSB receiver is to measure the passband of the crystal filters used, regardless whether are they home-brew or commercially available items. To obtain a reliable result, a very stable sweep generator is required in addition to a storage oscilloscope due to the slow sweeping speed required. Alternatively, a FFT spectrum analyzer can be connected to the receiver audio output and a wide-band noise source to the receiver input (if the receiver own noise is not sufficient). Thanks to the speed of the FFT spectrum analysis the result can be obtained quicker than with the sweep generator, allowing real-time tuning of the trimmers in and around the crystal filters. Two typical results, plotted on a logarithmic amplitude scale, are shown on Fig. 5.1.: a 2kHz SSB filter above and a 500Hz CW filter below. Using FFT spectrum analysis the trade-off between accuracy and speed is selected by choosing the averaging factor. However, even with no averaging, the FFT analysis will only provide a noisy plot while a too fast sweep generator will provide a completely distorted and thus useless result.

A FFT spectrum analyzer is able to reliably detect very weak signals hidden in noise, far beyond what a human ear can do, since it is not limited to certain frequency bands, resolution bandwidths or averaging intervals. The plots on Fig. 5.2. were obtained by tuning a 2m amateur SSB receiver to a weak CW beacon and then decreasing the signal level with an input attenuator to obtain a signal-to-noise ratio of about -15dB in a 2.5kHz bandwidth or -5dB in a 250Hz bandwidth.

Although the signal-to-noise ratio is around +10dB in the spectrum analyzer resolution bandwidth, some averaging is required to reliably detect the signal. On the frequency/amplitude plot (above on Fig. 5.2.) the averaging was performed by the computer and then the result was plotted. On the intensity spectrogram (below on Fig. 5.2.) no averaging was performed by the computer. Averaging is however performed by our eyes when observing the spectrogram!

Even weaker signals could be detected either by increasing the frequency resolution or by increasing the averaging factor or both. The practical limit is mainly imposed by the time required for the signal to be available for a reliable detection. In a practical communications system there are other constraints too: receiver and transmitter frequency instability or phase noise and propagation effects. The FFT spectrum analyzer can solve the problem of frequency uncertainty, since it allows to observe a frequency band of a few kHz instantaneously with a resolution of 5 to 10Hz. The phase noise of transmitters and receivers should be minimized anyway. Unfortunately, some propagation effect also show up as phase or amplitude noise, especially in EME (moon-bounce) communications. These effects are proportional to the carrier frequency so major advantages of using FFT techniques for EME communications can only be expected on VHF and UHF frequencies. On these frequency bands FFT signal processing may decrease the RF link performance requirements by 10 to 20dB, since higher figures would result in useless data rates. In any case, a communications protocol has to be agreed upon before these techniques can be used: hand-keyed CW is certainly not a good choice for computer processing.

Fig. 5.3. shows how a Morse-keyed CW transmission looks on an intensity spectrogram. The keying was too quick for the dots and dashes to appear on the spectrogram, the interruptions correspond to the longer spaces between letters. In order to be able to see the single dots, much more frequent FFTs should be made and each FFT should be taken on less data samples since time and frequency resolution are of course reciprocal.

Receiver AGC effects are easily visible, increasing the noise level during longer pauses. The spectrogram above was obtained from the AO-13 Mode-B 2m beacon and shows a slight "FM-ing": the trace has "tails" to the right at the beginning of each transmission. The spectrogram below was obtained from the LUSAT-1 70cm CW beacon. A quickly changing Doppler frequency shift is easily visible, as well as an interference to the left corresponding to the third harmonic of the signal, generated somewhere in the audio stages of the SSB receiver.

A FFT spectrum analyzer can be used to identify, tune-in and measure the parameters of a FSK RTTY transmission as shown on Fig. 5.4. The two RTTY tones are easily visible both on the amplitude/frequency display and on the intensity spectrogram. Both of them were obtained from the AO-13 50 baud RTTY Mode-B beacon. In the middle of the spectrogram a period with no keying is visible: only one of the tones, un-modulated, is transmitted during this period. During active keying the tone traces become wider due to the 50 bps modulation sidebands.

A FFT spectrum analyzer can therefore be considered as an

up-to-date replacement for the old-fashioned RTTY oscilloscope tuning indicator. Both tone frequencies and the keying shift can be quickly and accurately determined. In addition, a FFT display provides information about signal distortion (selective fading) or interferences. Especially in the latter case a FFT display will provide some useful information about the countermeasures to be taken (notch filter tuning for example) and about their effectiveness.

The parameters of a FSK or AFSK signal are more difficult to identify if the data rate is comparable to the frequency shift, as shown on Fig. 5.5. 1200 bps packet-radio uses 1200 Hz and 2200 Hz tones, the shift is therefore 1000 Hz, comparable to the data rate of 1200 bps. The spectrum of such a transmission is an almost contiguous frequency band with just a few peaks, that do not necessarily correspond to the tone frequencies as shown on Fig. 5.5. above. The intensity spectrogram below shows the intermittent nature of packet--radio signals. Due to the short duration of the packets, their spectrum could never be obtained by a scanning-receiver type spectrum analyzer.

A FFT spectrum analyzer can be used as a valuable tuning aid for PSK packet-radio communications. Phase-shift keying is used for satellite packet-radio communications since it allows a longer communications range with the same RF equipment performance. On the other hand, PSK and other coherent techniques require accurate tuning and good frequency stability.

The spectrum of a 1200 bps packet-radio transmission from the PACSAT-1 satellite is shown on Fig. 5.6. The plot above shows the spectrum during the transmission of flags between the packets. On this plot it is easy to identify the carrier, surrounded by sidebands spaced at 150 Hz. During the transmission of packets containing random data the spectrum looks almost like perfect noise (Fig. 5.6. below) and it is much more difficult to identify the correct tuning. In fact a straightforward PSK transmission contains very little redundancy and there are no residual carriers either.

Some more redundancy can be noticed in the AO-13 400 bps BPSK Mode-B beacon transmission due to Manchester coding. Fig. 5.7. above shows the spectrum during the transmission of the filling bytes (5OH) in between the data frames: this repetitive pattern generates the many discrete spectral lines. On the intensity spectrogram on Fig. 5.7. below it is easy to identify the single data frames and the filling byte periods in-between them. Even data frames sometimes contain repetitive patterns causing discrete spectral lines inside the data frames. Regardless of the modulation data the signal spectrum has two main lobes caused by the Manchester coding. The latter are positioned symmetrically around the carrier, suppressed by this modulation technique.

A FFT audio spectrum analyzer is useful to check WEFAX and APT satellite signals and related receiving equipment. Using a FFT spectrum analyzer important details of an unknown satellite transmission, like that of a new satellite, can easily be determined. Further, the receiving equipment can be checked for a correct deemphasis (to avoid loosing

geometrical resolution) and sources of eventual interferences.

All this is easier to describe on a well known example, like the Meteosat WEFAX transmission shown on Fig. 5.8. On the frequency/amplitude plot on Fig. 5.8. above the strongest spectral component is the 2400 Hz sub-carrier. The sub-carrier should be surrounded by symmetrical sidebands if the deemphasis is properly adjusted. The sidebands depend of course on the picture content. The most notable detail are the two symmetrical peaks corresponding to the line-sync 840 Hz bursts. These have a rather rounded peak, since their duration is very short: their spectral width is inversely proportional to their duration. The intensity spectrogram shown on Fig. 5.8. below shows that the sync bursts appear in about every second FFT conversion. Also, the spectrum close to the 2400 Hz sub-carrier also changes with the picture line period. The annotation transmitted at the end of the picture is well visible, followed by the discrete spectral lines of the 450 Hz stop tone lasting 5 seconds. What follows is just part of the spectrum of a DCP retransmission between two WEFAX pictures. The DCP flags create two strong discrete spectral line traces on the spectrogram, interleaved with three DCP data frames visible on Fig. 5.8. below.

The spectrum of a NOAA APT transmission, shown on Fig. 5.9. is a little more complex. There is still a strong 2400 Hz sub-carrier component, but there are two different sync bursts of 832 Hz and 1040 Hz respectively. The side-lobes generated by both sync bursts are well visible on Fig. 5.9. above. On the intensity spectrogram on Fig. 5.9. below it is easy to notice that these sync pulses appear interleaved in the APT signal. The remaining spectrum also changes with the same period.

Although a FFT spectrum analyzer is actually an audio--frequency test equipment, it has many interesting applications in the amateur-radio field. The above examples were chosen just to show a few possible applications of a FFT spectrum analyzer and the many different ways a FFT spectrum analyzer can be used. Probably most of the applications are yet to be discovered, since FFT spectrum analysis only became available to amateurs with recently developed low cost and high performance microprocessors and A/D converters.

6. The FFT spectrum-analyzer program for the DSP computer

The FFT algorithm can be implemented on almost any digital computer, ranging from 8-bit video-game toys to the largest and fastest mainframes. The FFT algorithm requires very little memory, since the results of a "butterfly" operation are stored in the same two memory locations where the arguments were taken from. Therefore the memory requirements are very low, including just a buffer of the size of the input data block and another buffer of a similar size to hold the phase coefficients.

Of course the execution time will depend on the computer used. Computing a 1024-point complex FFT on a 8-bit

microcomputer programmed in a high-level language like BASIC will require several minutes. Computing the same 1024-point FFT on an IBM AT compatible equipped with the mathematical co-processor and executing a compiled program will still require several seconds. A well-written machine code program for a general-purpose 16-bit microprocessor can compute a 1024-point FFT in a few hundred milliseconds using 16-bit integer arithmetics. Finally, dedicated DSP microprocessors are able to compute a 1024-point FFT in tens of milliseconds with the top-of-line devices requiring a few milliseconds.

A FFT spectrum-analyzer program was written for the DSP computer published in UKW-Berichte/VHF-Communications. This computer includes a 8-bit logarithmic A/D converter input port and a 512-pixel by 256-line, 256-gray-level video display. The MC68010 microprocessor used in this computer is able to compute a 1024-point complex FFT in about 150 milliseconds. Considering that some CPU time is required for other functions too, like interrupt handling or display update, the maximum sampling frequency for successive but non-overlapping FFTs without skipping any input data is about 8 kHz. This figure matches very well the performance of the switched-capacitor filter in front of the A/D converter as well as the audio bandwidth available from a communications or amateur receiver.

The FFT spectrum-analyzer program for the DSP computer includes all of the features from Fig. 4.1.: A/D conversion, data buffering, FFT algorithm with magnitude computation and display. Weighting, averaging, LIN/LOG conversion, display format and hard-copy are implemented as user-selectable options. The complete set of commands is shown on Fig. 6.1. The same HELP menu is obtained each time a wrong command is issued. Otherwise, the program uses 248 image lines for the function plot or spectrogram and the 8 bottom lines to annotate the main program parameters.

The program parameters are selected by typing their initial letter as shown on the HELP menu. The selected parameter will then appear in the annotation text line. The selected parameter value can be then adjusted: increased or decreased, by using respectively the + and - keys. The * key will set ALL of the parameters to their default values.

The program parameters include:

AVERAGING: averaging factor, ranging from 1 (no averaging) to 4096 and adjustable in powers of 2.

BORDER, DIVISIONS, ERASER, INK, PAPER: gray-level count, adjustable between 0 and 248 in steps of 8.

FUNCTION DISPLAY: selects an amplitude/frequency function-plot type display as indicated in the annotation text.

GAIN: adjusts the gain between the A/D converter output and FFT input to increase the dynamic range with low-level signals. Usually set to 1 to prevent saturation.

HARDCOPY: each time H is depressed, a hard-copy file PLOT.DAT is created. The format is selected to be understood by most dot-matrix and laser printers. Since these printers can not print gray levels, the threshold between black and white is set to a gray-level count of 127/128!

OVERLAP: switches the overlap (50%) option ON or OFF (toggle operation). When ON, the overlap option is indicated by an "O"

in the annotation text.
RESOLUTION: selects the sampling frequency and therefore the full-scale frequency span or resolution between 301 and 6400 Hz by setting the sampling-frequency divider modulo.
SPECTROGRAM DISPLAY: selects an intensity spectrogram type display, as indicated in the annotation text.
TRIGGER: starts or stops (toggle operation) the operation of the spectrum analyzer. The status RUN or STOP is indicated in the annotation text. In the STOP mode, the display is "frozen".
VIDEO: selects the video gain after the FFT algorithm in the LIN mode. The lowest setting selects a LOG video scale (default).
WEIGHTING: switches the raised-cosine weighting option ON or OFF (toggle operation). When ON, the weighting option is indicated by a "W" in the annotation text.

The performance of this FFT spectrum analyzer is of course limited by the analog hardware used: input bandpass filter and A/D converter. Fig. 6.2. above shows the passband of the input filter, obtained with the same spectrum analyzer by simply connecting a wide-band noise source to the input. The dynamic range of the 8-bit logarithmic A/D converter used is only about 40dB due to quantization noise. The performance of the spectrum analyzer is however better: in many practical cases the display dynamic range reaches 60dB since the quantization noise usually appears as wide-band noise.

The input switched-capacitor filter and A/D converter own noise is well visible on Fig. 6.2. below. Using a logarithmic amplitude display the quantization steps become very large at low signal levels. In the LOG vertical mode, the full scale amplitude range is 15 bits or 90dB. The difference between counts 1 and 2 should therefore correspond to the difference between counts 16384 and 32768 on a LOG scale, however in the former case there are no additional counts possible between 1 and 2 resulting in relatively large "steps" on the display.

With a 10 MHz CPU clock, the FFT spectrum-analyzer program is able to operate with a sampling frequency of about 8 kHz (resolution 4 kHz) on contiguous but non-overlapping data. The input data buffer routine is designed to skip data blocks automatically if the CPU is unable to further process them in real time, like in the case of a higher sampling frequency or overlapping FFTs. Sometimes it is very useful to be able to operate in these conditions even if some data is lost. On the other hand, an analog scanning-receiver type spectrum analyzer usually discards over 99% of the information contained in the input signal!

The FFT program includes a 1024-point complex FFT algorithm. The algorithm is used to process two independent sets of 1024 consecutive input samples. The results of the FFT are then separated using the Fourier transform symmetry laws into two independent 512-point signal spectra. The phase information is rejected and the magnitudes are computed using a quick approximation for the square root. The LIN/LOG conversion is obtained with a lookup-table algorithm to avoid the even slower log function.

The display routine includes automatic limiting if the magnitude exceeds the top on a function plot or the maximum

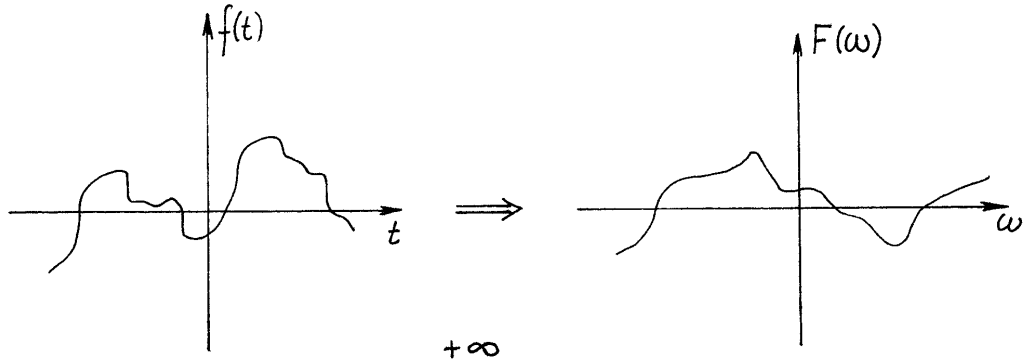
brightness on an intensity spectrogram. Hard-copies of the computer screen in the various modes are used throughout this article.

The FFT spectrum-analyzer program is written partially in the MC68010 machine code and partially in the DSP computer high-level language. The commented source program is about 15 kilobytes long while the compiled program requires about 47 kilobytes. This is actually less than most other DSP programs thanks to the compactness and low memory requirements of the FFT algorithm. Of course the FFT program can run in multi-task with other programs that do not use the same peripherals, like the TRACK program.

Future upgrades to the FFT spectrum-analyzer program should include the capability to adjust the size of the FFT (number of points) according to the requirements of the measurement: some problems (quickly changing signals) require less than 1024 points while others require a higher resolution and thus more than 1024 points. The 1024-point FFT is however a good compromise for most measurements.

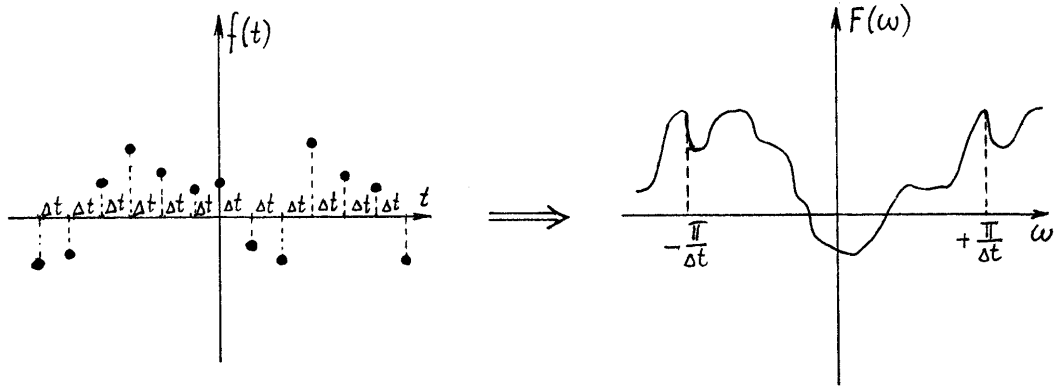
Finally, it is hoped that this article will encourage amateurs to use the FFT algorithm and FFT spectrum analysis, since the latter can easily be implemented on the described DSP computer and on many other computers as well. Presently FFT spectrum analysis is only available on top-of-line professional instrumentation and the reason for this is probably the poor understanding of this new technique by the majority of spectrum-analyzer users.

****



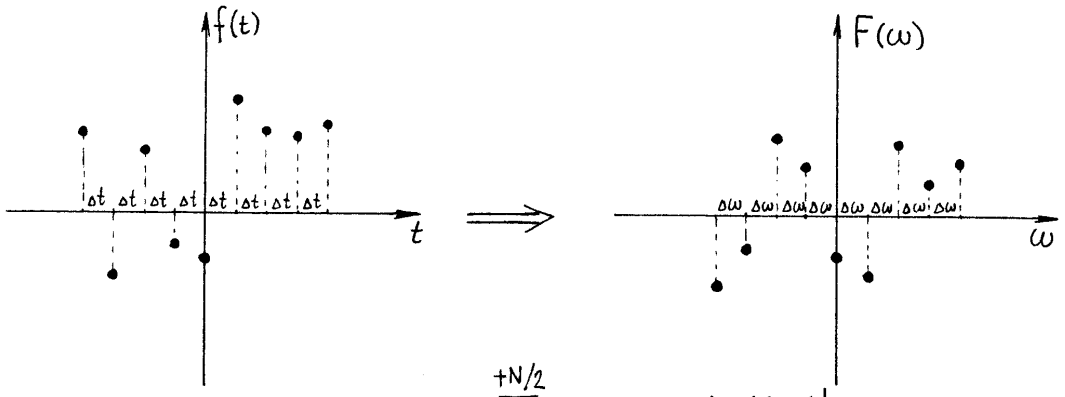
$$F(\omega) = \int_{-\infty}^{+\infty} f(t)\, e^{j\omega t}\, dt$$

Fig. 2.1. - Definition of the Fourier Transform.



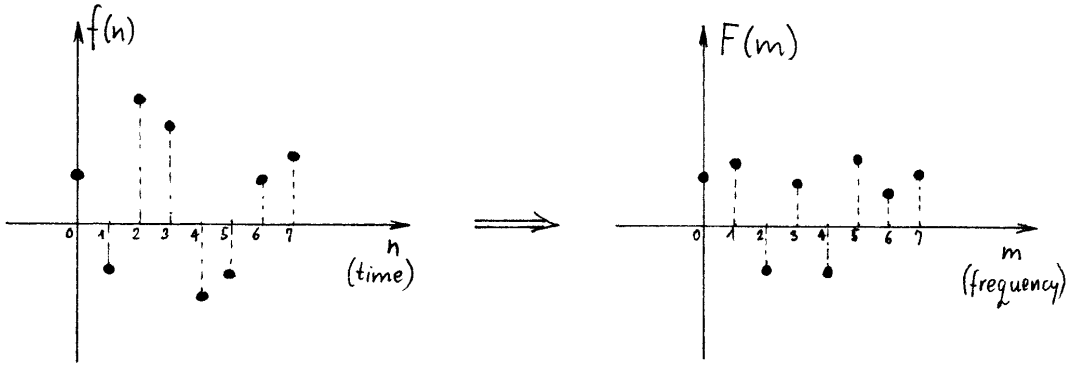
$$F(\omega) = \sum_{n=-\infty}^{+\infty} f(n\Delta t)\, e^{j\omega n \Delta t}$$

Fig. 2.2. - Fourier Transform of a sampled signal.



$$F(m\Delta\omega) = \sum_{n=-N/2}^{+N/2} f(n\Delta t)\, e^{jm\Delta\omega n\Delta t}$$

Fig. 2.3. - Definition of the Discrete Fourier Transform.



$$F(m) = \sum_{n=0}^{N-1} f(n)\, e^{j\frac{2\pi}{N}nm}$$

Fig. 2.4. - Normalizing time and frequency units.

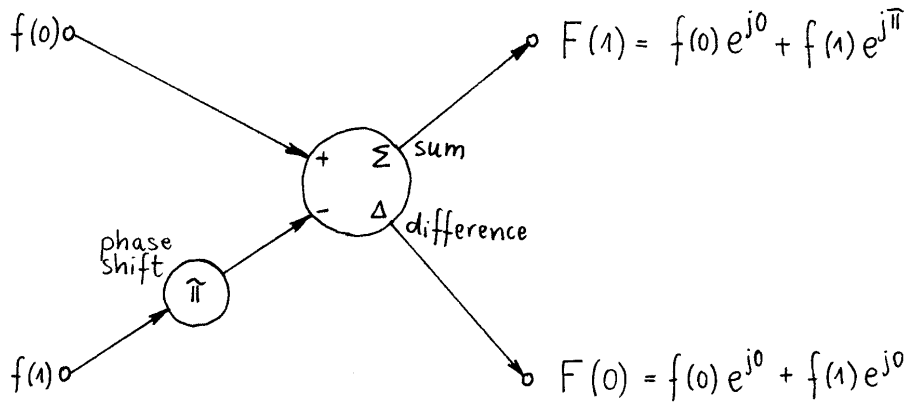


Fig. 3.1. - Two-point FFT - a single "butterfly" operation.

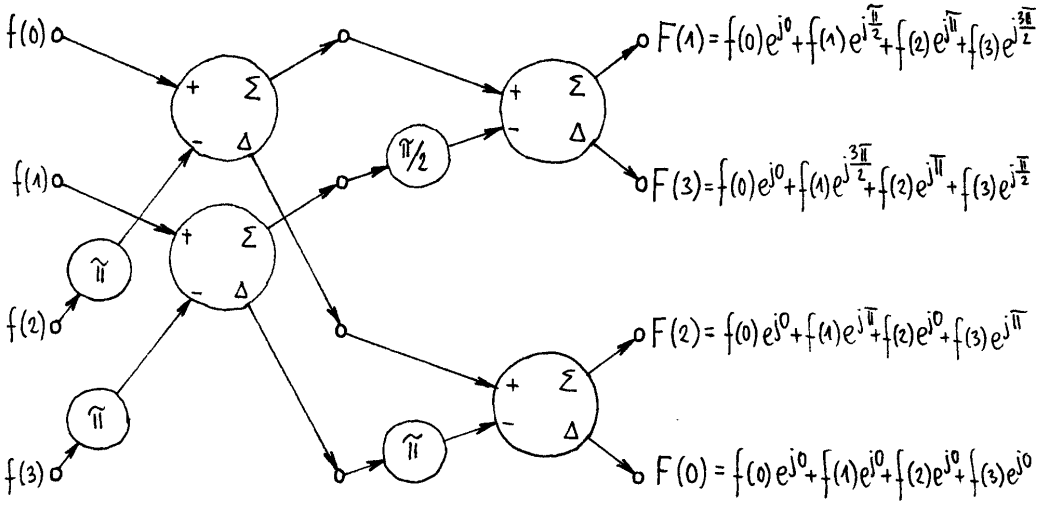


Fig. 3.2. – Four-point FFT, computed in two steps, each using two "butterfly" operations.

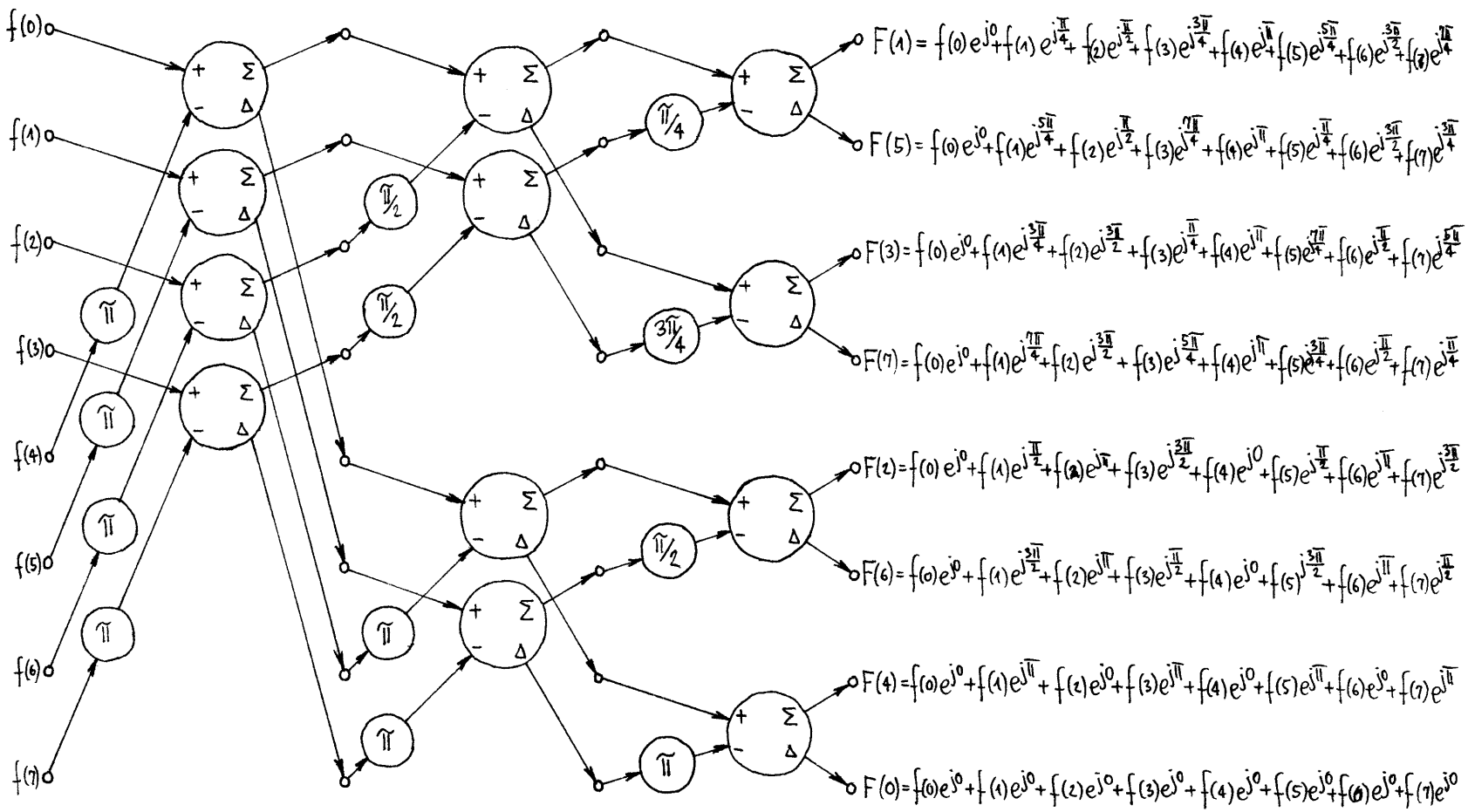


Fig. 3.3. – Eight-point FFT, computed in three steps, each using four "butterfly" operations.

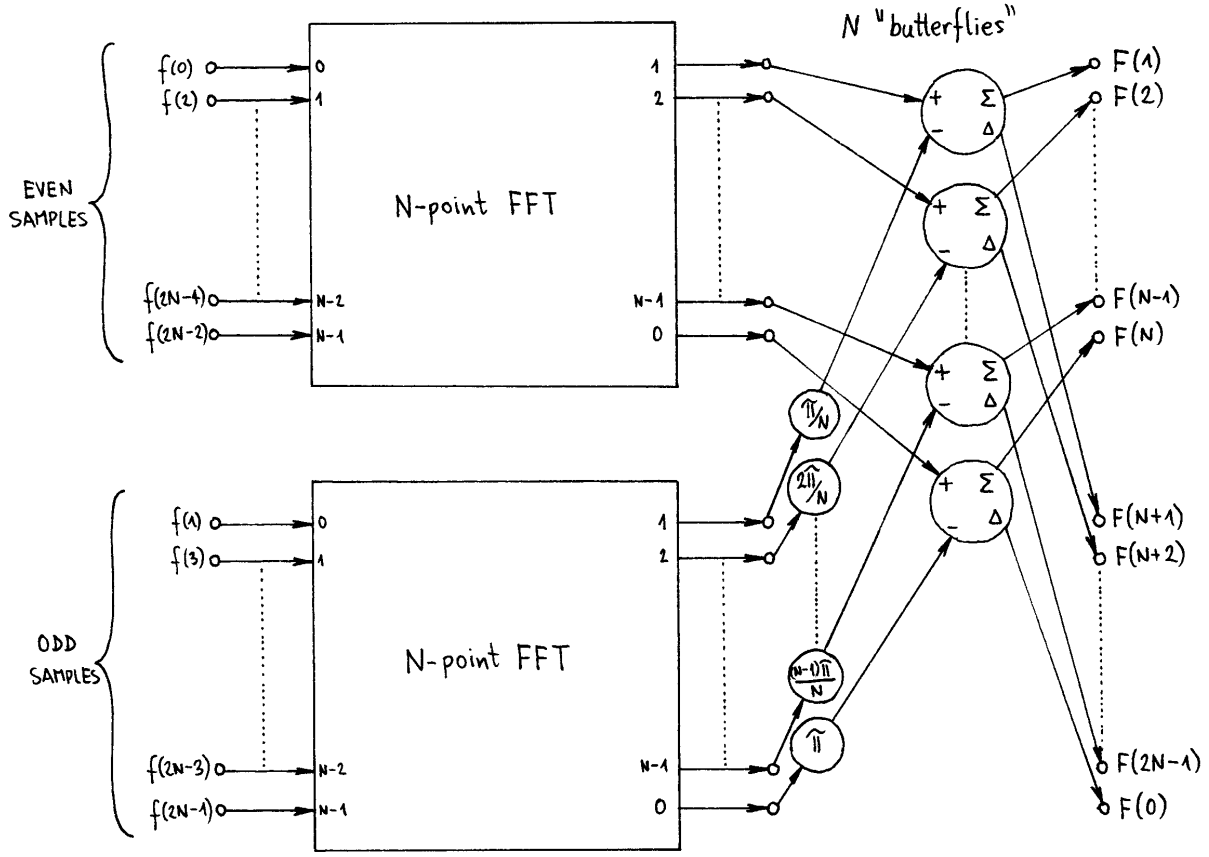


Fig. 3.4. - Obtaining a double length (2N) FFT from two single length (N) FFTs.

OUTPUT BIT-REVERSED MAPPING
==========

```
f(0) >>---> 0 = 000B >>---> 000B = 0   0 + 1 = 1 >>---> F(1)
f(1) >>---> 1 = 001B >>---> 100B = 4   4 + 1 = 5 >>---> F(5)
f(2) >>---> 2 = 010B >>---> 010B = 2   2 + 1 = 3 >>---> F(3)
f(3) >>---> 3 = 011B >>---> 110B = 6   6 + 1 = 7 >>---> F(7)
f(4) >>---> 4 = 100B >>---> 001B = 1   1 + 1 = 2 >>---> F(2)
f(5) >>---> 5 = 101B >>---> 101B = 5   5 + 1 = 6 >>---> F(6)
f(6) >>---> 6 = 110B >>---> 011B = 3   3 + 1 = 4 >>---> F(4)
f(7) >>---> 7 = 111B >>---> 111B = 7   7 + 1 = 8 >>---> F(0)
```

Fig. 3.5. - Output bit reversed mapping for a 8-point FFT.

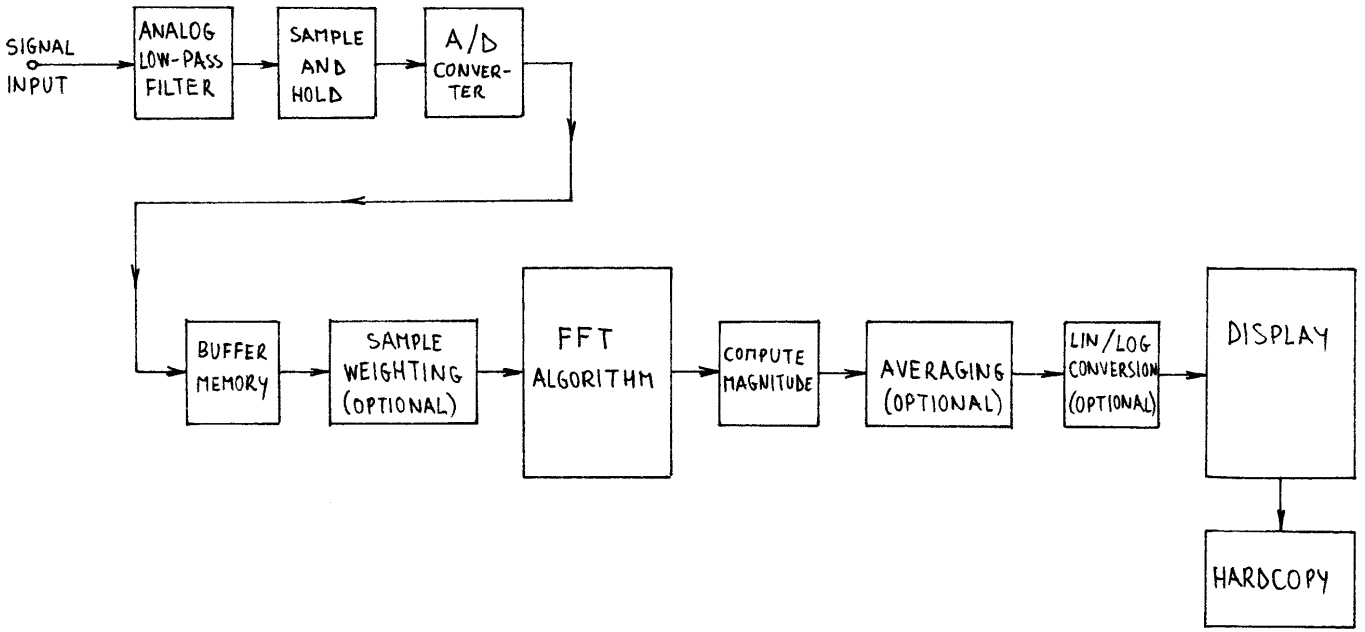


Fig. 4.1. – Block diagram of a FFT spectrum analyzer.

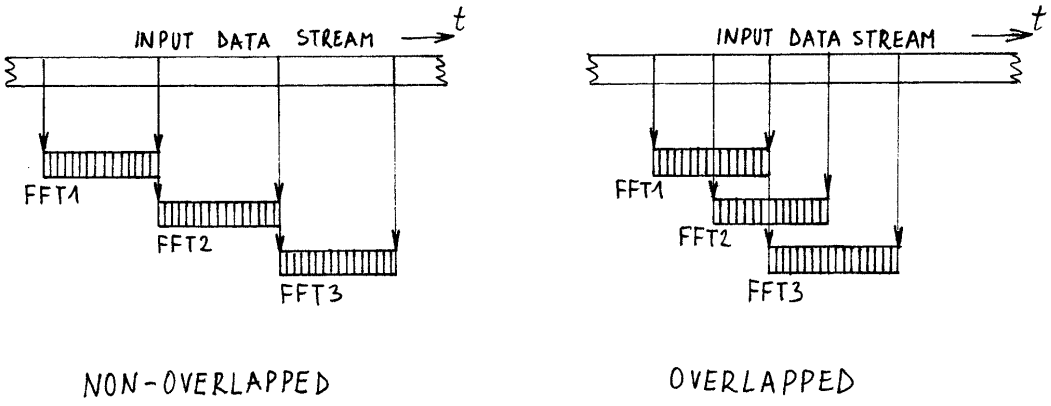


Fig. 4.2.-Overlapping FFTs.

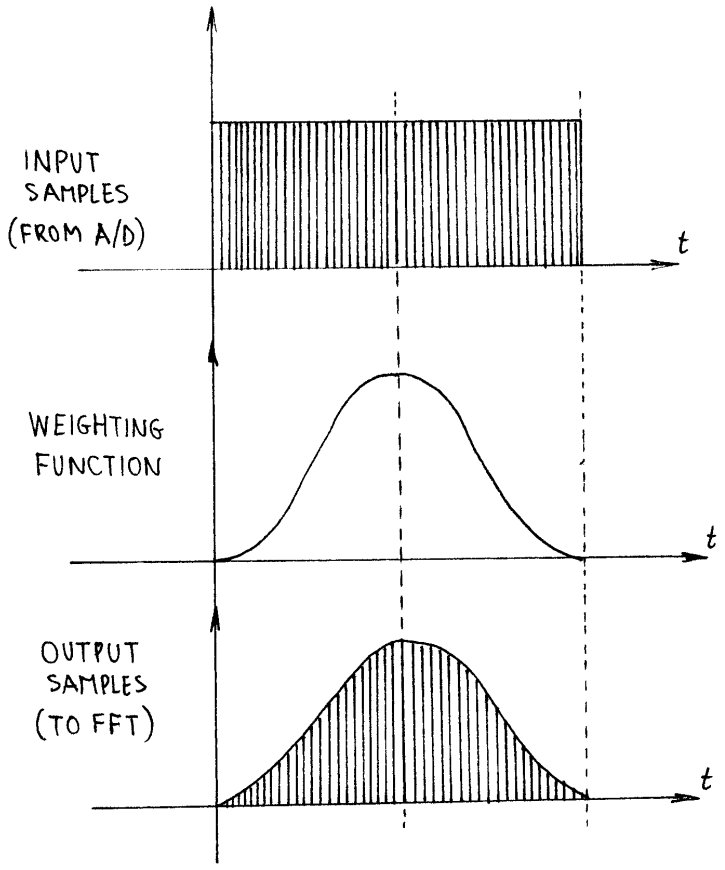
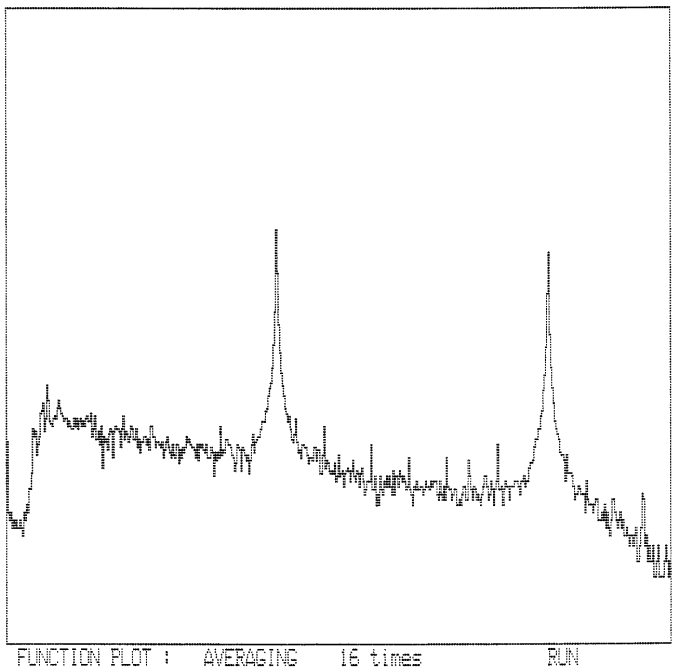


Fig. 4.3. - Weighting signal samples.



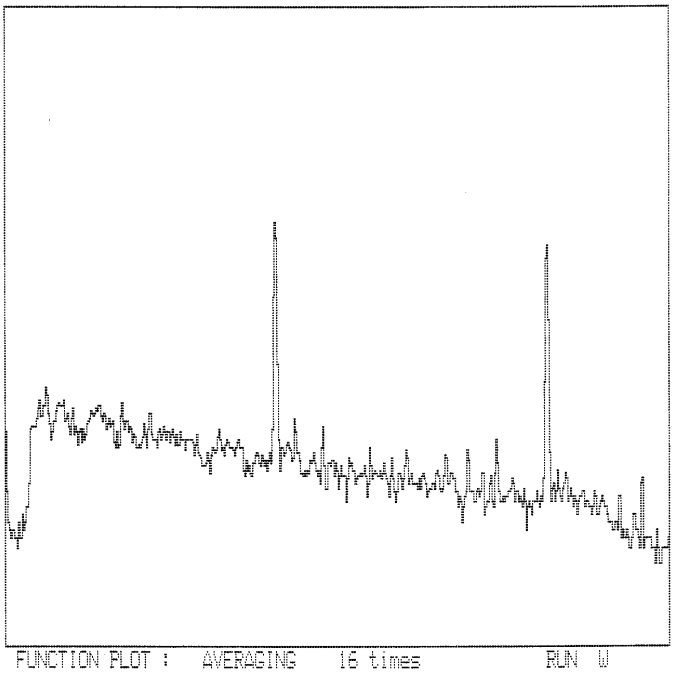
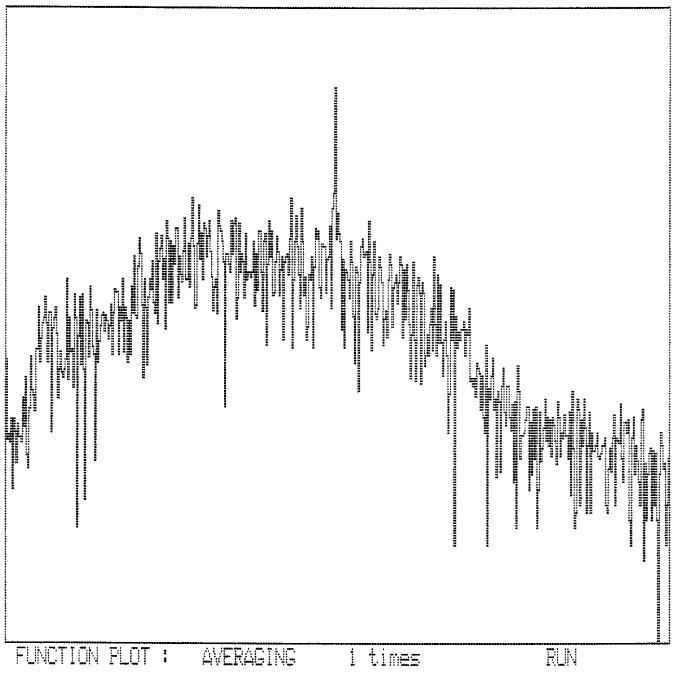


Fig. 4.4.- No weighting (above) versus raised-cosine weighting (below), same input signal, LOG vertical scale.



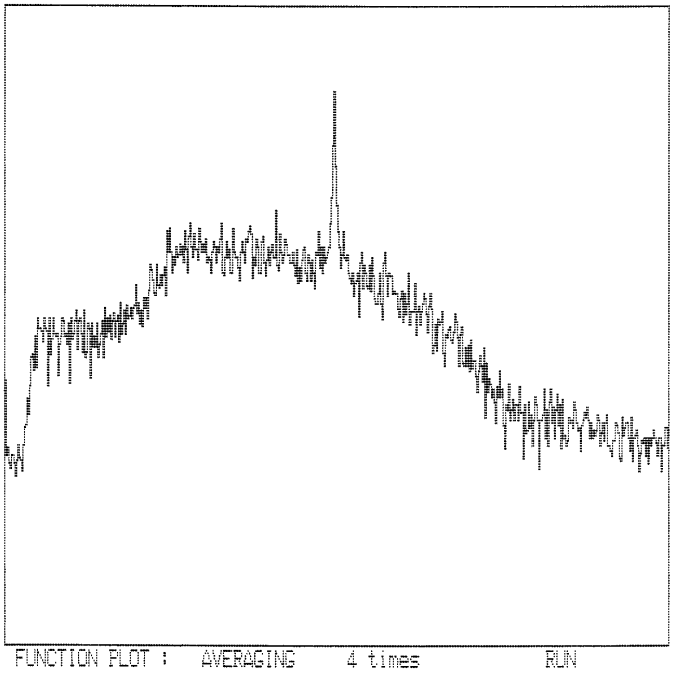
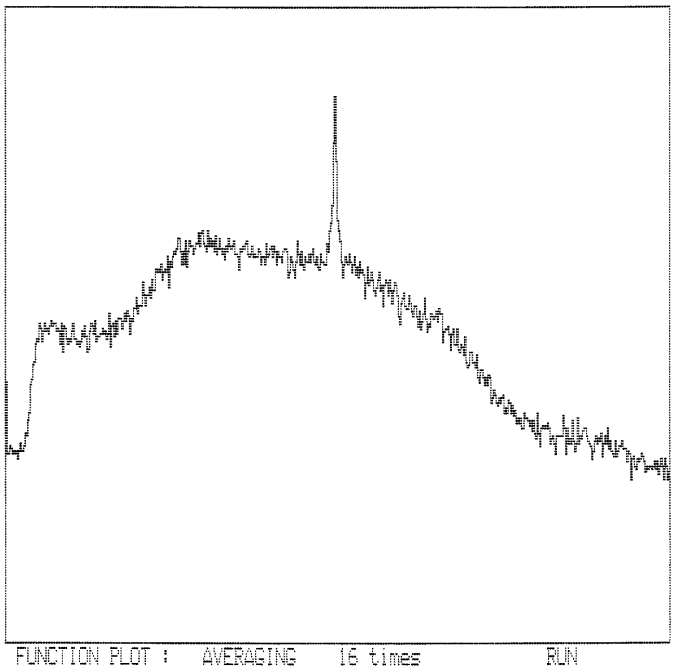


Fig. 4.5.- No averaging (above), averaging 4 times (below), same input signal and other settings as on Fig. 4.6.



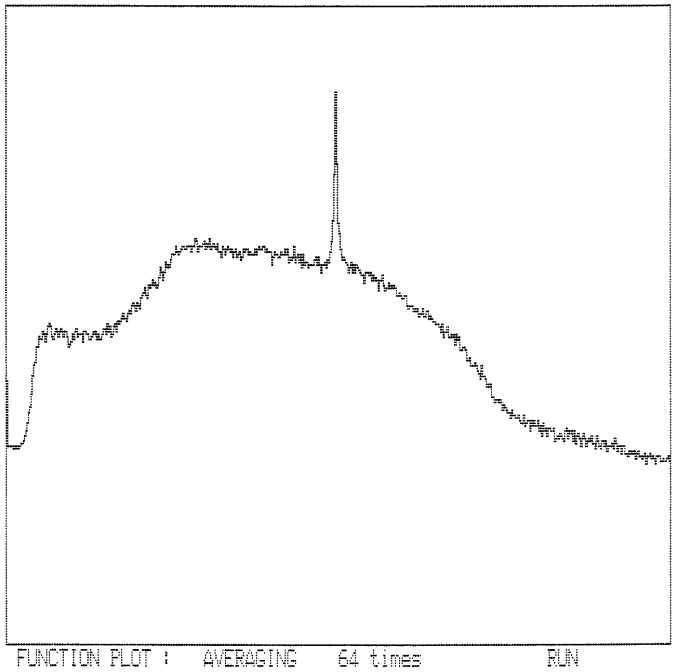
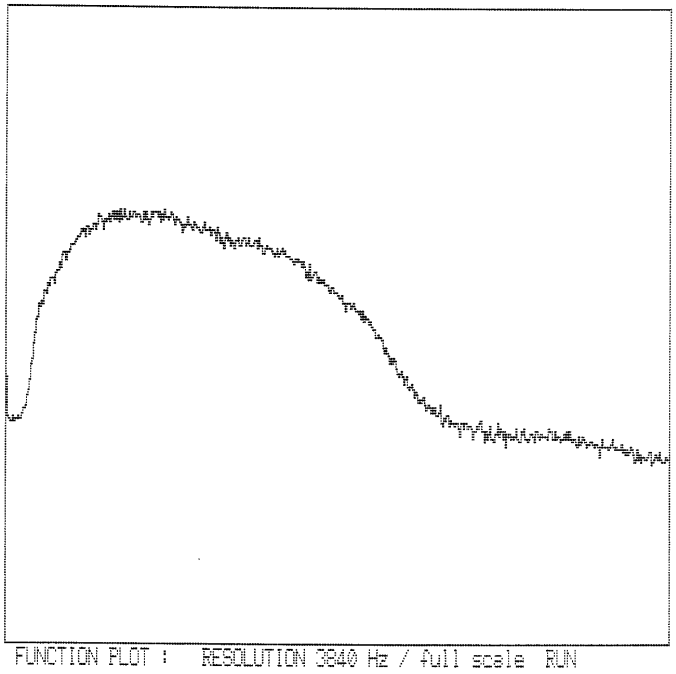


Fig. 4.6.- Averaging 16 times (above) or 64 times (below), same input signal and other settings as on Fig. 4.5.



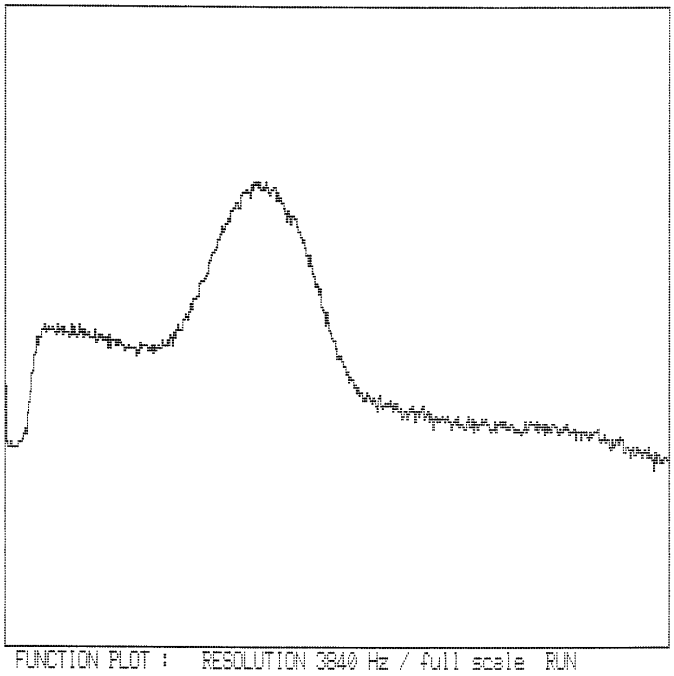


Fig. 5.1.- Audio frequency response of a SSB receiver: SSB filter (above) and CW filter (below).

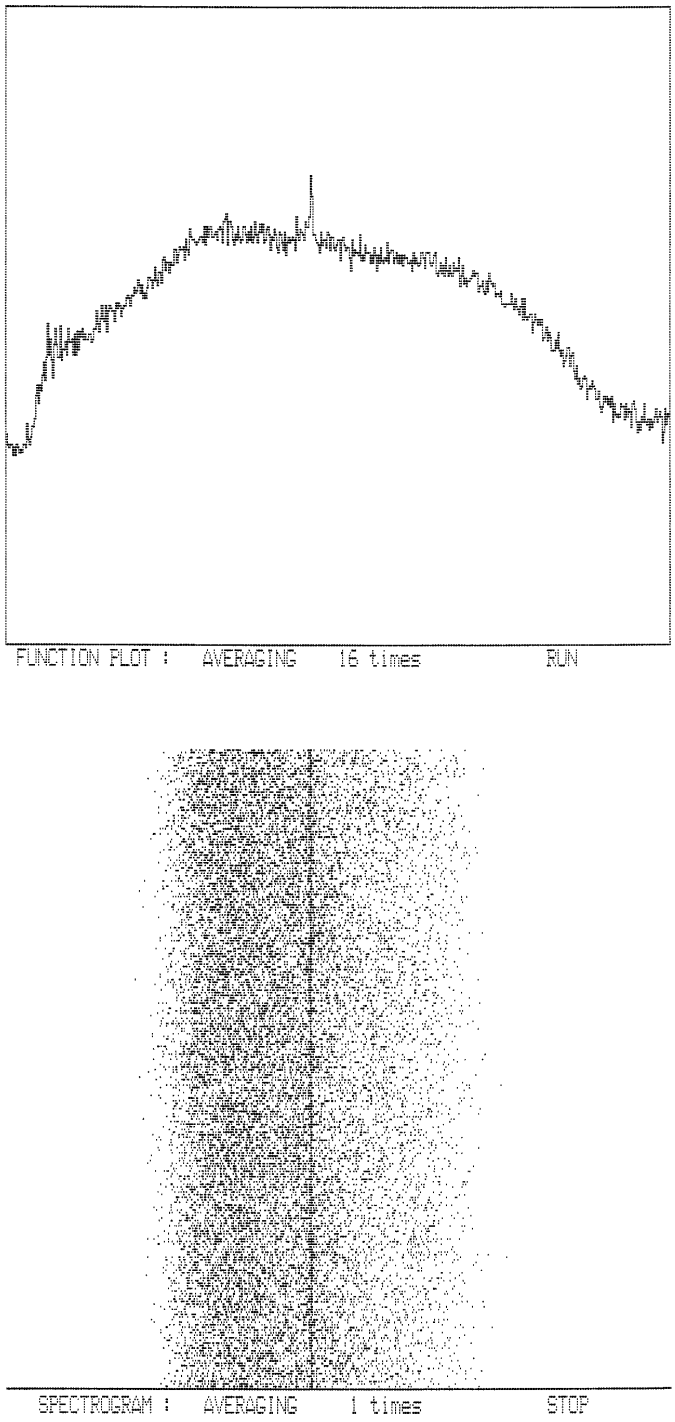


Fig. 5.2.- Weak signal detection: averaging, LOG scale (above) versus intensity spectrogram (below).

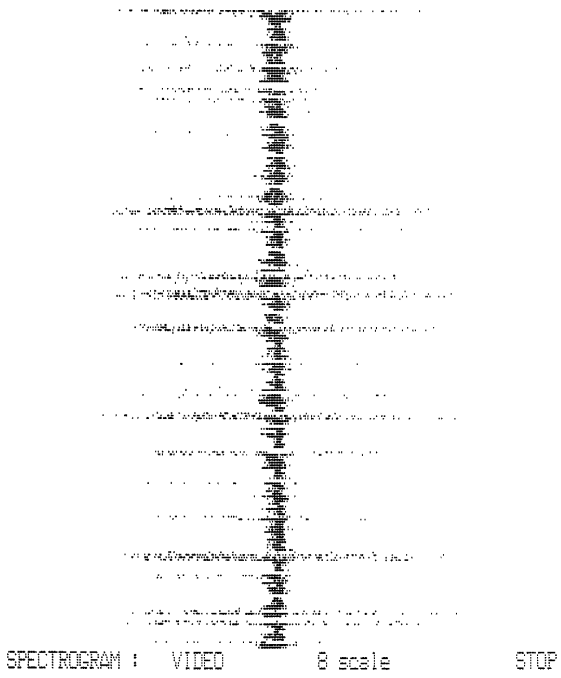
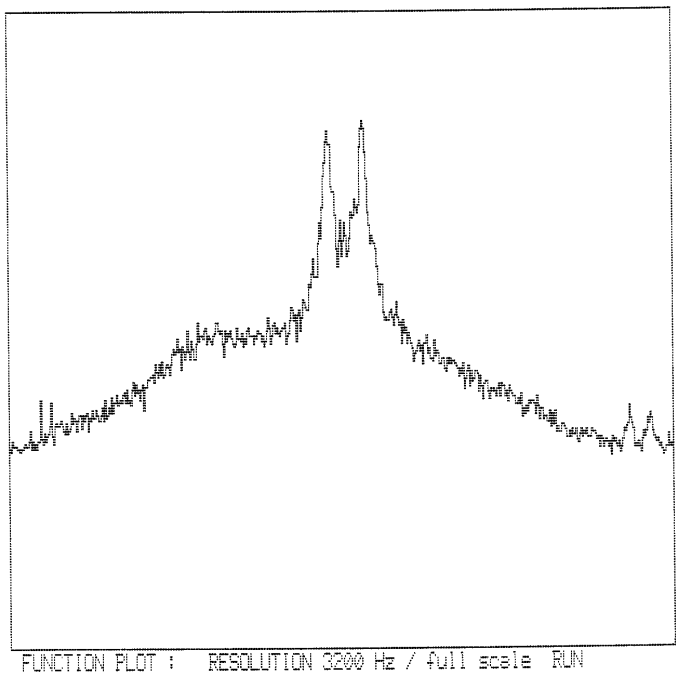


Fig. 5.3.- CW signal with a slight "FM-ing" (above) and affected by a quickly changing Doppler shift (below)



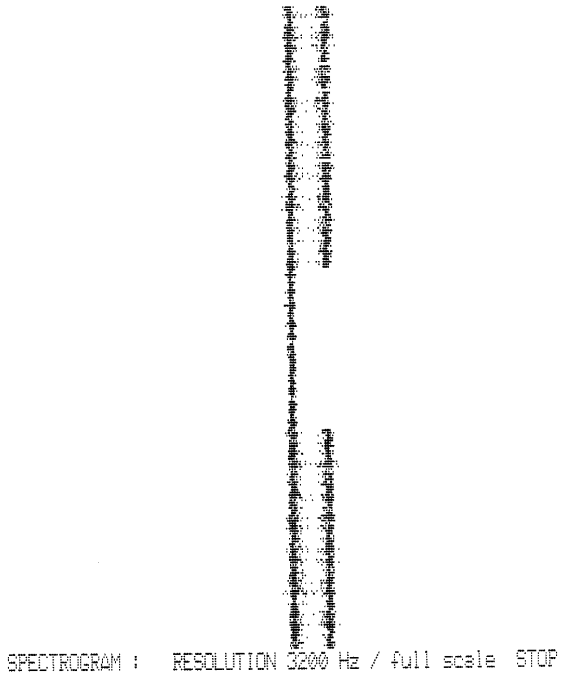
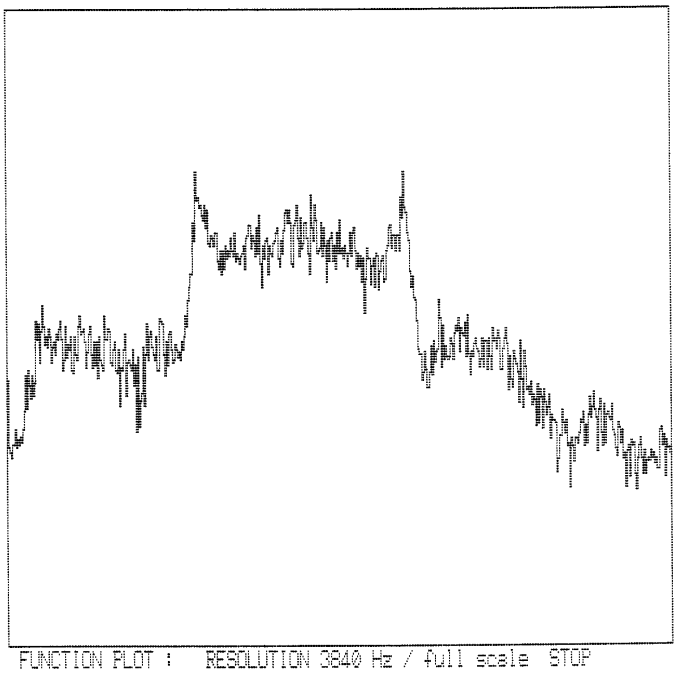


Fig. 5.4.- FSK RTTY signal spectrum: LOG vertical scale (above) and intensity spectrogram (below).



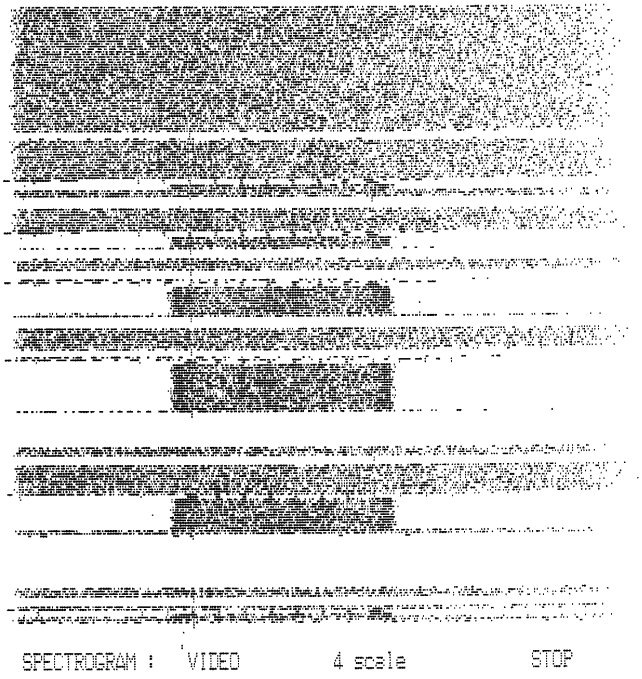
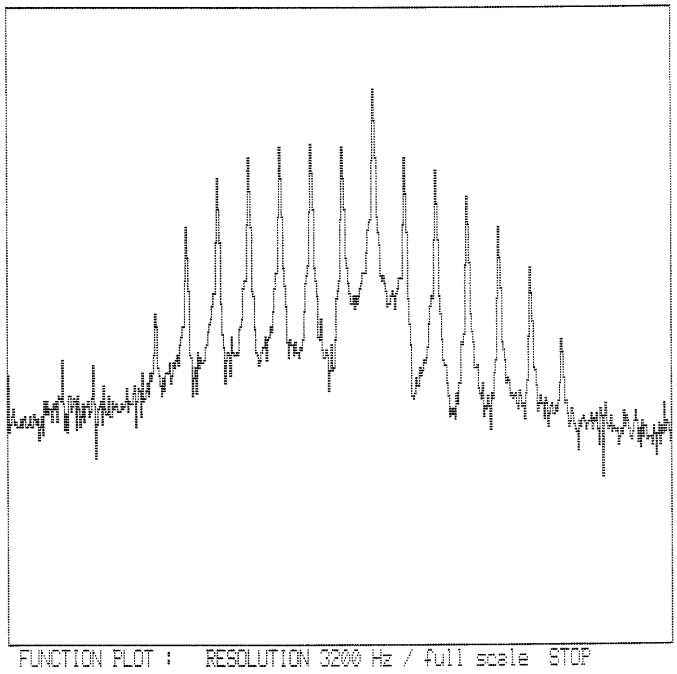


Fig. 5.5.- AFSK 1200bps packet-radio signal spectrum: LOG vertical scale (above) and spectrogram (below).



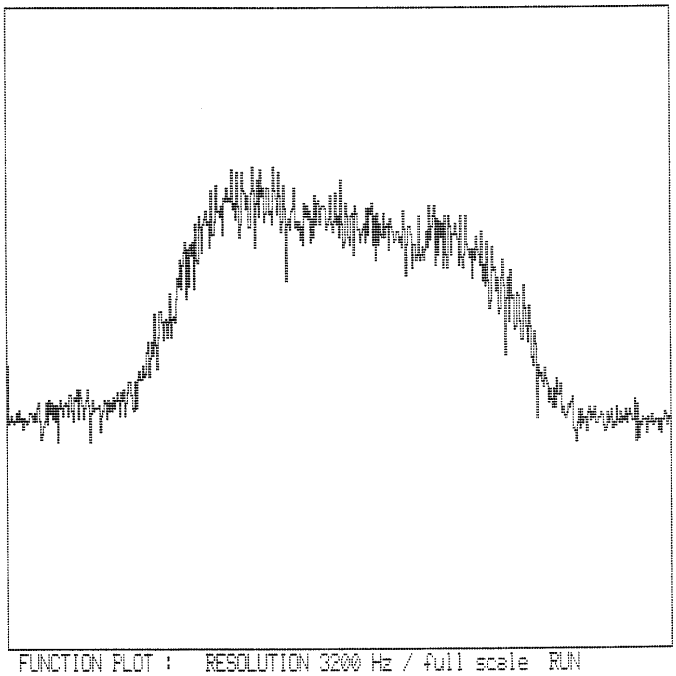


Fig. 5.6.- 1200bps satellite PSK packet-radio transmission: flags (above) and data (below).

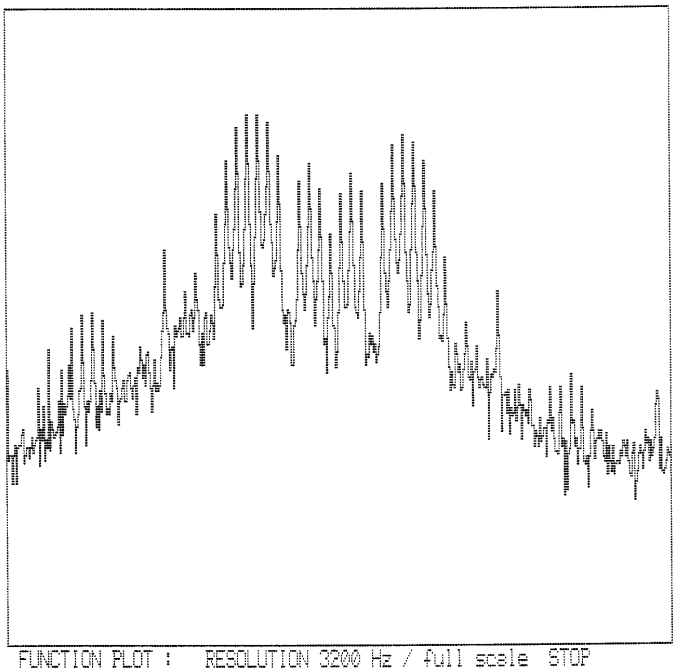
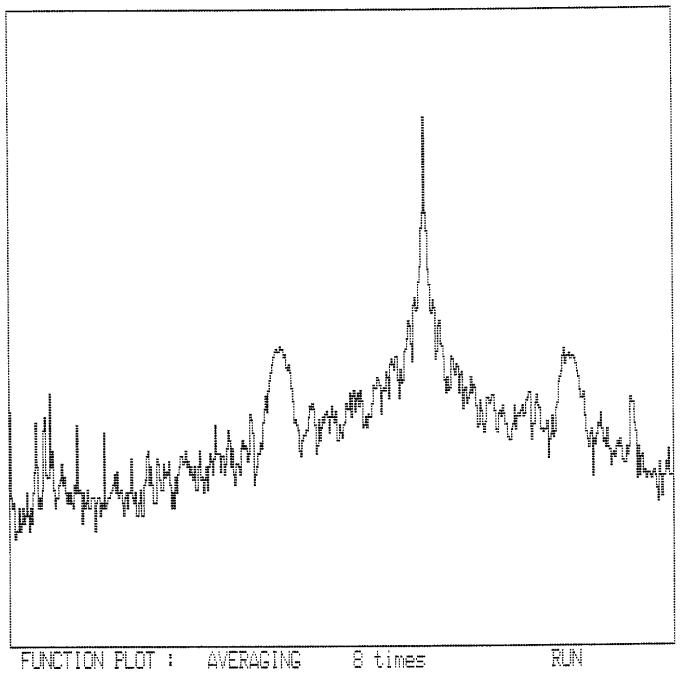


Fig. 5.7.- AO-13 400bps BPSK telemetry spectrum: LOG vertical scale (above) and intensity spectrogram (below).



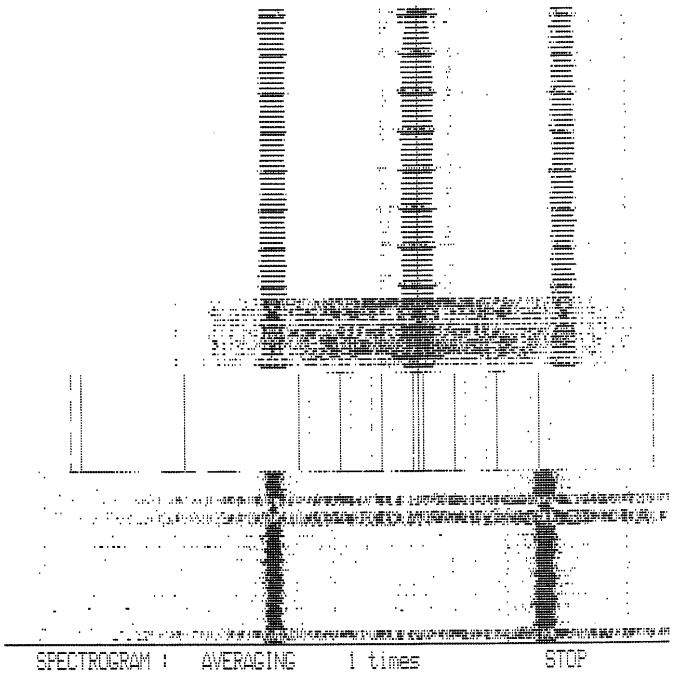
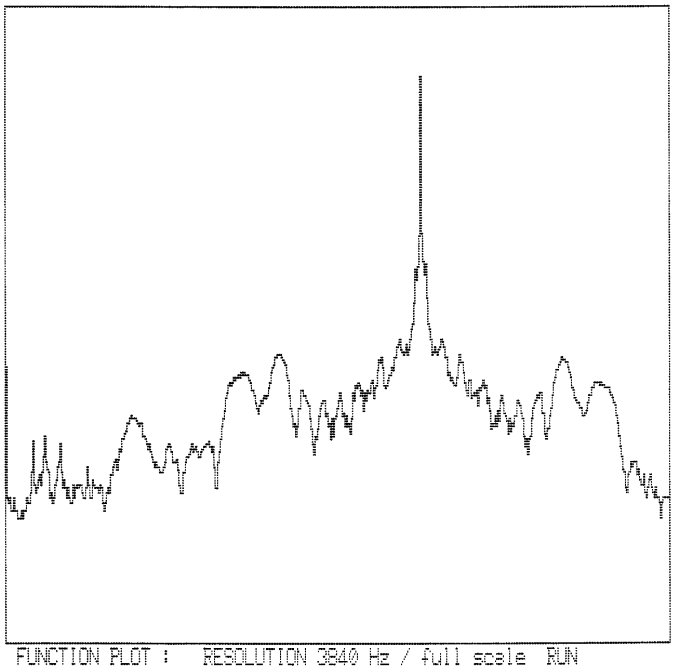


Fig. 5.8.- Meteosat WEFAX signal spectrum: LOG vertical scale (above) and intensity spectrogram (below).



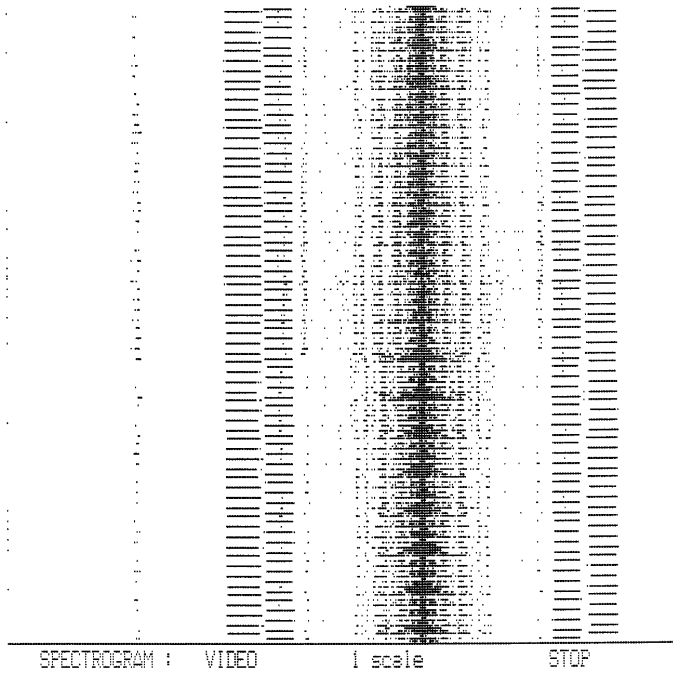
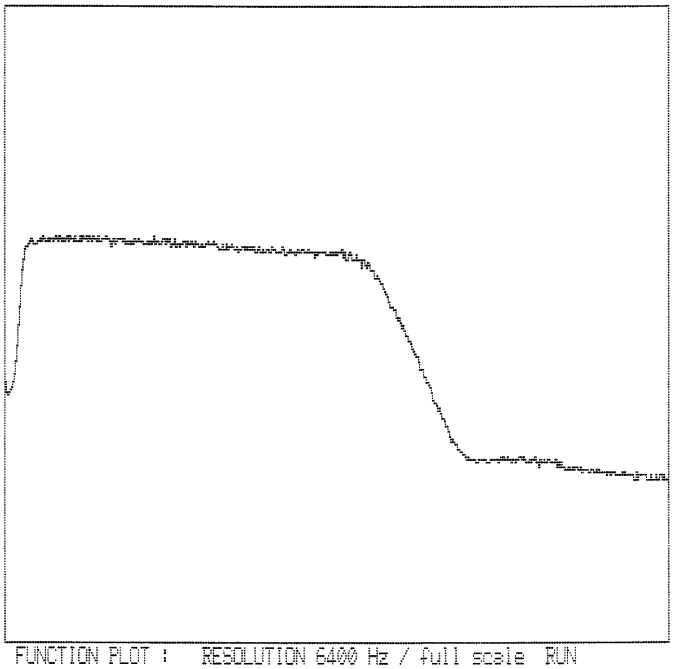


Fig. 5.9.- NOAA-10 APT signal spectrum: LOG vertical scale (above) and intensity spectrogram (below).

```
*** IMPLEMENTED COMMANDS ***

  +    Increment parameter
  -    Decrement parameter

  A    Averaging: factor
  B    Border: grey-level
  D    Divisions: grey-level
  E    Eraser: grey-level
  F    Function display
  G    Gain: A/D converter
  H    Hardcopy: create file
  I    Ink: grey-level
  O    Overlap: ON / OFF
  P    Paper: grey-level
  R    Resolution: full scale
  S    Spectrogram display
  T    Trigger: RUN & STOP
  V    Video: gain steps
  W    Weighting: raised-cosine ON / OFF

  *    Set default parameters

 FF    Clear screen & Update border, divisions, paper

 CR    Exit

*** Press any key to exit!
```

Fig. 6.1.- Help menu of the FFT spectrum-analyzer program showing all implemented commands.



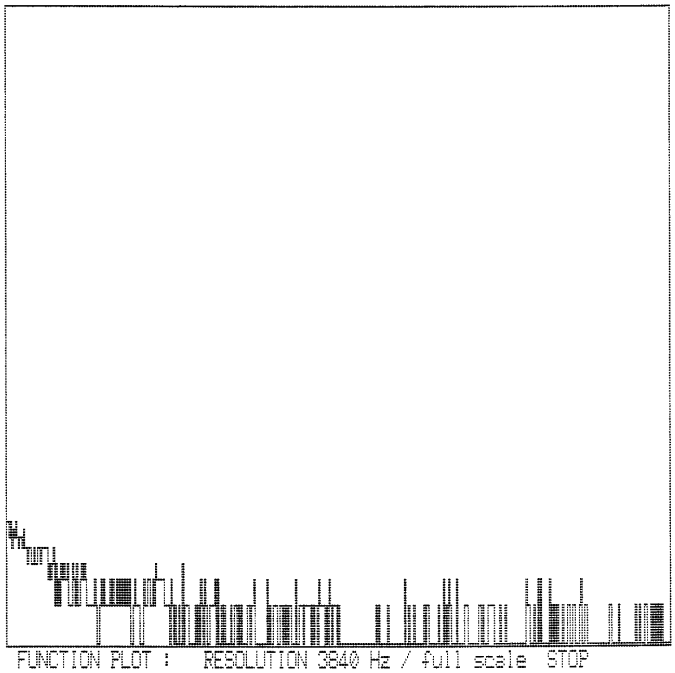


Fig. 6.2.- TP3040 input filter passband (above) and MK5156 A/D coverter noise with no input signal (below).