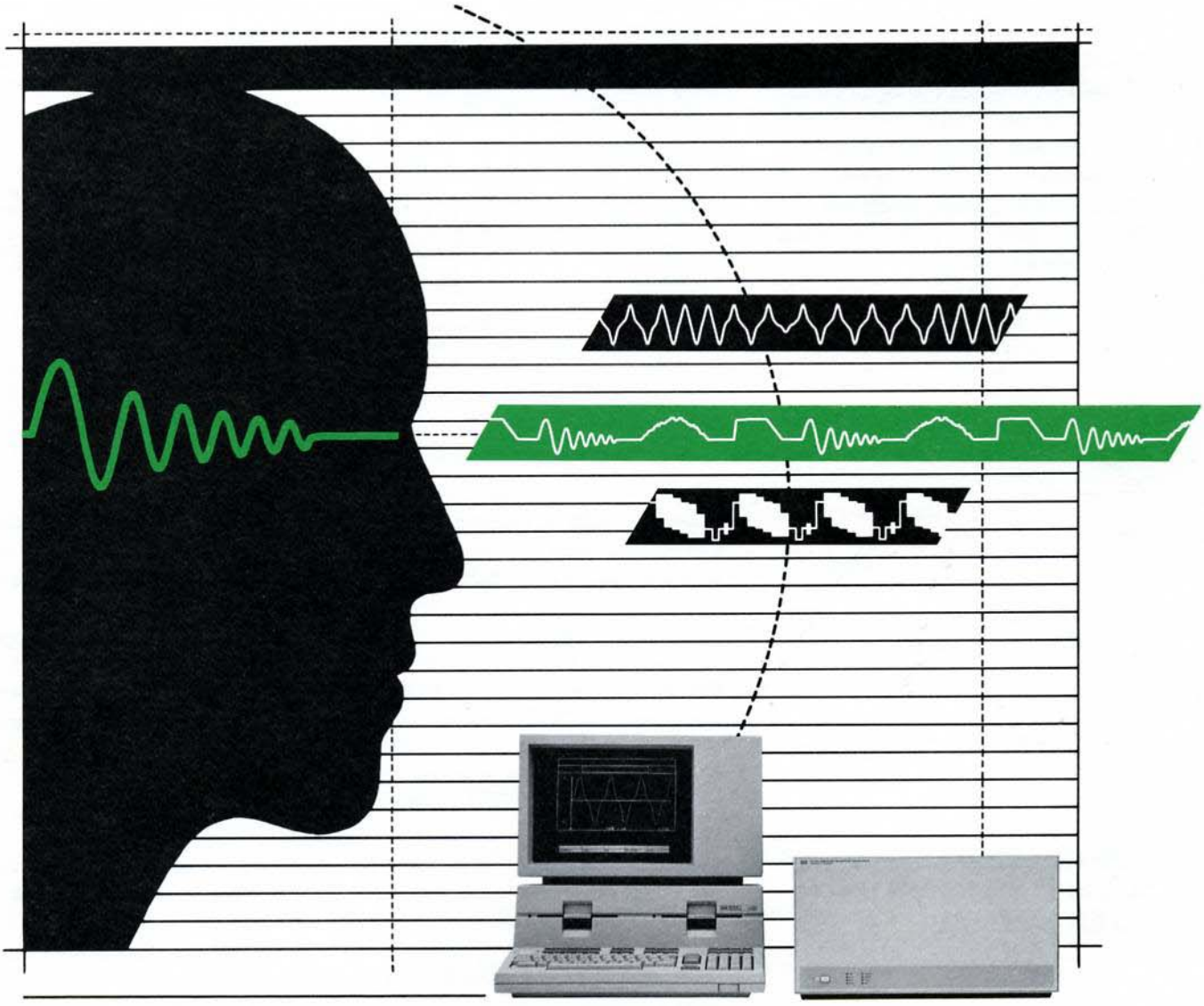
APPLICATION NOTE 314-1

Receiver Testing with the HP 8770S Arbitrary Waveform Synthesizer System



HEWLETT PACKARD

Receiver Testing with the HP 8770S Arbitrary Waveform Synthesizer System

INTRODUCTION

This application note describes the use of the HP 8770S to test receiver video and IF sections. It includes some examples of complex receiver test waveform creation and modification. General familiarity with the HP 8770S is assumed. If you have not already been introduced to the system, reading the data sheet and product note will be helpful (available at no charge from your local HP sales and service office; ask for HP literature numbers 5954-6355 and 5954-6360).

Ideal vs. "Real-life" Signals

Figure 1 shows a simplified block diagram of a typical receiver. A received signal is processed through the different stages of a receiver, each introducing various amounts of distortion and noise to the information signal. To characterize the performance of the video post-detection and processing circuits, the receiver must be exercised with complex test signals simulating the actual operating, or "real-life", environment. Testing receivers under "real-life" conditions provides a high degree of confidence that the receiver will operate as expected in actual use.

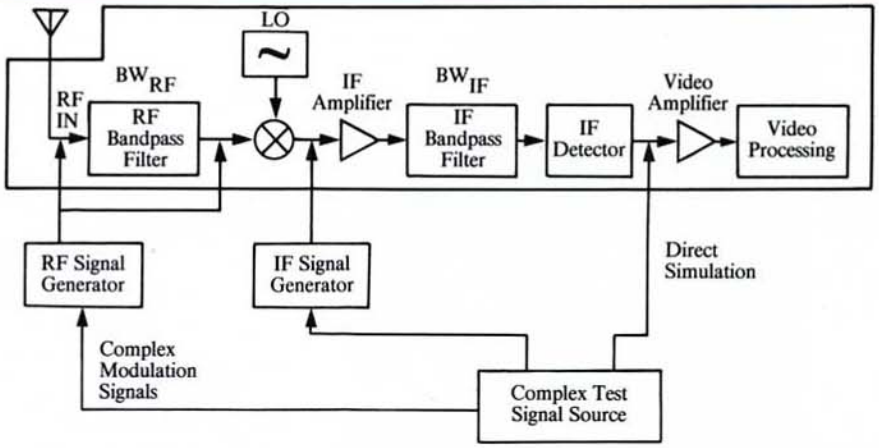


Figure 1. A complex test signal source, such as a high-bandwidth arbitrary waveform generator, can provide "real-life" signals to test the video section of typical receivers. These signals can also drive signal generator modulators to test the RF and IF sections.

Currently, racks of equipment containing RF sources, modulators, modulation sources, and noise sources may be required to accomplish this "real-life" signal generation task. The investment of time and money in these types of systems is high.

An arbitrary wavefrom generator is a source of complex receiver test signals, eliminating the need for racks of equipment in specialized generation systems. Up until now, however, arbitrary wavefrom generators have been limited by digital-to-analog converter (DAC) technology; they simply didn't have the frequency range to match receiver testing needs. In addition, small memory and limited front panel functions have been major drawbacks.

THE HP 8770S ARBITRARY WAVEFORM SYNTHESIZER SYSTEM

The HP 8770S provides the capability needed to test receivers with "real-life" signals. The HP 8770S consists of the HP 8770A Arbitrary Waveform Synthesizer, the HP 11775A Wavefrom Generation Software, and an HP Series 200 Model 216A or 236A technical computer. The system software features the Waveform Generation Language (WGL), enabling the user to create and modify waveforms as easily as using a calculator to work with numbers. Complex test signals can be constructed with simple operations, and then data representing these signals downloaded to the HP 8770A for generation.

With state-of-the-art gallium arsenide technology, the HP 8770S outputs waveform samples at a 125 MHz rate, giving signal generation bandwidths up to 50 MHz. It has a 12 bit, 128K word waveform sample memory for accurate simulation of receiver test signals. It also has a separate sequencer memory. "Packets" of waveform data can be computed with WGL and stored in arbitrary order in the waveform memory. The sequencer memory is then used to determine the order in which the corresponding waveforms will be generated, including the number of times each waveform is repeated.

A Powerful Source of "Real-Life" Signals for Receiver Testing

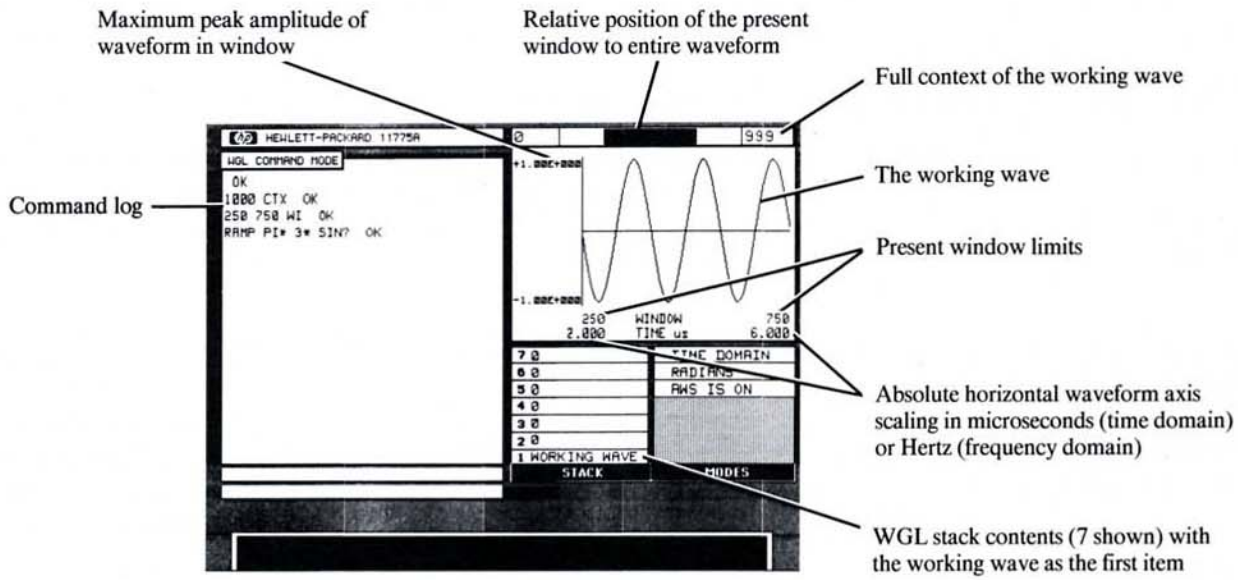
With its dc - 50 MHz coverage and 12 bit resolution, the HP 8770S can accurately generate test signals for all stages of a modern receiver. The IF and video sections can be directly tested. To test the RF sections, the system can generate complex modulation signals that drive RF and IF signal generators. In some cases, the complex signals might be directly upconverted to the test band.

As the HP 8770S simulates precise error conditions and distortion, it verifies the operation and susceptibility of the receiver detection circuitry.

GENERATING RECEIVER TEST SIGNALS WITH THE HP 8770S

In the following sections, waveform creation and generation examples show how WGL commands entered into the computer produce standard and complex receiver test waveforms with the HP 8770A. WGL has over 180 commands available for waveform development. Only 37 commands are required to develop all the waveforms in this note, giving an indication of the power of the language. A glossary of the WGL commands used in the examples is provided for convenient reference (see Appendix). **Figure 2** shows the WGL computer display and references some of the common WGL terms.

Figure 2. The typical HP 8770S waveform-creation computer display.



Generation of pulse waveforms, mainly for video section testing, are covered first. Complex signals for IF section testing, such as multitone, AM, FM, and "hopped" signals, are covered next. The note closes with a discussion on creating customized complex signal WGL routines for specific applications.

Pulse Waveform Generation

To characterize the post-detection circuitry of a pulsed-wave system, a pulse waveform with independently variable parameters must be simulated. Using WGL, pulse waveforms with arbitrary parameters can be developed quickly. The receiver can then be exercised under non-ideal conditions by adding quantifiable amounts of noise, glitches, and other distortion to the test signal.

In the following examples, WGL is used to construct typical pulse test signals with variable pulse width, rise/fall time, PRI, and variable signal-to-noise ratio.

A Pulse Train

Pulse trains with variable parameters can be designed using WGL. For example, an application may require a pulse waveform with the following pulse parameters:

Risetime (0-100% level): 40 ns
Pulse width (100% level): 200 ns
Falltime (0-100% level): 60 ns
PRI: 2 μs (PRF= 500 kHz).

The HP 8770A's internal waveform sample rate (125 MHz) results in 8 ns timing resolution, i.e., each waveform sample point generated comes at an 8 ns time interval. Since the period of the desired pulse train is 2 μs, one cycle of the waveform can be defined with 250 waveform elements, or "points". The pulse waveform in **Figure 3** is constructed by using the number of points required to construct each section of the pulse. The following WGL commands are entered in the computer to create the pulse train.

	250 CTX	Set the number of points required for 1 cycle of the pulse.
Rise time	**0 5 WINDOW?**	5 points are required for the 40 ns risetime. The question mark after the command updates the WGL graphical display screen.
	RAMP?	Generate a linear ramp to simulate the risetime.
	1+?	Add one to place the ramp between 0 and 2.
	NORM?	Normalize the maximum amplitude to 1.
Pulsewidth	**5 30 WINDOW?**	Set the window to the next 25 points (200 ns).
	1 LOAD?	Load a constant amplitude of 1.

Falltime

Since the desired falltime is 60 ns and the timing resolution is 8 ns/point, 7 points are used resulting in a 56 ns falltime. For precise timing resolution requirements, an external sampling clock can drive the HP 8770A to adjust the resolution. For example, a 100 MHz external clock will provide 10 ns resolution.

30 37 WINDOW?	Select 7 points for a 56 ns falltime.
RAMP REFL?	Reflect a ramp to generate the falltime.
1 + NORM?	Offset the ramp and normalize the amplitude to 1.

Interpulse period

37 249 WINDOW?	Set the window to the remaining portion of the waveform.
0 LOAD?	Load a constant level at 0.
FULL?	A look at the whole waveform.
STORE A?	Save the waveform in a temporary register for later use.

The internal memory architecture of the HP 8770A requires the total length of individual waveform packets stored in the memory to be a multiple of 8. Most complex waveforms are constructed by piecing together different waveforms of various lengths. Each waveform part can have any arbitrary length; only the total waveform must have a length which is a multiple of 8. The pulse train waveform in the example is constructed with 250 points. Since 250 is not a multiple of 8, the solution is to construct 4 cycles of the waveform, which corresponds to 1000 points (a multiple of 8), according to the following WGL commands.

1000 CTX?	Set the number of points for four cycles of the pulse waveform.
250 CLONE?	Copy the first 250 points to obtain four cycles of the pulse.
DOWNLOAD	Download the waveform data to the HP 8770A memory. The computer will prompt for a file name under which to store the waveform, and for an amplitude scale factor.
GO	Generate the pulse waveform.

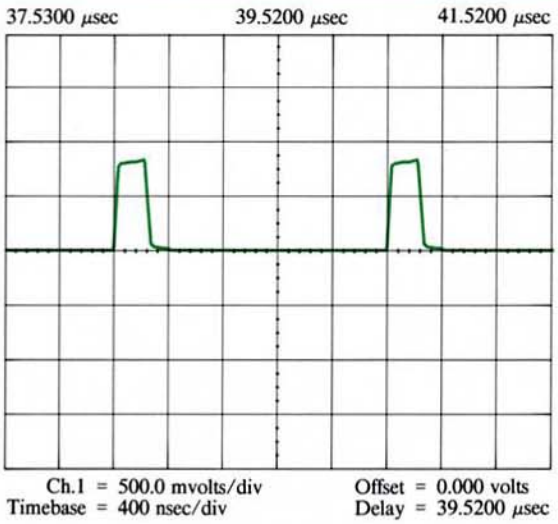


Figure 3. A 500 kHz pulse train with 40 ns risetime, 200 ns pulsewidth, and 56 ns falltime, created and generated with the HP 8770S.

Common waveforms, such as the pulse waveform in **Figure 3**, can be easily created using the FCNGEN program which is located on an application program disc included with the HP 11775A Waveform Generation Software. The FCNGEN program allows the user to generate sine, pulse, AM, FM, and other commonly used receiver test signals. This program is written in WGL, and its functions are explained

in the "CREATING APPLICATION PROGRAMS" section of this note. This program generates common signals by prompting the user for waveform parameters (frequency, modulation rate/depth, etc.), and then creating the data that produces the specified signal.

A Noisy Pulse Signal

Receivers seldom encounter clean pulse signals such as the ones created in the last example. Therefore, it is important to exercise the post-detection circuitry under more marginal conditions by adding precise amounts of noise to the test signal. Creating noisy signals with WGL is simple and fast. The command **NOISE** generates a random noise waveform over a given time span. This waveform can be scaled and modified to simulate a desired effect, and then be added to other waveforms. In the next example, the pulse train waveform of **Figure 3** is used as a base on which to construct a noisy pulse signal.

The period of this pulse train is 2 μs (250 points). To simulate this pulse signal contaminated with random noise, the period of the noise must be longer than the period of the pulse train. In the following example, the repetition rate of the noise signal is designed to be 1/8 the repetition rate of the pulse signal (noise period=16 μs corresponding to 250x8=2000 points). This simulates pseudorandom noise added to 8 cycles of the pulse train. The following WGL commands are entered to achieve the desired signal.

0 249 WINDOW A?	Recall the pulse waveform from register A.
2000 CTX?	Set up the noise context.
250 CLONE?	Create 8 cycles of the pulse waveform.
STORE A	Save the new pulse waveform.
NOISE?	Generate noise over 2000 points.
8/?	Scale the noise waveform (12.5% of full scale in this example).
A+?	Add the noise waveform to the pulse train.
DOWNLOAD	Download the waveform data to the HP 8770A memory.
GO	Generate the signal.

Figure 4 is an oscilloscope display and **Figure 5** is a spectrum analyzer display of the noisy pulse signal. The period of the noise can be varied if a more random signal is required. For the pulse signal shown in **Figure 6**, the period of noise is increased to 32 μs, doubling the number of noise spectral components.

Figure 4. The pulse train of **Figure 3** is contaminated by adding an arbitrary random noise waveform with WGL.

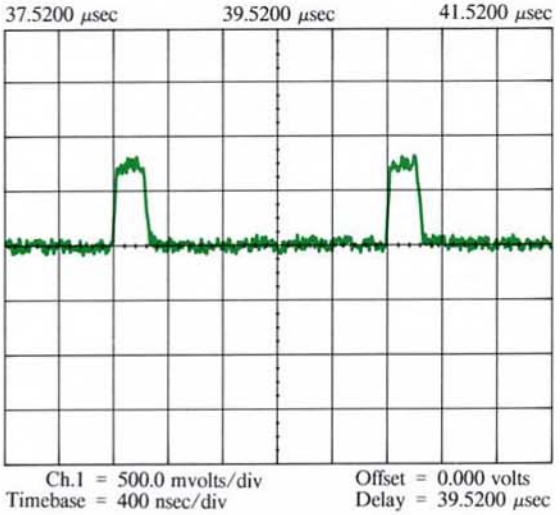


Figure 5. A spectrum analyzer display of the noisy pulse signal of **Figure 4**. The spacing of the noise spectral components is determined by the 16 μs noise waveform period.

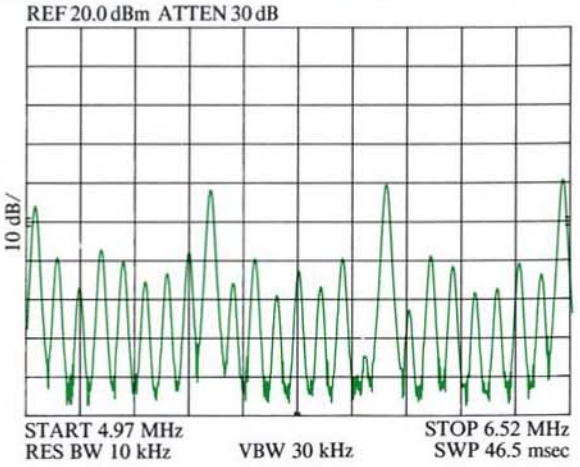
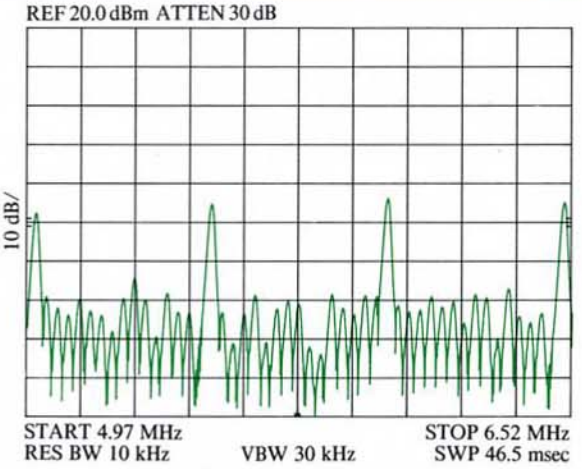


Figure 6. The pulse train signal of **Figure 3** is added to a 32 μs period noise waveform. The noise spectral components are doubled in number due to the decrease in repetition rate of the noise waveform.



Pulse Stagger

Simulation of pulse trains with varying PRF is a difficult task using traditional signal generators. The HP 8770S creates such signals easily.

The signal in **Figure 7** is an arbitrary pulse train composed of three pulses with varying shapes. This signal is designed in three steps as shown in **Figure 8.**

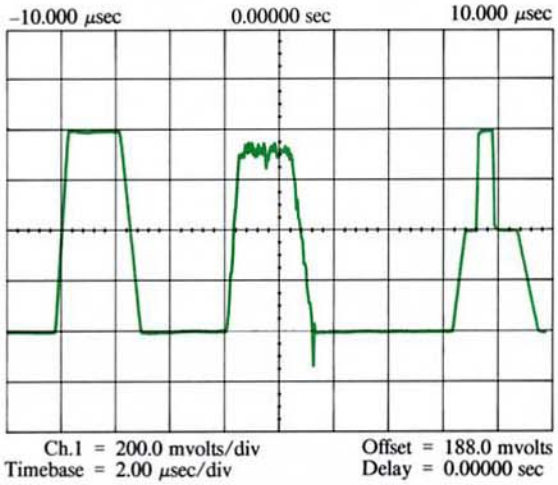


Figure 7. A staggered pulse waveform. The variable interpulse period is controlled by the sequencer program created with WGL and stored in the HP 8770A.

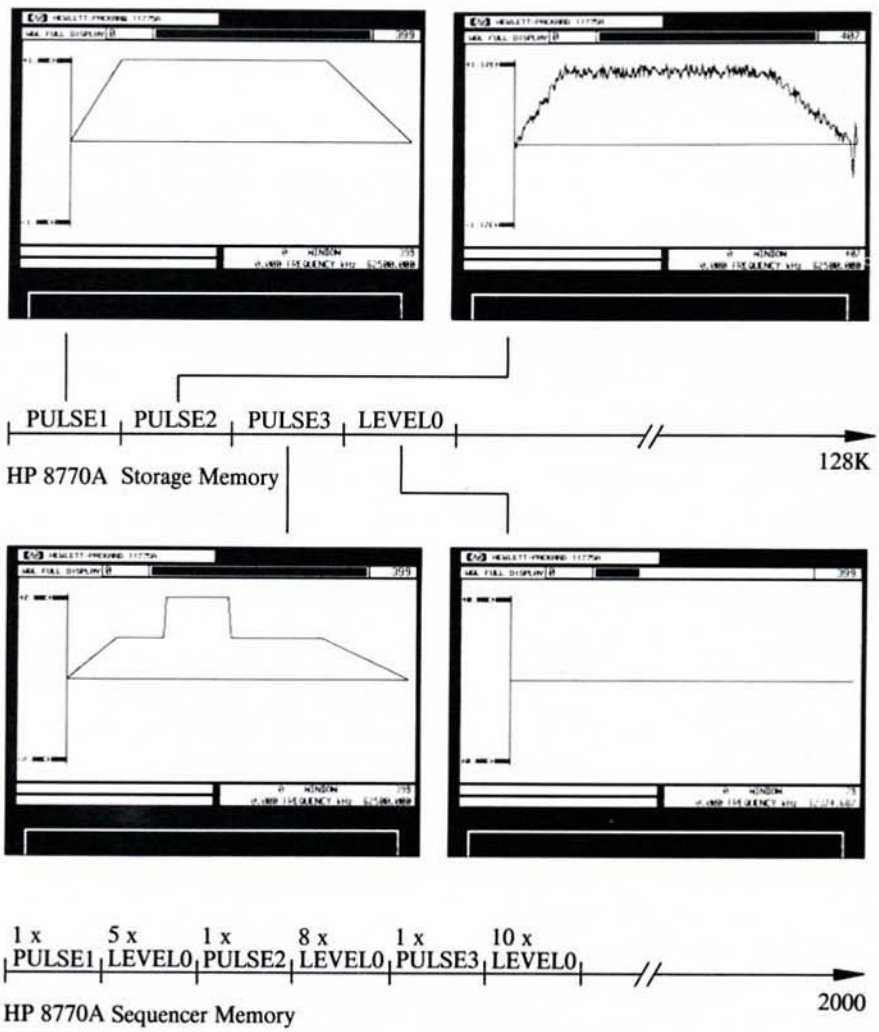


Figure 8. Individual waveforms are stored in the HP 8770A memory with assigned names. The synthesizer generates the signal according to the program in the sequencer.

First, the three pulses (not including the interpulse period) are stored in the memory with assigned waveform names **PULSE1**, **PULSE2**, and **PULSE3**.

The first pulse is created by entering the following WGL commands.

400 CTX	Set the pulse context.
0 60 WINDOW?	Set a .48 μs risetime window.
RAMP 1+ NORM?	Create the risetime.
60 300 WINDOW?	Set the pulsewidth (1.92 μs).
1 LOAD?	Load a constant level.
300 399 WINDOW?	Set a .8 μs falltime window.
RAMP 1+ REFL NORM?	Create the falltime.
FULL STORE A?	Store the pulse waveform.
"PULSE1;" $DOWNLOAD	Store **PULSE1** in the HP 8770A memory. The **$DOWNLOAD** command takes the items in quotes just before it as the waveform file name and the amplitude scale factor.

The second pulse is constructed by adding an arbitrary noise waveform and a glitch to **PULSE1**.

400 CTX	Set the noise context.
NOISE 8/?	Compute and scale the noise waveform.
A+?	Add the noise to **PULSE1**.
408 CTX?	Increase the context to add an arbitrary glitch.

The glitch is constructed by using the **STOREIN** command to add arbitrary amplitude levels point-by-point.

```
-.2 400 STOREIN
-.5 401 STOREIN
-.4 402 STOREIN
-.1 403 STOREIN
.3 404 STOREIN
.2 405 STOREIN
0 406 STOREIN
0 407 STOREIN
```

FULL?	View the complete **PULSE2** waveform.
"PULSE2;" $DOWNLOAD	Store **PULSE2** in the HP 8770A memory.

Applications such as coincidence circuit testing in radar receivers and simulating special codes in communication receivers require pulse signals with multiple levels. The windowing capabilities of WGL simplify construction of such waveforms.

PULSE3 is constructed by adding a .6 μs pulse to **PULSE1**.

400 CTX A?	Recall **PULSE1.**
115 189 WINDOW?	Set up a .6 μs window on an arbitrary portion of **PULSE1**.
2 LOAD?	Set a constant level twice the level of **PULSE1**.
FULL?	View the 2-level pulse.

The sharp transition between levels may cause ringing on the generated signal because of the HP 8770A synthesizer's 50 MHz upper frequency limit. To lessen this effect, the transitions can be "softened" by windowing in and adding ramps to simulate linear rise and fall times.

113 115 WINDOW?	Set up a window for the risetime.
RAMP .5* 1.5+?	Compute a ramp transition between level 1 and 2.
STORE B	Store the risetime for the next step.
189 191 WINDOW?	Set up a window for the falltime.
B REFL?	Reflect the rising ramp to simulate the falltime.
FULL?	View the complete **PULSE3** waveform.
"PULSE3;" $DOWNLOAD	Store **PULSE3** in the HP 8770A memory.

Next, a waveform which represents a small section of the interpulse period, called **LEVEL0**, is stored in the memory.

0 79 WINDOW	Set a window for 80 points.*
0 LOAD?	Load the 0 level.
"LEVEL0;" $DOWNLOAD	Store **LEVEL0** in the HP 8770A memory.

* The minimum waveform length that can be stored in the waveform memory is 64 points. 80 points is arbitrarily chosen here.

The final step in creating this pulse stagger signal is downloading a sequencer program which, when executed, will generate the desired pulse, repeat the waveform **LEVEL0** enough times to make up the interpulse period, then go on to the next desired pulse. By using the sequencer, only a small portion of memory is required to generate a pulse train with a long period. Sequencer programs are constructed by specifying the waveform name, the number of repetitions, and the packet advance information in the following format:

NEWSEQ	Initiates the sequencer program.
"Waveform name; # of repetitions; packet advance" $PACKET	

This command defines a packet that is stored in the sequencer memory as an instruction determining how the HP 8770A waveform memory will be accessed. According to these stored instructions, a waveform will be accessed and generated for the specified number of repetitions before an advance to the next waveform packet. "Packet advance" is triggering information that the sequencer uses for switching between packets. When this is set to **AUTO**, the sequencer switches to the next packet after completing the required number of repetitions.

The staggered pulse train in **Figure 7** is generated using the following sequencer program.

NEWSEQ	Initiate sequencer program.
"PULSE1;1;AUTO" $PACKET	Generate **PULSE1**.
"LEVEL0;5;AUTO" $PACKET	Set the interpulse period= 3.2 μs (5 x 80 points x 8 ns/point).
"PULSE2;1;AUTO" $PACKET	Generate **PULSE2**.
"LEVEL0;8;AUTO" $PACKET	Set the interpulse period= 5.1 μs.
"PULSE3;1;AUTO" $PACKET	Generate **PULSE3**.
"LEVEL0;10;AUTO" $PACKET	Set the interpulse period= 6.4 μs.
GO	Execute the sequencer program.

Other complex pulse modulations, such as PRI FM, and jitter, can be simulated with the HP 8770S using the sequencer techniques demonstrated in this example.

Pulse Jitter Simulation

The sequencing capability of the HP 8770S can be used to create complex signals with variable timing jitter. In the following example, a Gaussian pulse train waveform, consisting of 4 pulses, is designed and stored as **GAUSS1**. The third pulse in the train is then shifted in time by an arbitrary amount to the right of its original position. This second waveform is stored as **GAUSS2**. **GAUSS3** is stored with the third pulse shifted symmetrically in the opposite direction. Finally, a sequencer program is designed to generate each waveform for a number of times before going on to the next waveform. Since the third pulse is shifted in each waveform, it jitters between three positions.

The Gaussian pulse is generated using the general equation:

$$f(x)=e^{-Ax^2},$$

where A is a constant scale factor.

These WGL commands are used to generate the pulse waveforms.

1000 CTX	Set an 8 μs period for the pulse signal.
0 LOAD?	Clear the context of waveform data.
0 99 WINDOW?	The duration of the Gaussian pulse is arbitrarily set to .8 μs.
RAMP SQ?	Create the square term.
4* NEG?	An arbitrary scale factor is used (A=4).
EXP?	Create the Gaussian pulse.
FULL?	Set the window to the full context.
250 CLONE?	Create 4 Gaussian pulses.
STORE A	Store **GAUSS1** in register A for later use.
"GAUSS1;" $DOWNLOAD	Store **GAUSS1** in the HP 8770A memory.
424 674 WINDOW?	Set a window on the third pulse.
20 RIGHT?	Shift the pulse 20 elements (160 ns) to the right.
FULL?	View the full **GAUSS2** waveform.
"GAUSS2;" $DOWNLOAD	Store **GAUSS2** in the HP 8770A memory.
A 424 674 WINDOW?	Recall **GAUSS1** from A. Set a window on the third pulse.
20 LEFT?	Shift the pulse 20 elements to the left.
FULL?	
"GAUSS3;" $DOWNLOAD	Store **GAUSS3** in the HP 8770A memory.

The following sequencer program instructs the synthesizer to generate each waveform 4000 times. Thus, each pulse has a generated period of 3.2 ms (4000 x .8 μs).

```
NEWSEQ
"GAUSS1;4000;AUTO" $PACKET
"GAUSS2;4000;AUTO" $PACKET
"GAUSS3;4000;AUTO" $PACKET
GO
```

The resultant signal is a Gaussian pulse train from the HP 8770A output with each third pulse jittering.

Complex Signal Simulation in the Frequency Domain

WGL is an effective tool for analysis and creation of waveforms in the frequency domain. Complex analog signals can be described by specifying their frequency spectrum. The spectrum can be modified to reflect channel impairments, band limitation, and other distortion and non-ideal effects. Then, by using the inverse FFT or DFT, the time domain waveform data is calculated and can be downloaded to the HP 8770A for generation.

Multi-tone signals with variable amplitude and phase relationships can be constructed and modified with a few WGL commands. Using conventional methods to obtain multi-tone signals often results in a complex system with multiplexed sources. As test requirements change, simulation of new environments becomes more expensive. The HP 8770S provides a flexible solution with WGL available to create new environments without new hardware having to be bought or built. Hundreds of independent tones can be developed with software and generated by the HP 8770A.

The following example shows how a multi-tone signal with frequency components at 8, 20, 34, 42, and 50 MHz is generated.

Working in the Frequency Domain

WGL commands work in both the time and frequency domains. Since the WGL FFT and DFT routines require amplitude and phase information, both an amplitude and a phase waveform are defined in the frequency domain. Arbitrary amplitude and phase waveforms are computed using two separate waveform axes. Once these waveforms are computed, the data can be transformed to the time domain.

When working in the frequency domain, the frequency resolution depends on the selected context size.

Frequency resolution = 62.5 MHz / (Context-1)

To generate the multi-tone signal in the following example, a context size of 513 is selected.

FDOMAIN	Set up the frequency domain axis.
513 CTX?	Set context to 513 elements (spectral components).
0LOAD?	Clear the context of waveform data.
DISP2?	Display the two waveform axes.

This context corresponds to a minimum frequency resolution of 122 kHz (62.5 MHz/512). When the frequency information is transformed to the time domain, the time waveform is described by 1024 points.

The **STOREIN** command is used to specify each tone; the amplitude of each is arbitrarily set to a relative level.

Amplitude waveform

1 8E6 HZ .5+ INT STOREIN	Store level 1 at 8 MHz.
.8 20E6 HZ .5+ INT STOREIN	Store level .8 at 20 MHz.
.2 34E6 HZ .5+ INT STOREIN	Store level .2 at 34 MHz.
.4 42E6 HZ .5+ INT STOREIN	Store level .4 at 42 MHz.
.6 50E6 HZ .5+ INT STOREIN	Store level .6 at 50 MHZ.
FULL STORE A?	Save the spectral lines.

The command **HZ** returns the frequency axis element that corresponds to the specified frequency. If the minimum frequency resolution prevents the exact spectral element from being defined, **HZ** will return a real number. For example, 8 MHz corresponds to the 65.536th sample point. To define the nearest whole spectral component, 65.536 must be rounded off. Since the command **INT** returns only the integer portion of a real number, .5 is first added. Adding .5 to 65.536 and taking the integer portion results in 66. So, point 66 is the closest spectral position to 8 MHz with the given resolution. The actual frequency is 8.05664 MHz, due to the 122 kHz resolution set by the 513-point context.

Phase waveform

The phase information can be specified using the same commands that specified the amplitude information. For this example, however, the **NOISE** command is used to assign random phase to each spectral component.

XY?	Use the auxilary waveform axis to specify the phase information.
NOISE?	Generate random amplitude levels between 1 and –1.
A*?	Multiply the amplitude waveform by the random waveform.
PI*?	Transform random amplitude levels to random phase information.
XY?	Bring the amplitude information back to the main axis.

Multiplying the noise waveform by the amplitude waveform results in 5 random levels corresponding to the 5 tones. Multiplying this waveform by **PI** results in 5 random phase coefficients. Before transforming the data to the time domain, the amplitude information must reside in the main waveform axis. This is the reason for the last command (**XY**). **Figure 9a** is the HP 8770S computer display of the frequency domain axes.

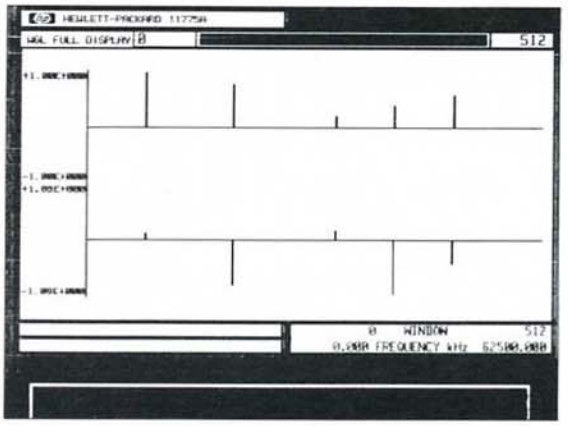


Figure 9a. The HP 8770S computer display of the frequency domain amplitude and phase axes. 5 independent tones are created with arbitrary amplitude (upper axis) and phase information.

Computing the time waveform

TOTIME?	Take the inverse **FFT***.
DISP1?	Display the main axis.
NORM?	Normalize the time waveform.
DOWNLOAD	Download the waveform to the HP 8770A.
GO	Generate the multitone signal (**Figures 9b** and **10**).

When the frequency information is transformed to the time waveform, the auxilary axis is cleared. **DISP1** changes the dual display back to the single axis display. **Figure 9b** illustrates how precisely the WGL signal simulation compares with the actual signal generated by the HP 8770A.

*When using the Fast Fourier Transform, the number of points describing the waveform must be an integral power of 2 in the time domain, or an integral power of 2 plus 1 in the frequency domain. If this condition is not met, WGL will automatically use a Discrete Fourier Transform (DFT) routine to compute the transform. The speed of the FFT for transforming 1024 points is 2 seconds. The Discrete Fourier Transform takes considerably longer.

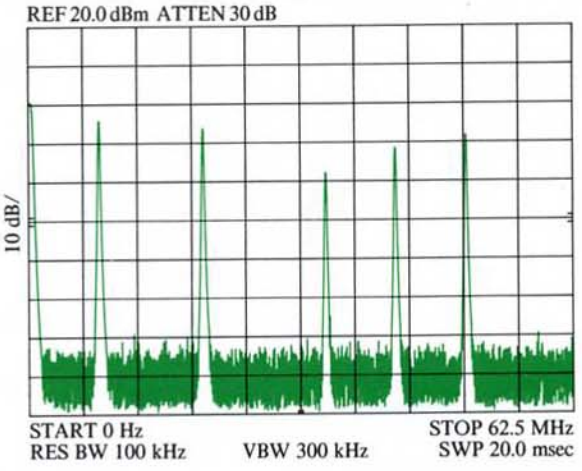


Figure 9b. A spectrum analyzer display of the 5-tone signal created and generated by the HP 8770S. This is a logarithmic display. The WGL display of the multitone signal (**Figure 9a**) is a linear display.

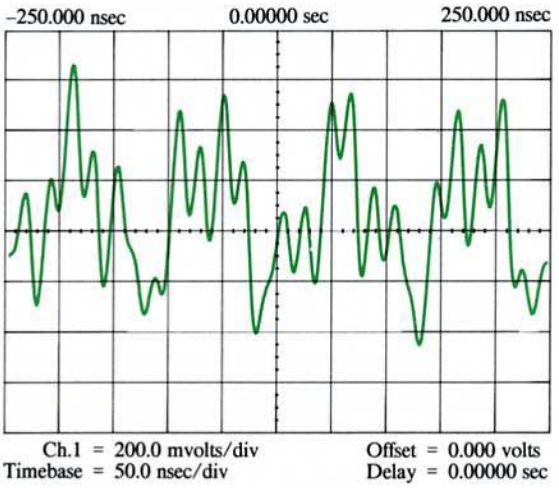


Figure 10. An oscilloscope display of the 5-tone signal generated by the HP 8770S.

Sine Wave Modulation

Amplitude and frequency modulated waveforms can be constructed by simple waveform operations. As the HP 8770S is designed for arbitrary signal generation, traditional function generator signals can be modified into more complex signals. The following examples illustrate how WGL constructs AM and FM signals.

AM signal

To generate an AM signal, the carrier and modulation signal must each be computed. In this example, a 20 MHz carrier signal will be modulated with a 1 MHz sinusoidal signal with a modulation index of .5.

f_c=20 MHz implies 6.25 points/cycle for 8 ns/point resolution.

f_m=1 MHz implies 125 points/cycle for 8 ns/point resolution.

There are 20 cycles of the carrier signal per cycle of the modulation signal (125/6.25). The WGL commands entered to create this signal follow.

125 CTX?	Set up the modulation period.
RAMP PI* SIN?	Compute the modulation signal.
STORE A?	Store the modulation signal.
RAMP PI* 20* SIN?	Compute the carrier signal.
STORE B?	Store the carrier signal.
A .5*?	Set 50% modulation depth.
B*?	Modulate the carrier.
B+?	Add the modulation term to the carrier.

Before downloading this signal for generation, the context must be increased to a number that is a multiple of 8.

1000 CTX?	Increase the context to 1000 points.

125 CLONE?	Copy the first 125 points across the full context.
DOWNLOAD GO	Generate the AM signal (**Figure 11**).

Figure 12 is a spectrum analyzer display of the AM signal.

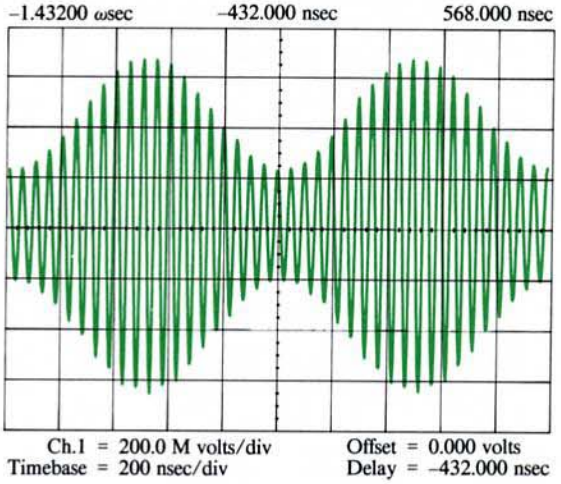


Figure 11. An AM signal with 20 MHz carrier frequency, 1 MHz modulation rate, and 50% modulation depth created and generated by the HP 8770S.

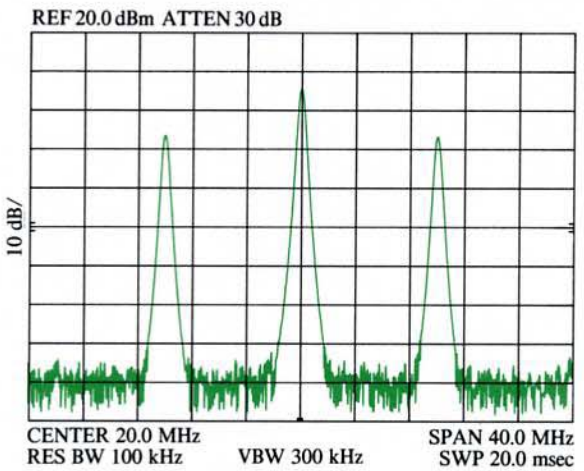


Figure 12. A spectrum analyzer display of the AM signal of **Figure 11**.

FM signal

A wide range of frequency modulated signals can be developed with the HP 8770S. Since arbitrary phase waveforms can be easily created, generation of signals with various frequency characteristics (e.g., linear, square, exponetial, etc.) is possible. In this example, an FM signal will be generated with the following parameters:

f_c=20 MHz implies 6.25 points/cycle for 8 ns/point resolution.

f_m=μ2 MHz implies 62.5 points/cycle for 8 ns/point resolution.

Δf=8 MHz.

The phase of the FM signal is defined by

$\phi(t)=w_c t+(\Delta f/f_m)SIN(w_m t)$,

where w_c is the carrier frequency (2 x pi x f_c),
w_m is the modulation frequency (2 x pi x f_m),
and Δf is the maximum frequency deviation.

A context of 125 points will be used to define the phase for 2 cycles of the modulation (2 x 62.5 = 125) and 20 cycles (125/6.25) of the carrier signal.

125 CTX

RAMP PI* 20* STORE A?	Compute and save the carrier phase ($w_c t$).
RAMP PI* 2* SIN?	Create the modulation signal (SIN($w_m t$)).
4*?	Multiply by $\Delta f/f_m$.
A+ SIN?	Create the FM signal.
1000 CTX 125 CLONE?	Achieve a multiple of 8 context.
DOWNLOAD GO	Generate the FM signal (**Figure 13**).

The frequency spectrum of the FM signal is shown in **Figure 14.**

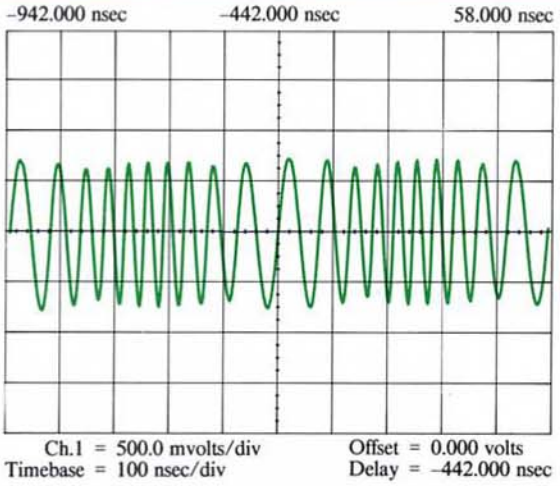


Figure 13. An FM signal with 20 MHz carrier frequency, 2 MHz modulation rate, and 8 MHz frequency deviation created and generated by the HP 8770S.

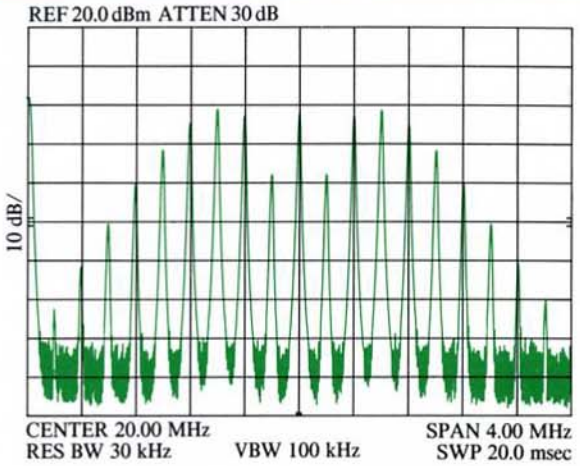


Figure 14. A spectrum analyzer display of the FM signal of **Figure 13**.

Frequency Hopping

The sequencing power of the HP 8770A greatly simplifies the simulation of "hopping" signals. Signal frequency, phase, and/or amplitude can be changed in less than 8 ns over the dc-50 MHz bandwidth. To create a hopping signal, the constituent waveforms are first computed and stored in the memory. Then, a sequencer program is used to define the hop from one waveform to the next.

The **FREQ** command can be used to generate the desired sine frequency. Two pieces of information are required to compute sinusoidal signals using the HP 8770S: (1) the number of cycles, and (2) the number of points to describe the waveform. The **FREQ** command simplifies computation of these parameters. When given the maximum number of points to be used and the frequency of a signal, **FREQ** computes the number of cycles and the optimum number of points required to create the desired sine frequency. For example, the command

1000 20E6 FREQ

instructs the software to create a 20 MHz (20E6) sine wave as closely as possibe using no more than 1000 points. The command then returns the context length and the number of cycles needed to create the desired sine wave. The frequency error in Hertz is also returned. The context information is the first item on the stack, the number of sine wave cycles is second, and the frequency error is third. The following commands create 4 sine waves with arbitrary frequencies at 4, 20, 34, and 48 MHz.

1000 4E6 FREQ	Get the context and number of cycles for a 4 MHz sine wave.
CTX RAMP?	Set the context to the value returned with **FREQ**.
*** PI *?**	Compute the phase information based on the second number on the stack.
SIN?	Create the 20 MHz sine wave.
"SINE1;" $DOWNLOAD	Store the first sine wave in the memory.

Similar commands are repeated three times to store the remaining sine waves (**SINE2** @ 20 MHz, **SINE3** @ 34 MHz, and **SINE4** @ 48 MHz).

The final step is to create a sequencer program to access each sine wave, generate the selected wave for the desired number of times to make up the dwell time, and then go on to the next sine wave.

The following example generates a hopping signal that switches between the 4 different sinusoidal signals:

```
NEWSEQ
"SINE1;30000;AUTO" $PACKET
"SINE2;20000;AUTO" $PACKET
"SINE3;15000;AUTO" $PACKET
"SINE4;24000;AUTO" $PACKET
GO
```

Each packet has a different number of repetitions, and depending on the size of the stored waveform, the dwell time of each signal will be different. The size of each waveform is computed by the **FREQ** command, and the dwell time can be set using that information.

Each packet must have a minimum dwell time of 2.752 μs (344 points X 8 ns/point) before the sequencer can go on to the next waveform packet. Therefore, either the sine wave must be composed of at least 344 points, or it must be repeated enough times to equal at least 2.752 μs if the number of points in the packet is smaller than 344.

Once a signal is generated for 2.752 μs, the sequencer can hop in 8 ns to the next packet. However, if an 8 ns switching speed is required within the 2.752 μs period, the signals can be concatenated using the software and stored together as one packet. Thus, the transition between signals does not involve a packet change.

Creating Application Programs

The same commands are often executed repeatedly when creating or modifying waveforms. To further increase efficiency and speed testing, personalized application commands can be defined which are strings of the Waveform Generation Language commands or other previously defined commands. New commands and programs can be developed and edited with the Waveform Generation Language to tailor the HP 8770S for specific applications.

The previous AM example will be used to construct an example application program. To create personalized application programs, a name has to be assigned to the required series of commands.

```
DEFINE AM                    Assign AM as the name of the program.

125 CTX                      ┐
RAMP PI* SIN                 │
STORE A                      │
RAMP PI* 20* SIN             │
STORE B                      ├ The AM example.
A .5*                        │
B*                           │
B+                           │
1000 CTX                     │
125 CLONE?                   │
"AM;" $DOWNLOAD GO           ┘
END                          End the definition of AM.
```

This AM program is now ready to generate the signal of **Figure 11** when the command "**AM**" is entered into the computer. The program can be designed to accept variable inputs as parameters for computing various signals. This method was used to create the FCNGEN program.

FCNGEN

To simplify the creation of common signals using WGL, the FCNGEN program is provided on an HP 11775A application programs disc. This program is written in WGL similar to the last example. **Table 1** is a list of the waveforms that can be generated using FCNGEN. When FCNGEN is entered, the user is prompted by a menu from which various options can be selected. Prompts are given to enter desired values for signal parameters.

Table 1. The list of the waveforms that can be generated by the FCNGEN program. This program is written in WGL for generation of commonly used receiver test signals. The FCNGEN program is included on an HP 11775A application programs disc.

RECEIVER	VARIABLE INPUT PARAMETERS
PULSE TRAIN:	
a) SIMPLE PULSE	PRF, pulsewidth, and rise/falltime (equal).
b) SYMMETRICAL PULSE	PRF and rise/falltime. The pulsewidth is equal to the interpulse period.
c) ARBITRARY PULSE	PRF, pulsewidth, risetime, and falltime.
TRIANGULAR WAVE	Frequency.
SINE WAVE	Frequency.
HAVERSINE	Frequency.
AM	Carrier frequency, modulation frequency, and modulation index.
FM	Carrier frequency, modulation frequency, and maximum frequency deviation.
CHIRP	Carrier frequency, modulation frequency, maximum frequency deviation, and pulsewidth.

As an example, this is how the FCNGEN program can be used to generate a triangular signal at 10 MHz:

Loading Instruction:

Place the application programs disc containing FCNGEN in the computer drive.
Type **"FCNGEN" $GET**, and press <enter>.
Press the EDIT key (for the HP 9816A/S, type EDIT and press ENTER).
Press the EXIT key to compile the FCNGEN program.

Using the FCNGEN Program

1) Enter **FCNGEN**
2) Enter option number (2) for triangular waveforms.
3) Enter 10 MHz when prompted for frequency.

WGL can manipulate a maximum of 16k waveform points which is 1/8 of the full memory of the HP 8770A. Generation of exact frequencies may require over 16k points. In such cases, the FCNGEN program will download the waveform piece-by-piece constructing the complete waveform automatically.

SUMMARY

This application note presents some examples of waveform creation and generation for receiver testing with the HP 8770S Arbitrary Waveform Synthesizer System. The system is a powerful waveform synthesizer that allows creation of complex signals by manipulating waveforms. Various test conditions and "real-life" operating environments can be simulated with simple waveform operations, eliminating the need for complicated test stations.

Appendix: Glossary of WGL Commands Used in this Note

CLONE Duplicates the specified number of waveform elements in the working wave as many times as necessary to fill the present window.

CTX Sets the total number of waveform elements to make up a waveform. Also known as "context".

DEFINE Used to create a new WGL command definition.

DOWNLOAD Stores the computed waveform data in the HP 8770A memory. The computer prompts for the name or address under which the waveform will be stored, and for the scale factor. The scale factor determines how much DAC dynamic range will be used to generate the waveform (a blank space, or the number 0, defaults the setting to full range).

END Used to signify the end of a WGL command definition.

EXP Natural antilogarithm e^x (e to the power x).

FDOMAIN Sets waveform creation in the frequency domain.

FREQ Determines the parameters to create a sine wave specified by the user. **FREQ** requires 2 parameters from the WGL stack and returns 3. Required are the maximum number of elements to be used and the sine wave frequency. Returned to the stack is the frequency error, if any, the number of elements to use, and the number of cycles contained by these elements.

FULL Selects all the elements of the present waveform to be manipulated.

GET Loads in predefined commands from an external file.

GO Initiates generation of waveforms.

HZ Converts a specified frequency into a discrete element position. This conversion is dependent on the clock rate and context size.

INT Computes integer portion of value(s).

LEFT Rotates a waveform to the left for a specified number of elements.

LOAD Fills all elements of the working wave in the present window with a specified value.

NEG Changes sign of a waveform or one element.

Appendix: Glossary of WGL Commands Used in this Note

NEWSEQ	Initiates a sequencer program definition.
NOISE	Fills all elements of the working wave in the present window with random values ranging from –1 to 1.
NORM or NORMALIZE	The elements of the working wave are scaled such that their values fall between ±1.
PACKET	Defines a packet with waveform name, number of repetitions, and triggering information in a sequencer program.
PI	The constant 3.14159265359.
RAMP	Fills all elements of the present window with linearly increasing values starting with –1. The incremental increase is 2/(number of elements contained within the present window). For example, a RAMP generated in a window of 200 elements would have the following values: –1.00, –.99, –.98, ..., .97, .98, .99.
REFL	Reflects a waveform in a window about the center of the horizontal axis.
RIGHT	Rotates a waveform to the right for a specified number of elements.
SIN	Takes the sin of each value in the working wave.
SQ	Squares a waveform or an element value.
STORE	Allows temporary storage of up to 5 waveforms and 150 real numbers.
STOREIN	Stores a single value into a specified element of a waveform.
TOFREQ	Converts a time domain signal to the frequency domain.
TOTIME	Converts a frequency domain signal to the time domain.
WINDOW	Selects portions of the entire waveform for manipulation.
XY	Exchanges the arrays in the working wave and auxiliary wave locations.
+	Adds the first two items on the stack.
–	Subtracts the first item on the stack from the second item.
/	Divides the second item on the stack by the first item.
*	Multiplies the first and second items on the stack.
?	Updates the WGL graphical display screen.

5954-6358 January 1986 Printed in U.S.A.