Tennessee Scholars Senior Project

Waveform Development Using Direct Digital Synthesis

Michael Symonds
May 10, 1991

ABSTRACT

The elements and operation of a typical direct digital synthesis system are described. The errors inherent in such a system are presented with a description of the corresponding noise in the output frequency spectrum. A direct digital synthesis system was designed, constructed, and tested to explore the capabilities of the AD9950 accumulator and AD9712A digital-to-analog converter. The hardware and software used to implement this system are described and results are presented.

INTRODUCTION

Direct Digital Synthesis is a technique which produces waveforms with very fine frequency and phase resolutions. A direct digital synthesizer, or DDS, creates these waveforms by first retrieving digitized samples of a sine wave from memory and then using these samples to drive a digital-to-analog converter. By selecting the order in which the samples of the waveform are converted, the frequency and phase of the waveform can be specified with a high degree of accuracy.

Direct Digital Synthesizers excel in systems requiring frequency or phase agility, such as frequency-hopping modems, tuneable tranceivers, spread-spectrum systems, and signal generators. In communication systems, they are often used as both the local oscillator and the modulator. Direct Digital Synthesizers are also useful when multiple sine waves must have a fixed, non-drifting phase and frequency relationship, such as quadrature oscillators.

A Typical Direct Digital Synthesis System

A block diagram of a typical DDS is shown in Figure 1. The stages in this diagram are present in all DDS systems, but may be grouped together differently depending on the intended output or the desired flexibility of the system.

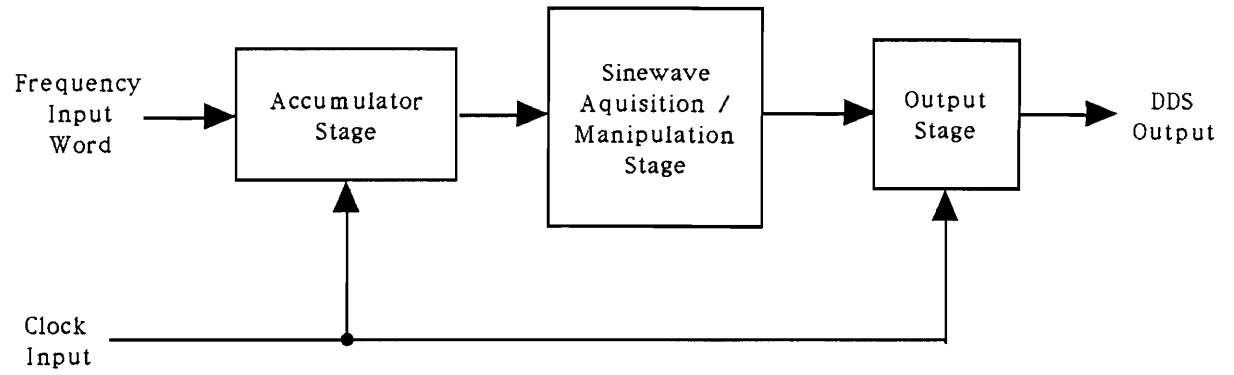


Figure 1. Block Diagram of a Direct Digital Synthesizer

The first stage of this system is the phase accumulator. The inputs to this stage are the system clock and an N-bit digital word corresponding to a specific output frequency. The digital word is stored in a frequency register in the phase accumulator, which is actually a special adder. Triggering on the clock input, the accumulator adds the contents of the frequency register to the current sum to obtain a phase word. This phase word is then used as the address for the memory contained in the second stage.

The second stage of the DDS is the sinewave acquisition/ manipulation stage. This stage contains waveform information in a look-up table stored in memory and digital logic gates to manipulate the digital sinewave data.

The output stage contains the digital-to-analog converter(DAC) and low-pass or band-pass filters. This stage takes the digital sinewave data and converts it into its analog equivalent, then applies an anti-aliasing filter to obtain the fundamental output. Figure 2 shows the basic transfer function of the DAC in this stage.

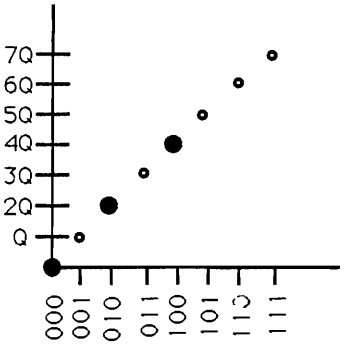


Figure 2. Transfer Function of a Digital-to-Analog Converter

Direct Digital Synthesis Operation

As an example of the operation of a DDS, it is assumed that the digital value of 1 is input as the frequency word and stored in the frequency register. At each trigger of the clock, the sum inside the accumulator increases by 1, and therefore the phase word, which is used to address the memory, also increments by 1 upon each clock. The DDS system will therefore step through each data point stored in the memory and send each point to the DAC. Assuming that one period of the waveform has been stored in memory which has a P-bit address word, the accumulator will step through the entire cycle of the stored waveform in 2^P clock cycles, resulting in an output waveform with frequency $(F_{clock})/2^P$. This waveform obtained by stepping through the memory as slowly as possible corresponds to the frequency resolution of the DDS.

If the digital value of 2 is input as the frequency word and stored in the frequency word register, the accumulator sum (and therefore the phase word) increases by 2 following each clock pulse. This means that the accumulator addresses every other value in the memory and will therefore step through an entire cycle of the stored waveform in $2^P/2$ clock cycles, resulting in an output waveform with a frequency of $2(F_{clock})/2^P$, or 2 times the frequency resolution. In this manner, the output frequency of the DDS can be calculated for any frequency input word as

$$F_{out} = F * (F_{clock} / 2^P)$$

or

$$F_{out} = F * (\text{frequency resolution})$$

where F is the value stored in the frequency word register.

Figure 3 shows the development of these two waveforms with the corresponding input frequency words and phase words sent to the memory.

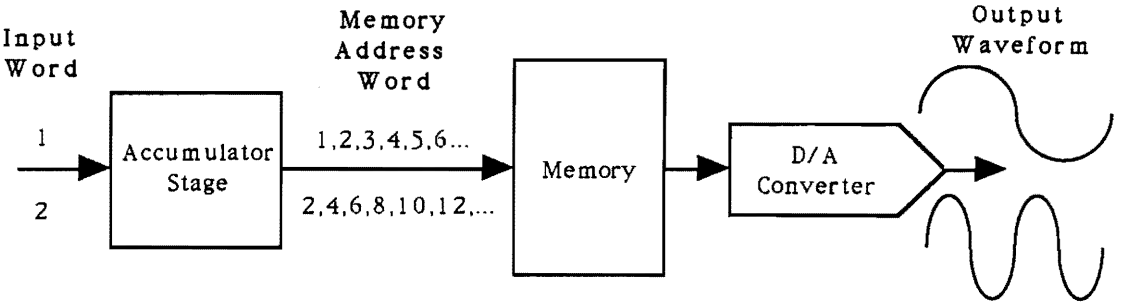


Figure 3. Development of Two Waveforms Using a DDS System

ANALYSIS OF DDS OUTPUT

Although a direct digital synthesis system is able to produce waveforms with highly accurate phase and frequency characteristics, there are several errors inherent in the DDS system that affect the resulting waveform and its frequency spectrum.

Amplitude Quantization Errors

The sinewave information stored in the memory and converted by the D/A converter of the system has 12 bits of resolution, but it actually represents an irrational number. This difference between the ideal amplitude value and the quantized value of the output leads to a quantization error in the output signal and quantization noise in the corresponding output frequency spectrum. The quantization effect of the D/A converter is shown in Figure 2, where any value within the region $nQ \pm 1/2$ is assigned to the level Q. The value of Q is called the resolution and represents the highest accuracy with which any value of the sinewave may be represented. Q may be found from:

$$Q = \frac{V_{fs}}{2^{N_b}}$$

where V_{fs} is the full scale peak-to-peak D/A converter output voltage and N_b is the number of input data bits for the converter. As the number of input bits increases, the value of Q, or the Least Significant Bit (LSB) value, decreases exponentially. Every D/A converter, regardless of the number of input bits, is subject to this finite resolution. The greatest error that can result from this resolution is half of the LSB, or Q/2.

To examine the quantization noise present in the output frequency spectrum, the waveform can be modeled as the summation of the exact sequence of sampled values plus an error component which shifts the exact level of the signal to the nearest quantization level. An input sine wave has the form:

$$F(nT) = Q\,(2)^{N_b - 1} \sin(\omega nT)$$

which has the maximum power value:

$$P_{sig} = \frac{(Q\,2^{N_b-1})^2}{2} = Q^2\,2^{2N_b-3}$$

Assuming that the quantization error of the signal is random and equally distributed over the range -Q/2 to +Q/2, the power of the error signal is given by:

$$P_{err} = \frac{\left[\frac{Q}{2} - \left(-\frac{Q}{2}\right)\right]^2}{12} = \frac{Q^2}{12}$$

The dynamic range (DR) of the system is a useful value which describes the relationship between the signal and noise contents of the frequency spectrum. It is defined as the ratio of the power of the output signal to the power of the error signal. Expressed logarithmically, this value is:

$$\begin{aligned} D.R. &= 10 \log_{10}\left(\frac{Q^2\,2^{2N_b-3}}{\frac{Q^2}{12}}\right) = 10 \log_{10}\left(2^{2N_b-3}\right) + 10 \log_{10}(12) \\ &= 10\left[(2N_b - 3)\log_{10}(2) + \log_{10}(12)\right] \\ &= 6.02\,N_b + 1.76 \end{aligned}$$

When examining the output of the DDS on a spectrum analyzer, the above equation will give the difference between the fundamental signal and the noise floor.

An interesting result of this quantization error occurs when the DDS system hits the same codes in the memory every cycle. As a result, the quantization noise is not spread out though the spectrum because the errors are not random, as expected. The noise is instead concentrated at the harmonics of the signal. This result can be avoided by using the largest practical number of codes per cycle and selecting the fundamental to be an odd and prime multiple of the resolution frequency.

Other Sources of Noise

One of the major sources of noise in a DDS system is the non-linearities of the D/A converter. These non-linearities are caused by the limitations in the design of a converter, and depend greatly on the actual converter used in the design. As a result, there is no good way to model these effects.

There are other non-ideal effects in the DDS system that lead to errors in the frequency spectrum. If the D/A converter is non-monotonic, it can lead to code-dependent frequency spurs that result when the non-monotonic codes are used. The phase noise inherent in the system clock is actually reduced by the accumulator because its output bits change at a lower frequency than the system clock, but this phase noise of the clock is reinjected at the D/A converter.

The Nature of the DDS System Output Spectrum

Ideally, the reconstruction of an analog signal from its samples would be represented in the frequency domain as the convolution of the original analog signal spectrum and a train of impulse functions:

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} x(nT_s)\,\delta(t-nT_s)$$

$$= x(t)\sum_{n=-\infty}^{\infty}\delta(t-nT_s)$$

The Fourier transform of the above result is taken, with multiplication in the time domain corresponding to convolution in the frequency domain.

$$X_\delta(f) = X(f) * f_s\sum_{n=-\infty}^{\infty}\delta(f-nf_s)$$

Interchanging the operation of summation and convolution:

$$X_\delta(f) = f_s\sum_{n=-\infty}^{\infty}\left(X(f) * \delta(f-nf_s)\right)$$

where:

$$X(f) * \delta(f-nf_s) = \int_{-\infty}^{\infty} X(u) * \delta(f-u-nf_s)\,du$$

$$= X(f-nf_s)$$

results in:

$$X_\delta(f) = f_s \sum_{n=-\infty}^{\infty} X(f - nf_s)$$

The resulting spectrum consists of the original signal spectrum plus its images centered at integer multiples of the sampling frequency. This means that there will be image responses at $(nF_{sample} \pm F_{sig})$, as shown in Figure 4. This Figure shows the relationship between the fundamental signal response and its images, but it only shows the result for the ideal case in which the signal is limited to a fundamental component. In actuality, the non-ideal effects of the D/A converter will result in error spikes in the spectrum which will also be 'folded back' in the same manner as the fundamental.

Since an impulse sampling function would place unrealistic slew rate demands on a real world D/A converter, its output is characterized instead by a 'staircase' function, as shown in Figure 2. The resulting frequency spectrum of the staircased waveform contains the original spectrum and its images, but it is attenuated by a $\text{sinc}(\pi F_{sig}/F_{sample})$ envelope. This envelope is the result of the multiplication in the time domain with a Π function. The effects of this envelope are also shown in Figure 4.

A Direct Digital Synthesis System: Introduction

To explore the basic concepts presented above, a direct digital synthesis system was designed, constructed, and tested. Additional goals of this project were to test the functionality and capabilities of the AD9950, an accumulator, and the AD9712A, a 12-bit D/A converter.

A system diagram of the DDS evaluation board is shown in Figure 5. The direct digital synthesis board contains the components used in the actual synthesis of the waveform. The inputs to this block include +5V, -5.2V, and ground supplies, and a variable reference clock. Computer control of the DDS system was implemented using a Metrabyte PI012 card and a 37 line interface cable. The reference clock was brought to the evaluation

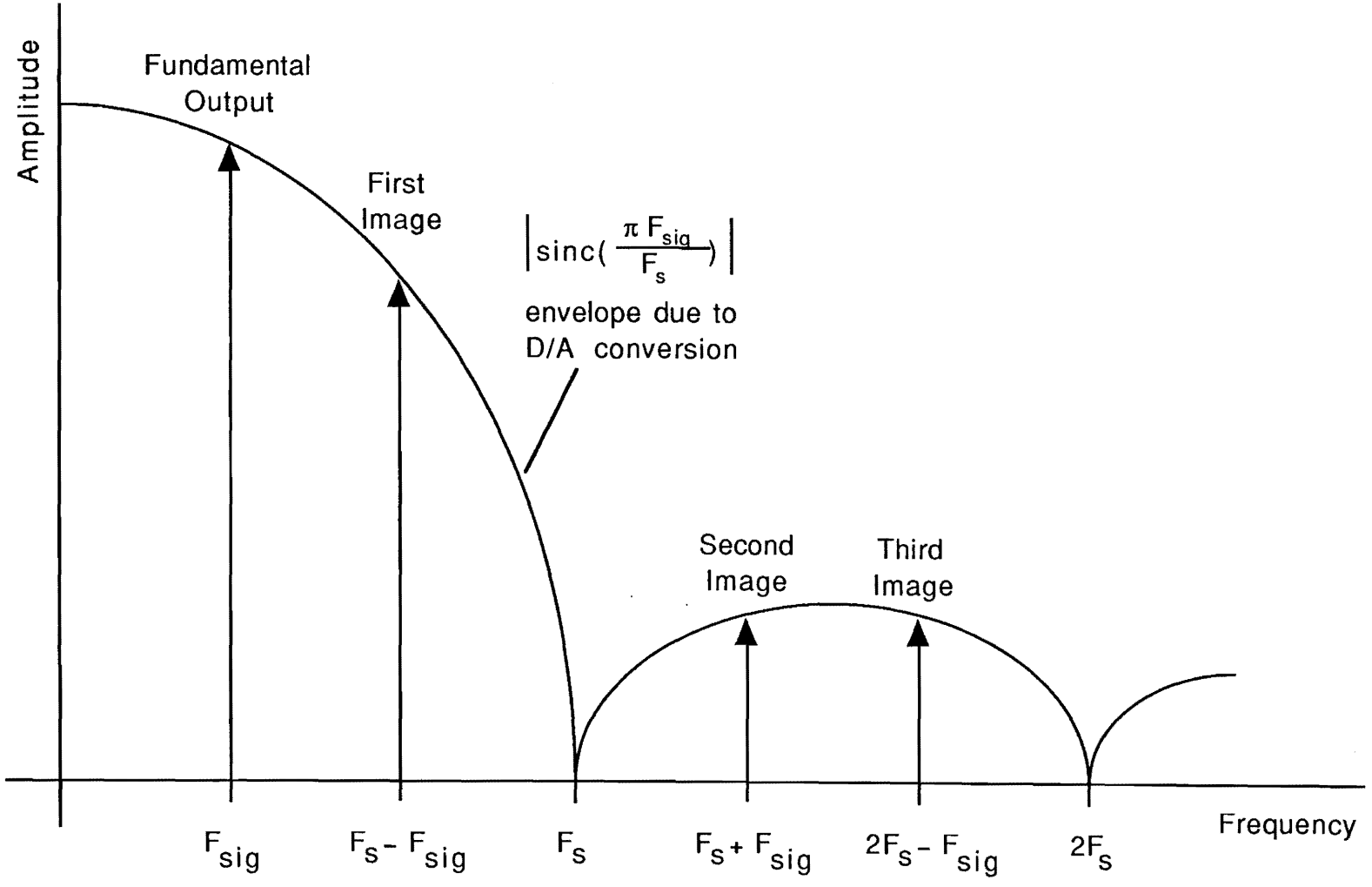


Figure 4. DDS Output Showing Effects of Sampling

board using standard coax cable. Both true and inverted output waveforms were available from SMA connectors for evaluation on both an oscilloscope and a spectrum analyzer.

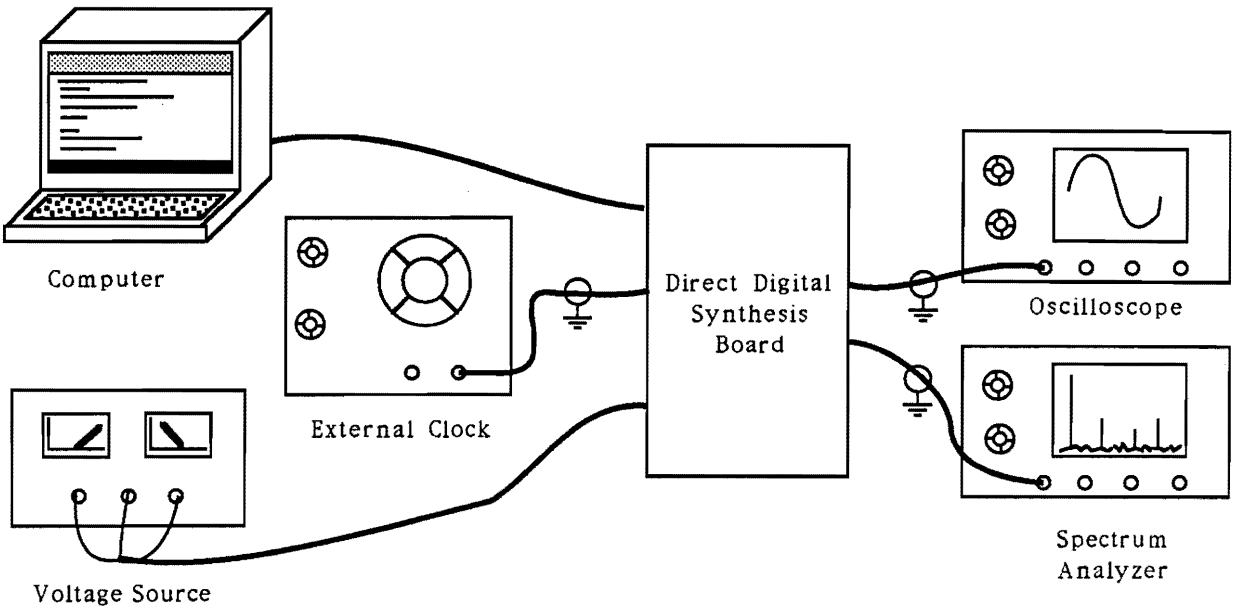


Figure 5. System Diagram for the DDS System

Figure 6 shows the block diagram for the DDS evaluation board. This diagram reflects the fact that the DDS system constructed for this project is very similar to the typical DDS system shown in Figure 1. It consists of four stages: the input stage, the accumulator stage, the sinewave acquisition/ manipulation stage, and the output stage. The input stage buffers the inputs from the computer and creates an appropriate clock signal. The accumulator stage contains the AD9950 accumulator. The sinewave acquisition/manipulation stage contains the memory in which the waveform is stored and digital logic which manipulates this data into an appropriate form. The output stage contains the AD9712A digital-to-analog converter and output connectors used to retrieve the signals from the board.

Figure 7 is a schematic of the direct digital synthesis evaluation board. The circuitry used in the input stage is on the left side and top of this schematic. The accumulator is given the reference number U10. The memory and data manipulation

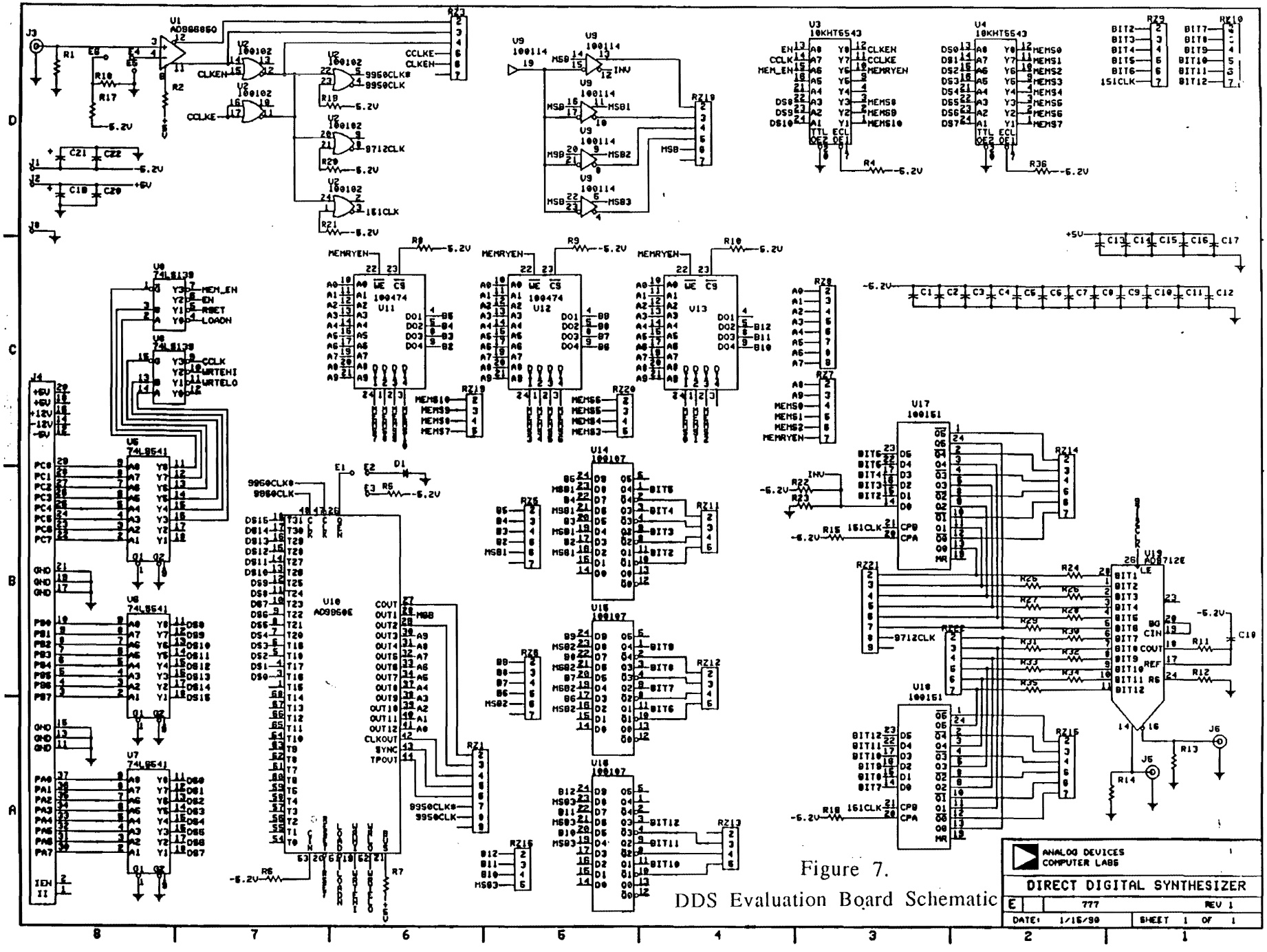


Figure 7.

DDS Evaluation Board Schematic

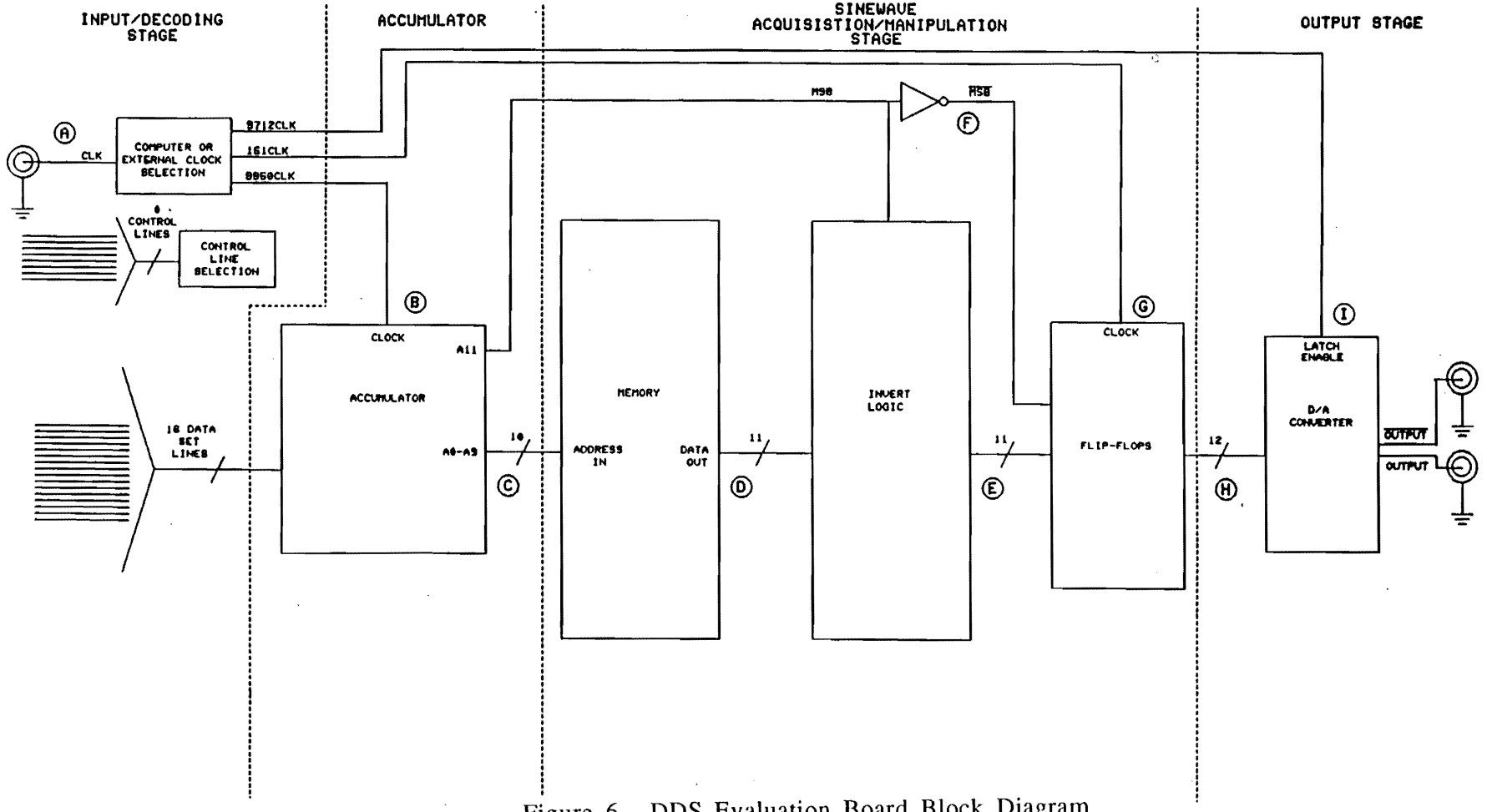


Figure 6. DDS Evaluation Board Block Diagram

circuitry are located in the middle of the schematic, and the output circuitry including the DAC is on the right.

This circuit includes a mixture of TTL and ECL logic level devices. The inputs to the board from the computer are all TTL levels, but the speed requirements of the rest of the system require a faster logic family. The inputs to the accumulator are TTL, but its outputs are ECL level, and every part after the accumulator is also ECL. In utilizing ECL logic, terminations must be placed on each run to prevent ringing.

Input Stage

The input stage of the DDS system consists of three major sections: the input buffers and control line selection, input clock selection, and TTL/ECL logic level conversion. Two different types of information are brought down to the evaluation board from the computer: control lines and data_set lines. A Metrabyte PI012 card inside the computer is used as the interface between the software and the DDS system during the creation of this information. The Metrabyte card accepts a port number and a hex value from the computer program and outputs an 8-bit digital word to a 37-pin output connector. This connector is interfaced via a cable to a 37-pin connector on the DDS board. The Metrabyte card is used to create three separate data words. Two of these words are sent to the board to be interpreted as sine wave data or accumulator data. The other is decoded into control commands used to direct the operations of the board.

The most significant data set word is brought down to the board on pins 3-10 of the 37-pin connector, while the least significant data set word is available on pins 30-37. These two data set words are buffered against the noise introduced by the cable and the computer using 74LS541 latches which have both output enables wired active. The control word is available to the DDS board on pins 22-29 of the input connector. This word is also buffered using a 74LS541.

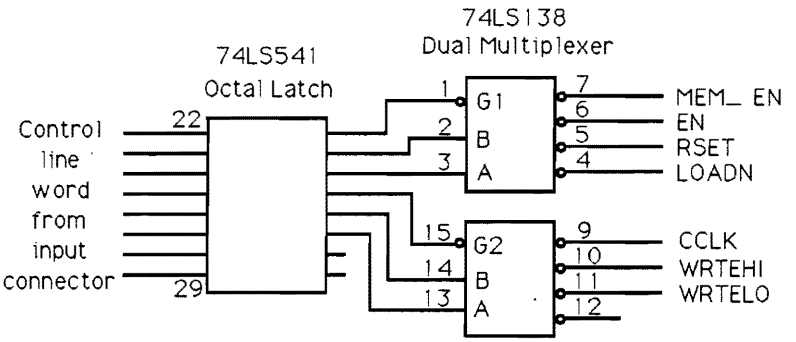


Figure 8. Circuitry Used to Decode the Control Lines

Figure 8 shows the arrangement used to interpret this input word once it has been received by the DDS board. The six most significant bits of the input data word are decoded using a 74LS138 Dual Multiplexer(mux). Inputs 1,2, and 3 of the mux enable a low logic level on pin 4,5,6,or 7. Similarly, mux inputs 13,14, and 15 enable a falling edge on pin 9,10, or 11. As a result of this arrangement, a maximum of two control lines can be active at one time, but the software is written and the board is designed in a manner such that no more than two active control lines will be needed to perform any operation. Table 1 shows the relationship between the control line word and the control line which is brought low as a result.

	DC7	DC6	DC5	DC4	DC3	DC2	DC1	DC0	
			A2	B2	G2	A1	B1	G1	MUX INPUT
CONTROL LINE									FUNCTION
MEM_EN	1	1	1	1	1	1	1	0	ENABLE MEMORY
EN	1	1	1	1	1	0	1	0	ENABLE EXTERNAL CLOCK
RSET	1	1	1	1	1	1	0	0	RESET AD9950 BUFFERS
LOADN	1	1	1	1	1	0	0	0	LOAD DATA WORD FOR AD9950
CCLK	1	1	1	1	0	1	1	1	COMPUTER-CONTROLLED CLOCK
WRTEHI	1	1	0	1	0	1	1	1	WRITE MSB DATA WORD FOR AD9950
WRTELO	1	1	1	0	0	1	1	1	WRITE LSB DATA WORD FOR AD9950
CLR	1	1	1	1	1	1	1	1	CLEAR THE CONTROL LINES
CLKLD	1	1	1	1	0	0	0	0	CLOCK BOARD WHILE LOADN IS STILL ACTIVE

Table 1. Control line assignments

The clock input section of the schematic is designed so that either an external clock input or a computer-controlled clock is selected as the system clock. This section is redrawn in Figure 9. Clock selection is implemented using two wired-NOR gates and two separate control lines. When either of these control lines is high, the output of its respective NOR gate is forced low. When the control line is low, the output of the NOR gate reflects the inverse of its other input. For the first NOR gate, the control line input is CLKEN, the ECL equivalent of the control line EN. The other input to this NOR gate is an external clock. As shown in Figure 9, this clock signal is forced into a square wave by a comparator. This implementation allows the input of sine waves as an external clock, yet still provides a well-shaped pulse for the system clock. Both inputs to the second NOR gate are tied to the control line CCLKE, the ECL equivalent of CCLK. As a result, the output of this gate will be the inverse of the control line CCLKE and by toggling this control line, the system clock can be toggled.

The outputs of the first and second NOR gates are tied together in a wired-NOR configuration, and the resulting output is used as the system clock. In this configuration, the output will be high if either of the gates tries to drive it high. When the DDS system is operating, either CLKEN or CCLKE will be enabled low, but only one will be low at a time to avoid both gates trying to drive the system clock at the same time. After the system clock has been determined, it is buffered to create three separate clock signals. These buffers are used to avoid any current limitations.

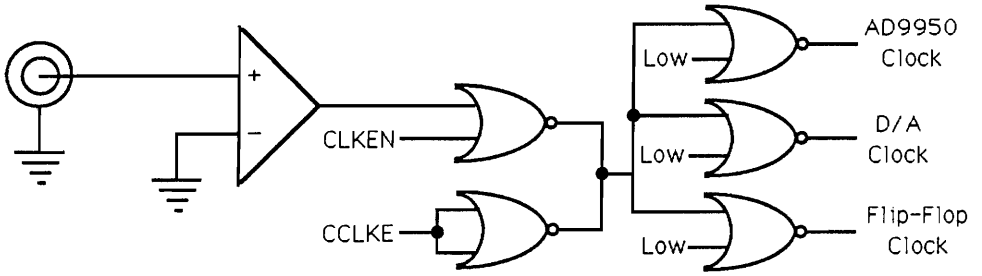


Figure 9. Clock Development Circuit

The final section of the input stage is the TTL/ECL converters. These octal converters are used to change the logic levels of control lines used in the clock development and both the

control line and the data lines used in waveform storage in the memory.

Accumulator Stage

The accumulator stage consists only of the AD9950 accumulator. The inputs to this stage are the 16 buffered data set lines and the control lines which direct the actions of the accumulator. The frequency word register of the AD9950 holds 32 bits of information. One option of the AD9950 utilized in this DDS system allows the accumulator to load this 32 bit register using only 16 input data lines. In utilizing this option, these data lines are selected to fill either the upper or lower 16 bits of the frequency word register by using the WRITEHI (pin 19) and WRITELO (pin 52) control line inputs. The outputs of the accumulator consist of 10 address lines used to access the memory and the MSB bit which is used in the sinewave data manipulation. Table 2 shows the input data_set bits and they are used to address the 9950 and the memory.

DS15	DS14	DS13	DS12	DS11	DS10	DS9	DS8	DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0	FUNCTION
T31	T30	T29	T28	T27	T26	T25	T24	T23	T22	T21	T20	T19	T18	T17	T16	9950 DATA INPUT
					MEM S10	MEM S9	MEM S8	MEM S7	MEM S6	MEM S5	MEM S4	MEM S3	MEM S2	MEM S1	MEM S0	MEMORY DATA INPUT

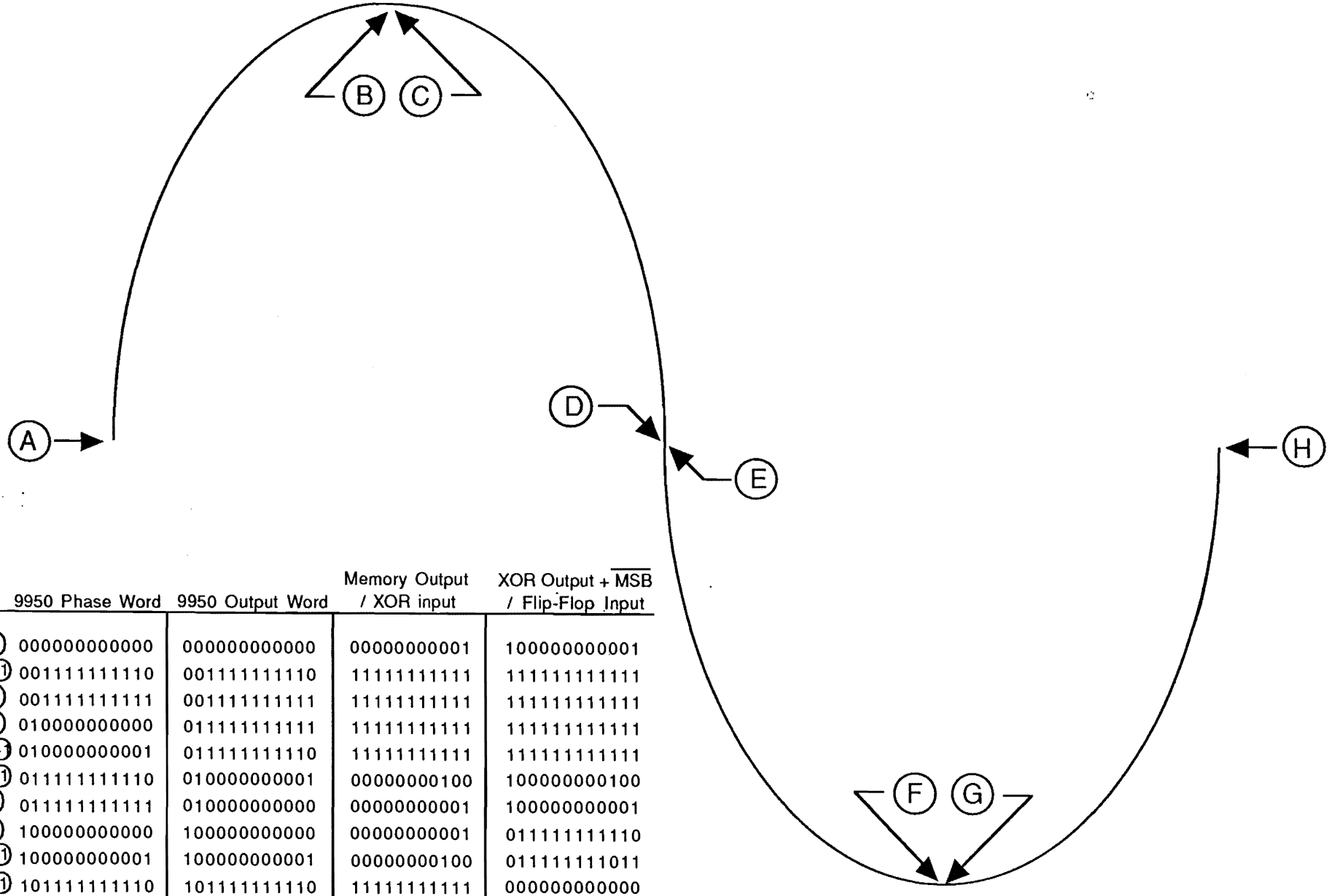
Table 2. Data_Set Bits and their input assignments

Acquisition/Manipulation Stage

The AD9950 has several special features which are utilized during the acquisition/manipulation stage of the DDS board. The most significant of these features is quadrature. The use of quadrature requires the DDS system to store only one quarter of the desired waveform and then externally manipulate this data to form the entire waveform.

First quadrant of the waveform

Figure 10 shows the digital data retrieved from memory during the construction of a sine wave. During the first quarter of the sinewave, the 10 least significant outputs (A0-A9) of the



	9950 Phase Word	9950 Output Word	Memory Output / XOR input	XOR Output + $\overline{\text{MSB}}$ / Flip-Flop Input
A	000000000000	000000000000	00000000001	100000000001
B-1	001111111110	001111111110	11111111111	111111111111
B	001111111111	001111111111	11111111111	111111111111
C	010000000000	011111111111	11111111111	111111111111
C+1	010000000001	011111111110	11111111111	111111111111
D-1	011111111110	010000000001	00000000100	100000000100
D	011111111111	010000000000	00000000001	100000000001
E	100000000000	100000000000	00000000001	011111111110
E+1	100000000001	100000000001	00000000100	011111111011
F-1	101111111110	101111111110	11111111111	000000000000
F	101111111111	101111111111	11111111111	000000000000
G	110000000000	101111111111	11111111111	000000000000
G+1	110000000001	101111111110	11111111111	000000000000
H	111111111111	110000000000	00000000001	011111111110

Figure 10. Development of a Sinewave Using Quadrature

AD9950 are used as the memory address and the accumulator counts through these addresses to acquire the data in the first quadrant. The most significant output of the AD9950 is logic level low during this quadrant. Each of the 11 output data bits from the memory serve as one of the 2 inputs to an XOR gate. The other input to this gate is the most significant output bit, A11, of the AD9950. This bit is low during the first quadrant, which means that the output bits of the memory propagate through this XOR gate unchanged. The waveform data is then deskewed using the 100151 D Flip-Flop. The data arrives at the input to these gates, but does not appear at the output until the system clock pulses the CPB input. It is at this point that an offset is added to the waveform data. This offset is created by using the inverse of the most significant bit of the memory address as the most significant input bit 12. During the first quadrant of the sinewave, this inverse will be logic level high as shown in the waveform diagram, Figure 10.

Second quadrant of the waveform

The second quadrant of the waveform changes slightly from the first quadrant. The most significant memory address bit is still low, as shown in Figure 10. As a result, the sinewave data is not inverted by the XOR gates and an offset is still added. The change occurs in the way that the sinewave memory is addressed. The beginning of the second quadrant corresponds to a change in the memory address word from 001111111111 to 010000000000. When high, Bit 11 (A10 of the output word) of the phase word is internally used by the AD9950 to count down instead of counting up. As the waveform enters the second quadrant, the internal phase word of the AD9950 counts up from 001111111111 to 010000000000 to 010000000001 while the memory address counts up to the maximum address and then begins to count down: 01111111111, 01111111111, 01111111110. The AD9950 will therefore count down through the sinewave data in the memory during this quadrant, as illustrated in Figure 10.

Third quadrant of the waveform

The function of the waveform manipulation becomes more obvious during the third quadrant. At the beginning of this quadrant, the internal phase word changes from 011111111111 to 100000000000. This corresponds to a change in the level of both the MSB and bit 11. Since bit 11 is no longer high, the accumulator output will now count up through the sinewave data as in quadrant 1. The MSB is also high during this quadrant, so an offset will not be added to the sinewave data and the data will be inverted by the XOR gates. The overall effect of these actions is to flip the waveform over and count through it again, as seen in Figure 10.

Fourth quadrant of the waveform

During the fourth quadrant, the MSB is still high so the offset is not added and the sinewave is inverted. Bit 11 is now high, however, and the accumulator will count back through the stored data, returning the waveform to its origin and completing the cycle.

Output Stage

The output stage consists of the AD9712A and the output connections. The inputs to this stage are the deskewed outputs of the D Flip-Flops. The AD9712A is configured to use the system clock as the latch enable signal. Both true and inverted outputs are available from the DAC for evaluation using an oscilloscope or spectrum analyzer.

Timing Diagram and Board Layout

Figure 11 is a timing diagram for the evaluation board. The letters in the left hand margin correspond to the location of the signal in the block diagram shown in Figure 6. The timing diagram shows how the data is acquired from the memory, inverted if appropriate, and then held (along with the MSB) until the next clock cycle before going to the AD9712A. The possibility existed that if the clock signal arrives at the flip-flops early, the data input to the D/A converter could change before the latch

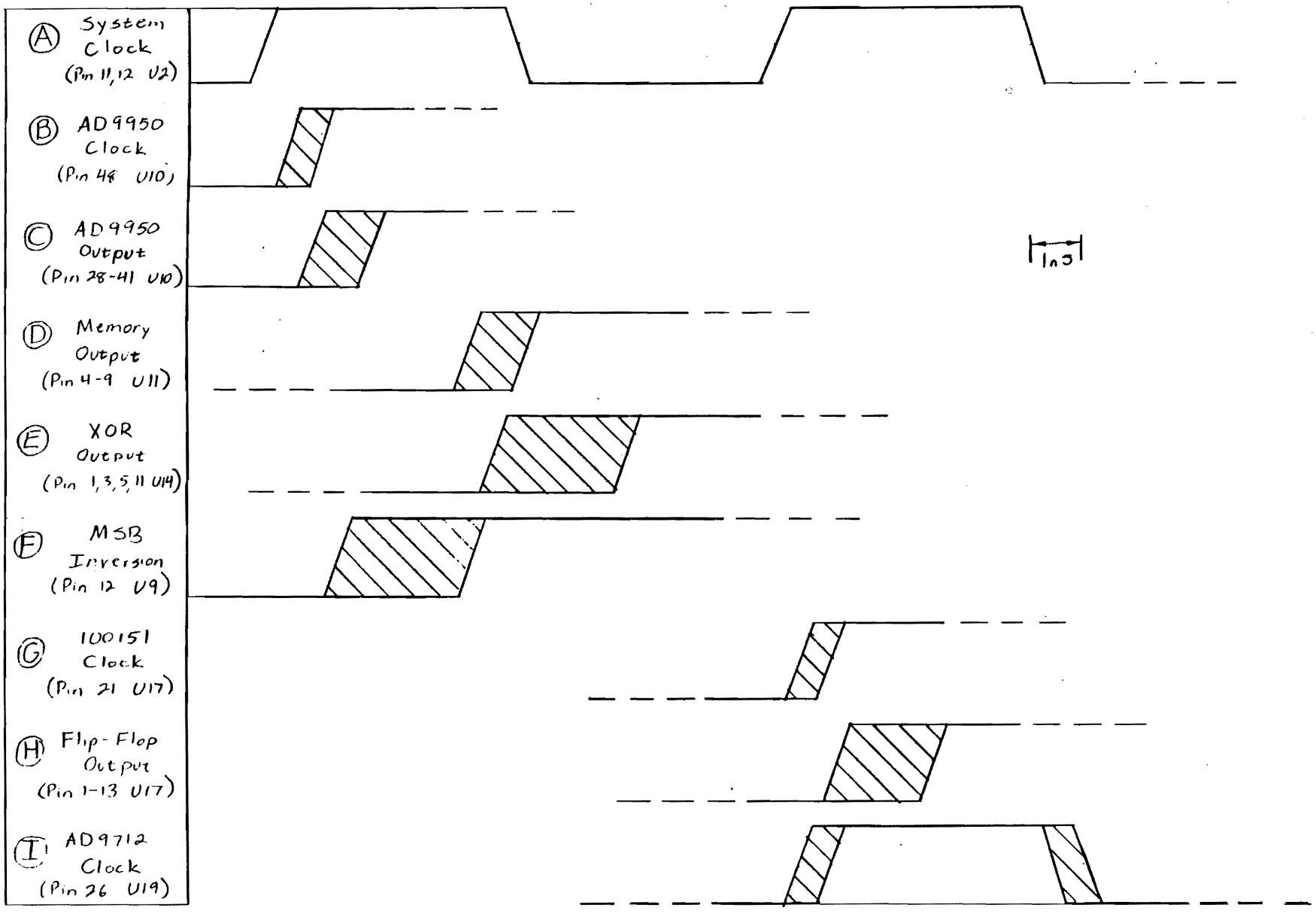


Figure 11. DDS Evaluation Board Timing Diagram

enable input returns high. This occurence was avoided by taking special precautions regarding run length on the final layout and was never observed to occur.

A layout of the DDS evaluation board is show in Figure 12. This drawing shows the location of the integrated circuits but not the runs used to make connections between the parts. The board layout shows an attempt to reduce run lengths and efficiently organize the sections of the DDS system. The input buffers and control line decoding circuitry are placed near the input connector at the left side of the board. The clock selection is placed at the top of the board for easy distribution. It is critical to reduce the path length of these runs as much as possible to avoid problems associated with noise. The AD9950 is placed in the large test socket located next to the input buffers. This allows easy substitution of new parts in testing for functionality. The rest of the IC placements follow the flow-through approach of the DDS system, where run lengths and run placements are considered to minimize cross-talk to other signal traces and coupling through the power and ground planes.

Software Development

The software routines developed to control the operation of the DDS system consist of two separate areas. The first is the preparation of the DDS system to produce waveforms and the second is the computer control of the frequency of the resulting waveforms. The program was written in QuickBasic and the source code is contained in Appendix A.

The first step in preparing the DDS system is to clear the buffers in the AD9950 accumulator. This is accomplished by first writing zeros to all of the data inputs. These inputs are then loaded into the frequency word register by first taking the LATCH MSB line low and then taking the LATCH LSB line low using the control words defined in Table 1. The contents of the frequency word register are then transfered to the delta phase register by first taking the LOADN control line low and then clocking the AD9950 manually. These two commands are also executed using

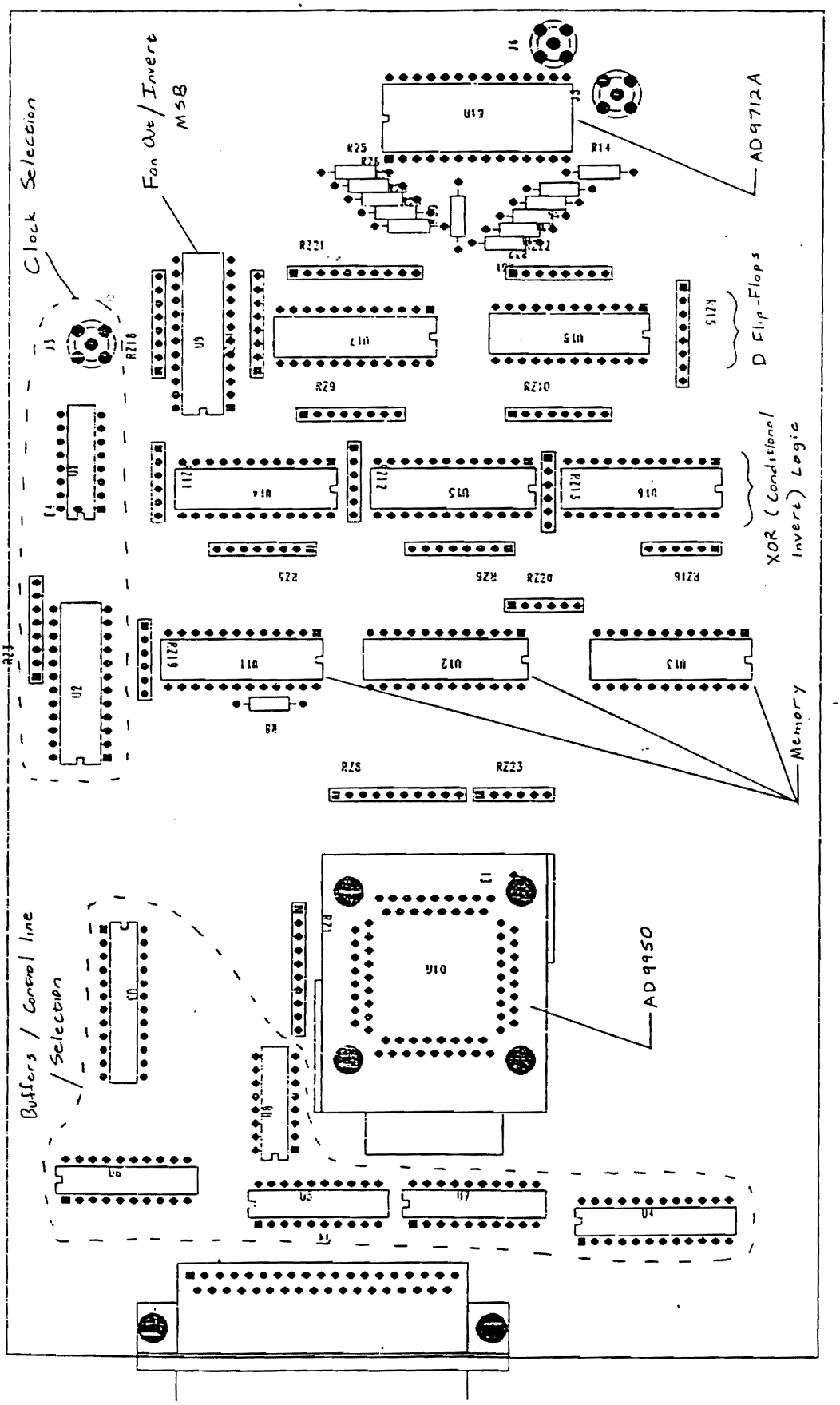


Figure 12. DDS Evaluation Board Layout

the control words in Table 1. The output register is then cleared using the RSET control line.

The next step in preparing the DDS system is storing the waveform in the memory. This is accomplished by allowing the accumulator to count through the memory addresses and inputting the appropriate waveform data at each address. The accumulator is set up to count through the addresses by sending a digital value of 16 to the data set lines. This value will cause the accumulator to count by 1 at its output. The input word is stored in the AD9950 delta phase register by sending the LATCH MSB, LATCH LSB, and CLKLOADN control words. Due to propagation delays inside the AD9950, the accumulator must be clocked five times to achieve the desired initial state. At this initial state, the accumulator output is zero, but after the next clock, the output will increase by 1 with each clock. The program then calculates the first data point in the sinewave and sends this data word to the memory inputs. The storage capabilities of the memory chips are then enabled by pulsing the MEM_EN control line. The AD9950 clock is then pulsed to create the next memory address and the above routine is followed. This procedure continues until all 1024 (2^10) memory addresses have been loaded with data from the first quadrant of the waveform. The program then clears the AD9950 buffers and registers using the routine developed above. The DDS board is now initialized and ready to start creating waveforms.

The desired waveform frequency is then requested from the user and then converted into the corresponding delta phase word. This 32 bit word is broken up into two 16 bit words. The least significant of these is sent to the data_set lines and the LATCH LSB line is taken low. Then the most significant data word is loaded using the LATCH MSB control line. The delta phase word is loaded into the delta phase register using the LOAD and CCLKLOAD control lines. The external clock is then enabled to convert at high speeds or the functionality of the board is tested using the manual computer clock. Once the waveform data has been stored, a new delta phase word can be entered at any time to change the output frequency.

Results

The Direct Digital Synthesis system functions as designed and is able to create waveforms with precise frequecies up to several Megahertz. The output waveforms were evaluated using both an oscilloscope and a spectrum analyzer. A plot of the frequency spectrum from the spectrum analyzer is shown in Figure 13. This plot shows the relationship between the fundamental frequency of the sinewave and the noise floor of the system. The plot also shows the effects of the non-linearities of the digital-to-analog converter.

Conclusions

This system shows that Direct Digital Synthesis is an effective technique for producing waveforms with highly precise frequencies. The AD9950 accumulator and AD9712A DAC were also shown to be both functional and well-suited for a high speed signal reconstruction system.

F_{CLOCK} = 50 MHz
F_{OUT} = 1.00001 MHz

PART # 2
AD9912A (DDS BOARD)

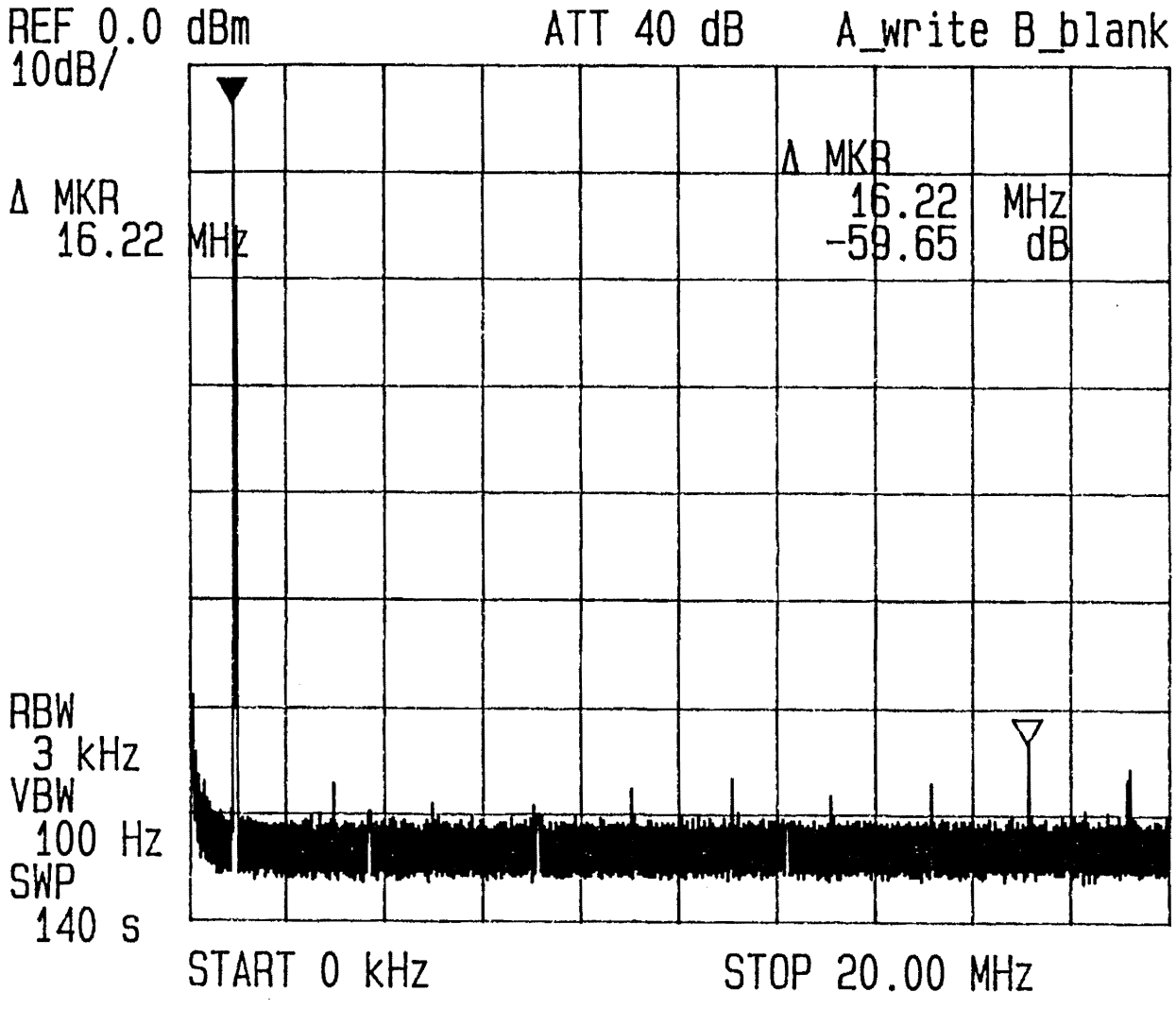


Figure 13a

9959/9712 EVAL. BOARD

F_{CLOCK} = 61.44 MHz
F_{OUT} = 11 MHz

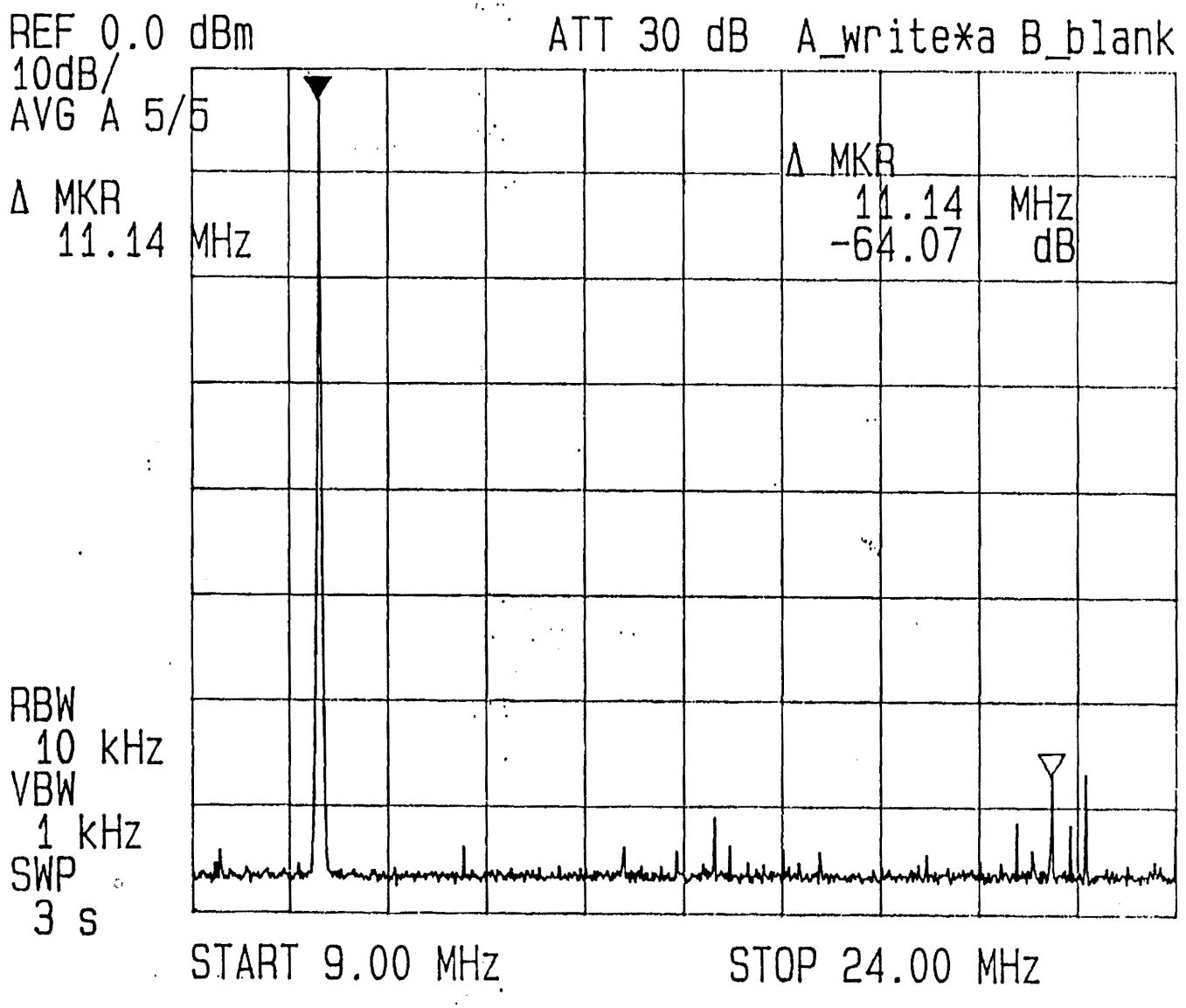


Figure 13b

F_{CLOCK} = 50 MHz

F_{OUT} = 11 MHz

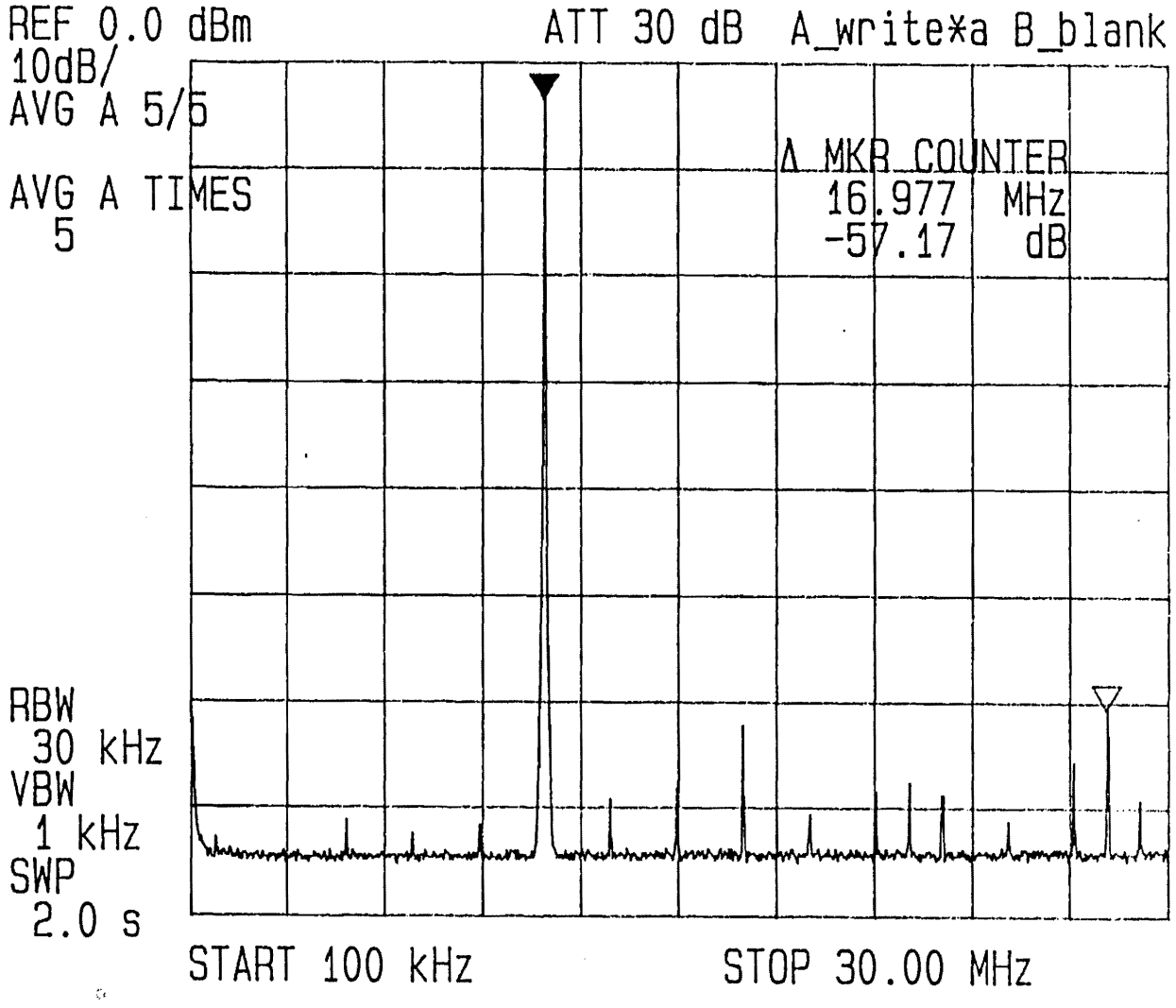


Figure 13c

F_{CLK} = 40.96 MHz

F_{OUT} = 5.5 MHz

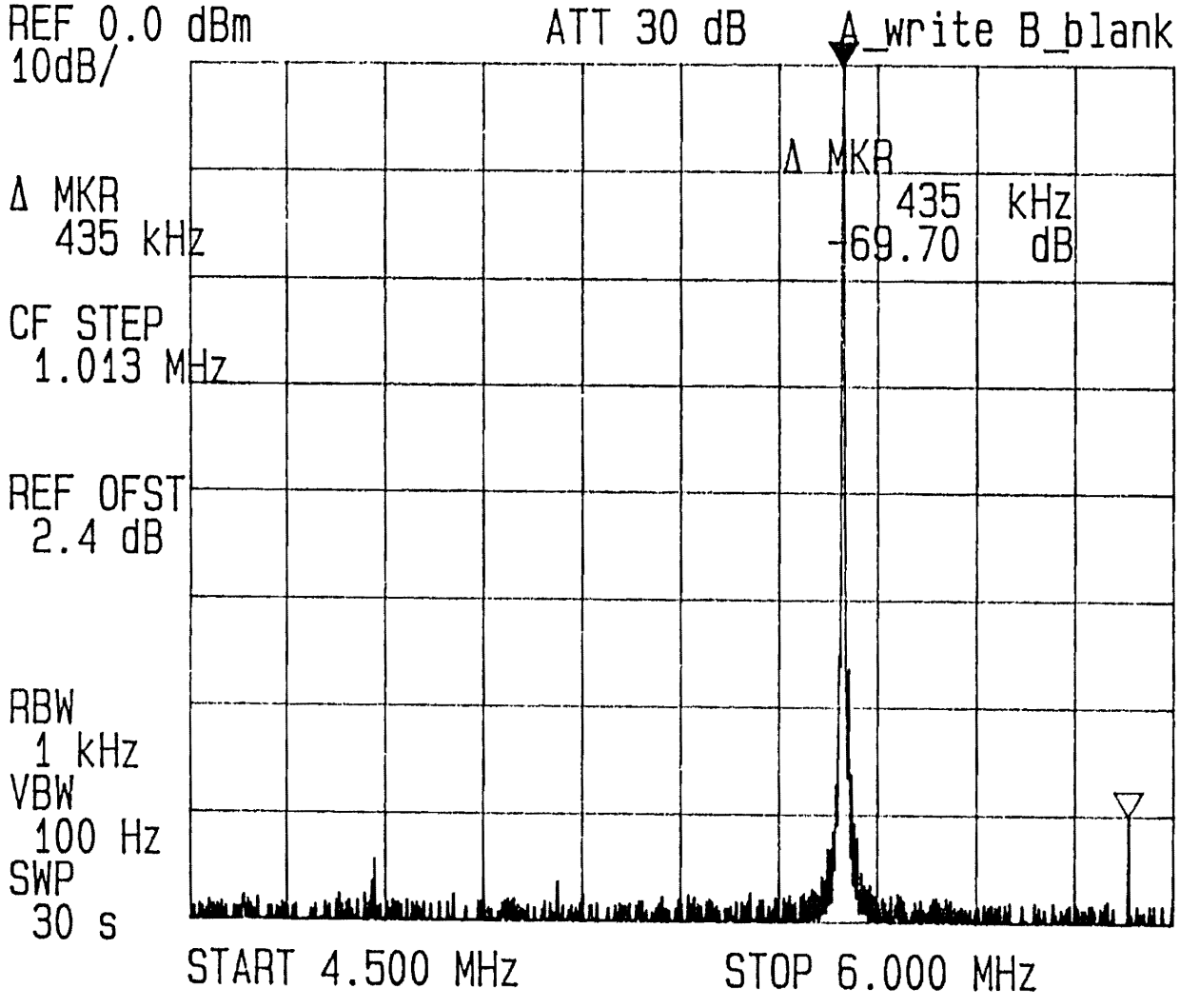


Figure 13d

```
'*******************************************************************************
'
'     AD9950/9712 EVALUATION BOARD PROGRAM     M.J.S.       3-10-89
'
'*******************************************************************************
DIM nmber AS DOUBLE

REM $INCLUDE: 'gpmb.BI'

'*******************************************************************************
'   CREATE THE ADDRESS VARIABLES TO BE USED BY THE EVALUATION BOARD
'*******************************************************************************

CONST memen = &HFE
CONST en = &HFA
CONST BUFRSET = &HFC
CONST LOADN = &HF8
CONST cclk = &HF7
CONST WRTEHI = &HD7
CONST wrtelo = &HE7
CONST clkld = &HF0
CONST PI = 3.1415926535897#

'*******************************************************************************
' Motherboard (GPMB) Address variables - PIO12 Cards
'*******************************************************************************

 setadd% = &H202                ' -   Address byte for data set bus
 sethigh% = &H201               ' -   High byte on data set bus (DS15-DS8)
 setlow% = &H200                ' -   Low byte on data set bus (DS7-DS0)
 READADD% = &H302               ' -   Address byte for data read bus
 readhigh% = &H301              ' -   High byte on data read bus (DR15-DR8)
 readlow% = &H300               ' -   Low byte on data read bus (DR7-DR0)

'*******************************************************************************
' Define Ports for GPMB
'*******************************************************************************

 OUT &H203, &H80                        'ports a,b,c output
 OUT &H303, &H92                        'ports a,b input / port c output

'*******************************************************************************
' CLEAR THE 9950 BUFFERS
'*******************************************************************************

 CLS

 OUT setlow%, 0             ' SET BOTH INPUT DATA BITS LOW
 OUT sethigh%, 0
 OUT setadd%, WRTEHI        ' WRITE TO THE UPPER-16 BUFFER OF THE 9950
 CALL DELAY(10)
 OUT setadd%, wrtelo        ' WRITE TO THE LOWER-16 BUFFER OF THE 9950
 CALL DELAY(10)
 OUT setadd%, CLR
 OUT setadd%, LOADN         ' ENABLE LOAD PIN OF THE 9950
 OUT setadd%, clkld         ' CLOCK THE 9950 WHILE KEEPING THE LOAD PIN LOW
 OUT setadd%, CLR
 OUT setadd%, BUFRSET       ' RESET THE 9950 ACCUMULATION AND OUTPUT BUFFERS

'*******************************************************************************
'
'ENTER THE WAVEFORM INTO THE MEMORY
'
'*******************************************************************************
```

```
 OUT setlow%, 0
 OUT sethigh%, 0
 OUT setadd%, WRTEHI
 CALL DELAY(10)
 OUT setadd%, wrtelo
 CALL DELAY(10)
 OUT setadd%, CLR
 OUT setadd%, LOADN
 OUT setadd%, clkld
 OUT setadd%, CLR
 OUT setadd%, BUFRSET

'*******************************************************************************
' ASK FOR AND ENTER A DELTA-PHASE WORD WHICH WILL OFFSET THE INITIAL MEMORY
' ADDRESS. THIS IS USEFUL FOR CASES IN WHICH THE 9950 OUTPUTS THE SAME
' MEMORY ADDRESSES EACH CYCLE. AN OFFSET IN THE MEMORY ADDRESS WILL CHANGE
' THE OUTPUT SPECTRUM IN THESE CASES.
'*******************************************************************************

 INPUT "high bit offset"; hbo
 INPUT "low bit offset"; lbo

 OUT sethigh%, hbo
 OUT setlow%, lbo
 CALL DELAY(10)
 OUT setadd%, WRTEHI

 FOR COUNT = 1 TO 5          ' CLOCK THE WORD THROUGH TO THE OUTPUT
  OUT setadd%, clkld
'  CALL DELAY(10)
  OUT setadd%, CLR
 NEXT COUNT

'*******************************************************************************
' ASK FOR THE DESIRED OUTPUT FREQUENCY AND THE CLOCK FREQUENCY AT WHICH IT
' WILL BE GENERATED, THEN CALCULATE THE DESIRED MEMORY STEP SIZE AND DIVIDE
' IT INTO 4 8-BIT WORDS TO SEND TO THE 9950
'*******************************************************************************

freqselect:

 INPUT "enter the clock frequency in MHz"; p
 IF p = 0 THEN END
 INPUT "enter the desired output frequency in MHz"; q

 number = ((q / p) * 4096 * (2 ^ 20))    ' CALCULATE THE DESIRED STEP SIZE

 lowlowbit = CINT(number AND 255)             'FIND THE LOW BIT, LOW REGISTER
number = number - lowlowbit

 lowhighbit = CINT((number AND 65280) / 256)  'FIND THE HIGH BIT, LOW REGISTER
number = number - (number AND 65280)

highlowbit = CINT((number AND 16711680) / 65536) 'FIND THE LOW BIT, HIGH REG.
number = number - (number AND 16711680)

highhighbit = CINT(number / 16777216)        'FIND THE HIGH BIT, HIGH REGISTER

PRINT "lowlowbit"; lowlowbit
PRINT "lowhighbit"; lowhighbit
PRINT "highlowbit"; highlowbit
PRINT "highhighbit"; highhighbit

OUT sethigh%, lowhighbit
OUT setlow%, lowlowbit
```

```
' PLACE A WORD IN THE DELTA PHASE REGISTER SUCH THAT THE 9950 WILL INCREASE
' THE OUTPUT (MEMORY ADDRESS) WORD BY ONE EACH CLOCK PULSE.
'*******************************************************************************

 OUT sethigh%, 0
 OUT setlow%, 0
 CALL DELAY(10)
 OUT setadd%, wrtelo
 OUT setadd%, CLR
 OUT sethigh%, 0
 OUT setlow%, 16
 CALL DELAY(10)
 OUT setadd%, WRTEHI
 CALL DELAY(10)
 OUT setadd%, CLR
 OUT setadd%, LOADN
 OUT setadd%, clkld
 OUT setadd%, LOADN
 OUT setadd%, CLR

'*******************************************************************************
' CLOCK THE 9950 SO THAT THE MEMORY ADDRESS IS ZERO, BUT WILL CHANGE TO 1
' AFTER THE NEXT CLOCK
'*******************************************************************************

 FOR COUNT = 1 TO 5
  OUT setadd%, clkld      'USE THE COMPUTER TO CLOCK THE 9950 5 TIMES
  OUT setadd%, CLR
 NEXT COUNT

'*******************************************************************************
' ENTER THE VALUE OF THE SINEWAVE AND THEN CLOCK THE 9950 TO
' INCREMENT THE MEMORY ADDRESS AND REPEAT UNTIL THE MEMORY IS FILLED.
'*******************************************************************************

   FOR ADDCOUNT = 0 TO 1023
      SINEVAL = CINT(2047.5 * SIN((2 * PI * (ADDCOUNT + .5)) / 4096) - .5)
                  '2047.5
      LOWVARIABLE = SINEVAL AND 255  'TAKE OFF FIRST 8 BITS FOR LOW WORD
      'LOWVARIABLE = LOWVARIABLE AND 240
      HIGHVARIABLE = ((SINEVAL AND 1792) / 1792) * 7  'TAKE OFF 2ND 8 BITS

      'LOWVARIABLE = 0          ' THESE MEMORY VALUES WILL CREATE
      'HIGHVARIABLE = 0         ' A SQUARE WAVE OUTPUT

      OUT setlow%, LOWVARIABLE
      OUT sethigh%, HIGHVARIABLE
      OUT setadd%, memen        ' ENABLE THE MEMORY TO STORE THE DATA POINTS
      OUT setadd%, CLR

'*******************************************************************************
' CLOCK THE 9950 TO INCREASE THE MEMORY ADDRESS BY 1 AND LOOP AGAIN
'*******************************************************************************

       OUT setadd%, cclk
       OUT setadd%, CLR

   NEXT ADDCOUNT

'*******************************************************************************
'PREPARE THE 9950 FOR ACCUMULATION BY CLEARING THE BUFFERS AND ENTERING
' A NEW DELTA PHASE WORD.
'*******************************************************************************

' CLEAR THE 9950 BUFFERS
```

```
 OUT setadd%, wrtelo
 OUT setadd%, CLR
 OUT sethigh%, highhighbit
 OUT setlow%, highlowbit
 CALL DELAY(10)
 OUT setadd%, WRTEHI
 CALL DELAY(10)
 OUT setadd%, CLR
 OUT setadd%, LOADN
 OUT setadd%, clkld
 OUT setadd%, LOADN
 OUT setadd%, CLR

' CLOCK THE 9950 SO THAT ZERO IS AT THE OUTPUT, BUT THE FIRST MEMORY ADDRESS
' WILL APPEAR AT THE OUTPUT AFTER THE NEXT CLOCK

' FOR COUNT = 1 TO 6
 ' OUT setadd%, clkld
'  CALL DELAY(10)
  'OUT setadd%, CLR
' NEXT COUNT

'*****************************************************************************
' CLOCK THE 9950, ADDRESSING THE MEMORY
'*****************************************************************************

GOTO EXTERNALCLOCK    ' THE CLOCK CAN BE CREATED EXTERNALLY OR ONE PULSE AT
                      ' A TIME USING THE COMPUTER

'*****************************************************************************
'    **    CLOCK USING THE COMPUTER      **
'*****************************************************************************

COMPUTERCLOCK:
        INPUT "ENTER 'D' FOR DONE, ANYTHING ELSE TO CLOCK THE 9950"; ANSWER$
        IF ANSWER$ = "D" THEN GOTO done
        OUT setadd%, cclk
        OUT setadd%, CLR
GOTO COMPUTERCLOCK

'*****************************************************************************
'    **     CLOCK USING AN EXTERNAL SOURCE    **
'*****************************************************************************

EXTERNALCLOCK:

        OUT setadd%, en       ' ENABLE THE PATH FOR THE EXTERNAL SOURCE

        GOTO freqselect       ' GO BACK TO THE FREQUENCY SELECTION

done: END
```