

# AD9850 Direct Digital Synthesizer Module Manual

Issue 4. February 2002

## Background

The AD9850 is an extremely useful DDS chip which allows any frequency from DC up to about 40MHz to be generated with tiny step sizes. The chip accepts a master clock frequency up to 120MHz and generates any output frequency up to about 1/3 of the master frequency. Resolution is  $2^{-32}$  times the master frequency (about one four-billionth). For example, a 100MHz reference frequency could be used to make a VFO covering 0 to 30MHz with a resolution of less than 0.025Hz.

The problem with the chip is that it is tricky to program, requiring a long serial word to set its various parameters. It is also rather difficult to assemble manually because it comes in a 28 pin Shrink SO package with a lead pitch of 0.65mm. The HF DDS PCB was developed to overcome these problems. It accommodates the AD9850 chip and a PIC microcontroller which converts simple ASCII RS232 commands into the serial words required by the DDS chip. It also incorporates space for a voltage regulator and output filter components onto a single convenient PCB.

## Application

The module is intended for use under various circumstances:

- A PC- or other intelligent module-controlled RF signal source
- An arbitrarily programmable / settable stand-alone RF signal source
- A local oscillator for a PC controlled radio receiver or transmitter
- A module for experimentation

## Description

The module is based on the Analog Devices AD9850 Direct Digital Synthesizer chip. The device will accept a 0-5V logic level signal clock of up to 120 MHz and produce an output frequency up to around 35% to 40% of this value with a resolution of 1 part in  $2^{32}$  of the clock rate. Full details of Direct Digital Synthesis or functioning of the AD9850 chip are not given here, and the data sheet available from Analog Devices should be consulted for further details. This module is designed to work with any valid clock rate up to the maximum allowed by the chip, the only components that need to be varied from those shown in the circuit diagram to suit a change in clock frequency are those in the output filter. Values listed below are suitable for a clock frequency of 100 MHz, for any other clock simply scale the values appropriately. The RF output sinewave is 0.35V RMS (1V pk-pk) from a source impedance of 100 ohms.

The block diagram is shown below.

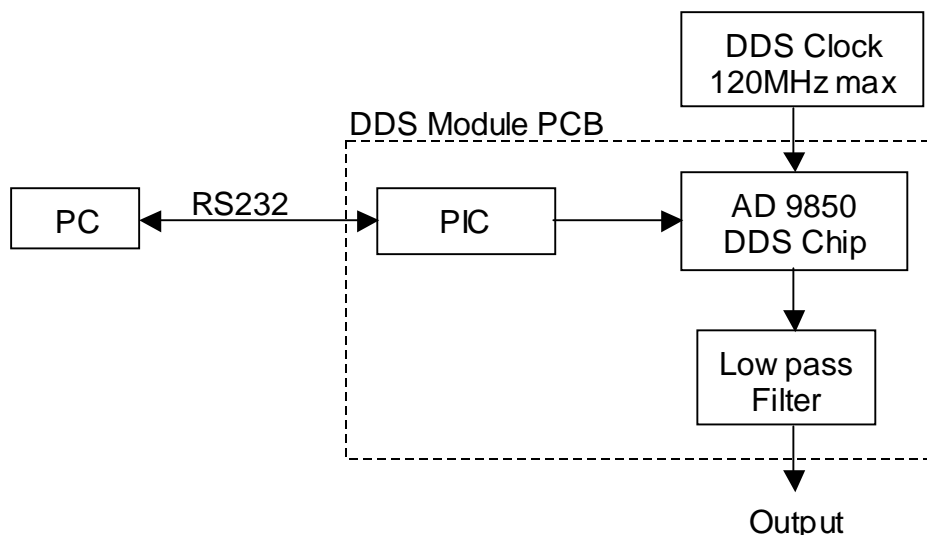


Figure 1. DDS Module Block Diagram

An on board PIC microcontroller provides a convenient-to-use interface from a serial port of a PC or similar using plain text commands. This in turn decodes the commands and formats the data into the serial stream required by the AD9850 chip. The command protocol adopted is not the fastest scheme that could be employed - a frequency change takes at least 24ms - but there is scope for comprehensive error checking with the bidirectional data link, as well as the ability to control the DDS from any available ASCII serial terminal.

As well as the normal CW output, a times four frequency option is provided where the actual frequency produced is 4 times that requested by the command data. This is to enable a quadrature divider to be driven from the logic level output, and give its output at the wanted frequency. This function can be activated by grounding the B7 input to the

PIC and is primarily intended for when the DDS module is used as the local oscillator for a zero IF based receiver / transmitter with I/Q LO drive. No error checking on data validity is performed, and when in \*4 mode the requested frequency can only have a value up to 40% / 4 of the clock, ie. no higher than a tenth of Fc

As a convenience for any custom software that may be used with several modules, the clock frequency (or any other data in numeric or hex format) can be stored in 5 bytes of the PIC EEPROM memory and read out at any time. The data is not used internally for DDS control so the format is left entirely to the user; one suggested format for storing clock frequency is given at the end.

Two versions of the PIC firmware are available. DDSPC3 requires the PIC to have a 4MHz crystal or ceramic resonator, usually independent of the DDS clock. DDSPC3F is designed for a PIC clock of 5MHz, in particular for LF use where a standard frequency source of this value can be employed for both DDS and PIC clock generation. The PIC can then conveniently be used to buffer a low level 5MHz signal to give the 0 – 5V waveform wanted by the AD9850 . The PIC clock only significantly affects the communication baud rate, and by changing to a value of 8MHz (or 10 MHz with DDSPC3F), the rate will then become 38400 baud allowing commands to be sent at twice the speed although this has not been tested.

Also not yet tested, the AD9851 DDS chip is pin compatible and includes an on chip X6 clock multiplier allowing higher output frequencies. Minor changes to the PIC software to enable / disable the multiplier should allow this device to be used instead. See the device data sheet available from [WWW.ANALOG.COM](http://WWW.ANALOG.COM)

### External requirements

The module requires:

- Power supply: 7 to 12V DC unregulated (it is regulated to +5V on-board)
- DDS Clock: 0 to 5V P-P clock signal at up to 120MHz
- RS232 commands: From an external PC or other intelligent programming source

### Choosing the DDS clock frequency

The choice of the DDS clock frequency depends on two principal needs:

- a) **Maximum output frequency.** The module will work with an output frequency up to about 1/3 of the DDS clock frequency. The limitations here depend on the design of the output filter and the level of spurious outputs which are acceptable.
- b) **Frequency resolution.** The DDS chip will generate frequencies in discrete steps equal to the DDS clock frequency divided by  $2^{32}$ . Thus, if you need to have frequency steps of 0.01Hz you will need a DDS clock frequency of 42.94967296 MHz. Note that even at the maximum DDS clock frequency of 120MHz the frequency step size will be about 0.028Hz which will be sufficiently small for many applications.

### Designing the Output filter

The output filter is essential for the correct operation of the DDS circuit as a whole. Its design will depend of the choice of DDS clock frequency and there is a section below which goes into the detail of the filter design.

### PCB Construction

The circuit diagram and layout are as shown in the drawings below.

Surface mount construction and components are used throughout, with the exception of the PIC microcontroller and 5V regulator and possibly the filter components. The PIC is mounted in a socket to allow rapid changes of firmware. Larger size surface mount components are employed to allow easier home construction, the PCB pads allow the use of 1206 or 0805 devices rather than the really tiny SMT components becoming available now.

There is one wire link that needs to be installed on the board. This is the input to the internal comparator, Pin 16, on the AD9850 that is used to generate a square wave output from the filtered sine wave. If this function is wanted, connect a thin insulated wire from the pad adjacent to pin 16 to the RF output pad. If the square wave output is not needed then pin 16 should be grounded to the adjacent ground plane connection to prevent the pin floating.

The PIC Clock Source uses a 4 MHz crystal or ceramic resonator. Resonators with built in capacitors are available, in which case the two additional 33pF capacitors should not be fitted.

The decoupling capacitors are shown as 100nF on the circuit diagram. Some of these should be adjusted in value depending on the DDS Clock frequency being used. For clock frequencies above 20 MHz the four capacitors immediately adjacent to the AD9850 chip need to be reduced in values to prevent series resonances from degrading the decoupling function. Use around 10nF for 20 to 80 MHz clock and 2.2nF or 1nF for clocks above this.

Where a really accurate clock source is needed it is possible to use an accurate 5 MHz reference signal source. In this case it will be necessary to amplify the customary low level reference signal to be usable with the DDS chip. A good solution here is to feed the low level 5 MHz signal to the PIC clock input, Pin 16, via a DC blocking capacitor. Take the clock output which will be a squared up 0-5V signal from pin 15 of the PIC directly to the DDS clock input. Because the PIC clock frequency is now 5MHz you will need to use the DDSPC3F firmware in the PIC controller.

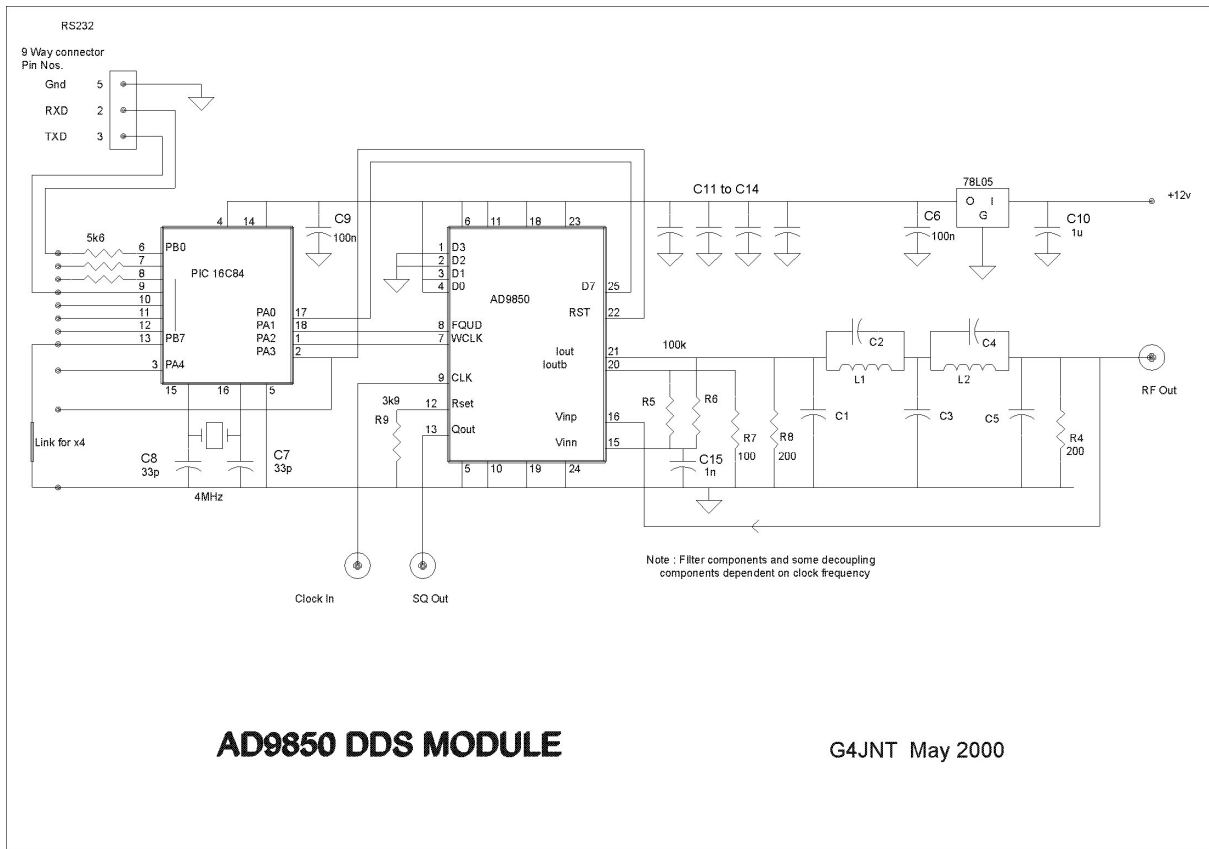


Figure 2. Circuit Diagram

There is no grounding pin / pad immediately adjacent to the clock input pin. Use the grounding pad adjacent to the squarewave output for the braid of coax feeding in this signal.

A 0-5V logic level clock is needed for the DDS clock, 74ACT, or 74F series logic families will allow logic level signals up to well over 100 MHz to be generated.

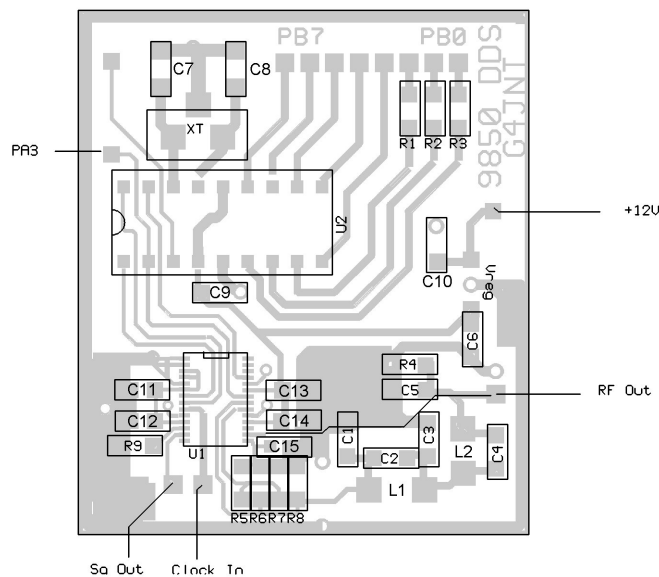


Figure 3. PCB layout

If the square wave function is not required, the two 100k resistors and associated 1nF capacitor are not required, but ensure pins 15 and 16 are connected to ground to prevent their high impedance inputs floating.

### Sample Complete designs

The component values and types which would be used for designs using a 5MHz DDS clock frequency and a 100MHz DDS clock frequency are shown below.

Ref	Description	Value		Tolerance	Package
		100MHz	5MHz		
C1	Multilayer ceramic 25 V or higher	27pF	560pF	5%	0805
C2	Multilayer ceramic 25 V or higher	1.8pF	36pF	5%	0805
C3	Multilayer ceramic 25 V or higher	47pF	1nF	5%	0805
C4	Multilayer ceramic 25 V or higher	4.7pF	100pF	5%	0805
C5	Multilayer ceramic 25 V or higher	33pF	680pF	10%	0805
C6	Multilayer ceramic 25 V or higher	100nF		10%	0805
C7	Multilayer ceramic 25 V or higher	33pF		10%	0805
C8	Multilayer ceramic 25 V or higher	33pF		10%	0805
C9	Multilayer ceramic 25 V or higher	100nF		10%	0805
C10	Tantalum 16V or higher	1uF		10%	MCCT Series
C11	Multilayer ceramic 25 V or higher	1nF	100nF	10%	0805
C12	Multilayer ceramic 25 V or higher	1nF	100nF	10%	0805
C13	Multilayer ceramic 25 V or higher	1nF	100nF	10%	0805
C14	Multilayer ceramic 25 V or higher	1nF	100nF	10%	0805
C15	Multilayer ceramic 25 V or higher	1nF		10%	0805
R1	Resistor. Metal film 1%; 1/4 or 1/8W	5.6kΩ		1%	1206
R2	Resistor. Metal film 1%; 1/4 or 1/8W	5.6kΩ		1%	1206
R3	Resistor. Metal film 1%; 1/4 or 1/8W	5.6kΩ		1%	1206
R4	Resistor. Metal film 1%; 1/4 or 1/8W	200Ω		1%	1206
R5	Resistor. Metal film 1%; 1/4 or 1/8W	100kΩ		1%	1206
R6	Resistor. Metal film 1%; 1/4 or 1/8W	100kΩ		1%	1206
R7	Resistor. Metal film 1%; 1/4 or 1/8W	100Ω		1%	1206
R8	Resistor. Metal film 1%; 1/4 or 1/8W	200Ω		1%	1206
R9	Resistor. Metal film 1%; 1/4 or 1/8W	3.9kΩ		1%	1206
XT	Crystal	4MHz			HC49/U
L1	Inductor	1uH	22uH	10%	
L2	Inductor	1uH	22uH	10%	
U1	AD9850				28SSOP
U2	PIC	PIC16F84-04P			18DIL
VREG	Voltage regulator	ZR78L05C			3/TO-92

### Interconnections

The two RS232 control lines are configured as follows. Tx Data from the PC is applied to the PIC through a resistor to limit the current drawn. The voltage excursions are clipped to 0 / 5V by integral diodes with the PIC. The RS232 output, PC Rx Data, is at 0-5V levels rather than the more normal +/- 12V for this signalling protocol. This was done to avoid the complication of adding a level converter chip. No PC Com Port has ever been found which does not work perfectly well with these voltage levels at up to 115200 Baud.

The X4 / X1 link contains an integral pull up resistor and the default state is high, at X1 output frequency. Other PIC lines are unused in the present design and are available for custom user software, two have provision for input resistors to allow connections to RS232 control or handshaking lines. All unused PIC ports are configured as outputs to prevent them floating if left unconnected

### Software Control

Control is via a bidirectional RS232 link with no handshaking. Parameters are 19200 baud, 8 data, No parity, 1 stop bit.

Commands are sent using ASCII / Hex characters, so that a simple terminal programme such as HYPERTRM or PROCOMM can be used to command the frequency source. Set this to 19200 N81, Full duplex, no flow control and all start up / modem commands set to null. Carriage return / linefeed pairs are returned, but where needed, a single [cr] is all that is required for commands.

Alternatively, custom software can be written to drive the COM port with the commands.

In all the following command descriptions , [cr] carriage return is the hex character \$0D, decimal 13, and [lf] linefeed is hex \$0A, decimal 10.

The first character sent is a Board Address which precedes all commands. This is a single Hex character sent as ASCII 0 – F and potentially allows up to 16 modules to be driven from the same COM port.

The next character is a command which may have hex data following it.

- Q Followed by eight hex digits for the frequency command word terminated by a carriage return
- P Followed by two hex digits for phase word and [cr]
- U Writes the data sent above to the AD9850 DDS chip
- W As for U, but also stores the data in the PIC's non-volatile EEPROM memory for switch on next time.
- Y Followed by one Hex digit, changes the board address and stores in EEPROM. No [cr] needed
- K Followed by 10 hex digits & [cr].  
(In practice read as decimal number for user data, typically clock frequency)
- R Readback current data values – not necessarily those in EEPROM

No echoing is provided of typed characters, the simple UART implemented in the PIC would run into trouble if commands were sent too fast, see below for control limitations.

The 32 bit or 8 hexadecimal character, value N required for frequency setting can be derived from :

$$N = F_{out} / F_{clock} * 2^{32}$$

Phase can be set to any one of 32 values in increments of 11.25 degrees. These form the five highest significant bits of the phase word Pxx. The lowest three bits are ignored.

The following codes are sent back from the board to the terminal / software :

At switch on, the startup string : **9850 DDS Controller Addr. \* [cr] [lf]**

Is returned. (\* is the board address) followed by the default data stored in EEPROM in the format:

**Q xxxxxxxx Pxx [cr][lf].**

This data is loaded into the AD9850 for automatic power on startup. Finally the user K data is sent back in the format **Kxxxxxxxx[cr][lf]**

After the address byte, a single character , **Z** , is returned to acknowledge correct addressing, followed by **[cr][lf]**

After the [cr] terminating Q or P hex data , the complete data is echoed back in the form :

**Q xxxxxx Pxx [cr][lf]** Note that at this stage the data has not yet been programmed into the synthesizer

Data shifts in from right to left, LSB to MSB, so if more than eight bytes are sent, the last 8 form the frequency data, (or last 2 for phase data, last 10 for K data). If less than the required number, the data is right justified.

For address changes, the complete start up string is returned with the new address appended.

If the X4 / X1 input select is toggled, the relevant state is returned in the form **X1[cr][lf]** or **X4[cr][lf]**

Example commands, assuming a conventional ASCII terminal and Board Address 5 :-

Characters sent, or action	Echoed Back	Function
(Switch on)	9850 DDS Controller Addr. 5 [cr] [lf] Q xxxxxxxx Pxx [cr][lf] K xxxxxxxxxx[cr][lf]	sign on message and stored default data
5 Q54FB1200[cr]	Z[cr][lf] Q 54FB1200 Pxx [cr][lf]	frequency data setup, but not sent
5 P45[cr]	Z[cr][lf] Q 54FB1200 P45 [cr][lf]	phase data setup but not sent
5 U	Z[cr][lf] Q 54FB1200 P45 [cr][lf]	programmed into DDS

Characters sent, or action	Echoed Back	Function
5 Y6	Z[cr][lf] 9850 DDS Controller Addr. 6 [cr] [lf]	change and save address

6 W	Z[cr][lf] Q 54FB1200 P45 [cr][lf]	note that the address has now changed write current data to EEPROM
6 Kxxxxxxxxx[cr]	Z[cr][lf] K xxxxxxxxxxx[cr][lf]	saves user data into EEPROM
6 R	Z[[cr][lf] K xxxxxxxxxxx[cr][lf] Q 54FB1200 P45 [cr][lf] Addr. 6 [cr] [lf]	readback all current data
(X4 Input = 0) (X4 Input = 1)	X4[cr][lf] X1[cr][lf]	X4 control pin set

### Command Timing

Due to the simple way the serial interface, or UART, is implemented in the controller PIC, it is not possible to send a command whilst any characters are being echoed back. This is not a problem when using manual control from a terminal programme, as natural typing delays are more than adequate to allow the completion of all data returned.

Problems can arise however, in software designed to control the board in real time. The following guidelines must be adhered to in writing software to control the module : (timings are worst case and rounded to the nearest millisecond higher)

After switch on wait for the initial startup delay of 300ms, and for the complete 58 character startup message to be returned.

After the address is sent, wait 2ms for the Z to be returned. No Z means the address was wrong or the serial link is not operational.

The Qxxxxxxxx[cr] string can be sent contiguously as no echoing is performed. After the final [cr] wait for the returned data string which is 18 characters and will take 11ms.

5        wait 2ms for Z  
Pxx[cr] command, as for Q...

5        .....  
U        wait 11ms .....

5  
R        wait 22ms

Ideally the returned string would be parsed and checked for validity, but this is not essential once any driver software has been debugged, provided the minimum wait times above are adhered to. Checking for the Z only, allows command acceptance to be confirmed. Note the locations of spaces in the returned strings. The easiest way to get an exact picture is to test with a conventional terminal programme and see the exact format returned.

### Manual Updating

Instead of issuing the U command to write data previously sent to the DDS, toggling of the X1/X4 input will also update the chip; either a high or low going transition will work. There is a fixed delay of approximately 17ms after the transition before the command is sent. This was not an intentional part of the software, but is an incidental function of using the Port B change on interrupt to monitor this pin.

Later versions of the PIC software will probably include this function intentionally on one of the unused PIC pins, PB4 – PB6. In this event, the time delay will be controlled precisely to allow for eg. GPS time controlled setting. Also, only the positive edge of the trigger pulse will be acknowledged.

### EEPROM Data

The 16F84 stores start up data, user data and the board address in its internal EEPROM registers. User data is stored in locations \$00 to \$04, the frequency word is stored in registers \$05 to \$08, the phase data in \$09 and the board address in \$0A. When blowing the firmware into a chip, it may be worth setting these to initial known values, particularly the chip address value.

Five bytes of hex user data can be stored. For easy readability, It is suggested that this consists of the clock frequency in Hz stored in 'packed BCD' format so it is read out as hex data, although consisting of decimal digits. The hex D

character can be used as a decimal point. For example, a clock of 101.234569 MHz would be represented by the hex data \$10 \$12 \$34 \$56 \$7D which would be read out as 101234567D or 101234567 Hz. A clock of 4.9999983MHz would be represented by 49 99 99 9D 83

### Low Pass Filter Design

The fifth order elliptic given here represents a compromise between attenuation and fast roll off suitable for this application. The component values tabulated below result in a filter with a cut off frequency of 33MHz and an ultimate rejection of 60dB, occurring initially at a value ( $F_s$ ) of 70MHz. This ensures that for operation of the DDS at up to 33% of a clock frequency of 100 MHz, all DDS generated spuri will be at least -60dBc. A clean, but attenuated output is available at up to 40% of the clock.

By scaling the values given below an equivalent filter can be generated for any desired cut off frequency. For a different filter, perhaps optimised to a particular range of frequencies, consult one of the many books on filter design such as the Handbook of Filter Synthesis, by Zverev. Alternatively, tables of elliptic filter values are given in the ARRL Handbook and these may be scaled for a wide range of useful frequencies. When designing from first principles, note that the characteristic impedance of the filter used here is 200 ohms. This allows the DDS to 'see' an impedance of 100 ohms and have the filter correctly terminated at each port. All capacitor values are taken to the nearest E24 values

*Standard elliptic filter component values for  $F_c = 33\text{MHz}$ , (suitable for a clock frequency of 100 MHz) showing 60dB ultimate rejection starting at 70 MHz.*

C1	L1	C2 (series)	C3	L2	C4 (series)	C5
27pF	1.1uH	1.8pF	47pF	1uH	4.7pF	33pF

To generate values for a different value of  $F_c$ , simply multiply the values in the table by the ratio :  
 $33\text{MHz} / F_c$  (or by the ratio  $100 \text{ MHz} / F_{\text{clock}}$ )

For example, a clock frequency of 5MHz and  $F_c = 1.6 \text{ MHz}$  gives filter component values :

C1	L1	C2 (series)	C3	L2	C4 (series)	C5
560pF	22uH	36pF	1nF	20uH	100pF	680pF

A plot of the frequency response for the 100MHz clock filter is given below.

The DDS module is designed to feed into a high impedance buffer amplifier or load and the output filter is therefore matched to 200 ohms on both sides. If a 200 ohm input buffer amplifier is used, remove R4. Alternatively, if you want to try designing from scratch, filter designs are available for unequal impedance transformation which may prove of more use for embedded designs. See Zverev or other filter design handbooks for suitable designs

### Use with aliased frequency products.

Theoretically, frequencies up to a half of the clock rate can be generated, this limit being set by the rules of aliasing and Nyquist sampling. In practice this frequency cannot be reached due to the need to filter out spurious components and hence the 35 to 40% of  $F_c$  normally quoted is the practical upper limit. A very high order filter may well allow operation to 45% of  $F_c$  or higher.

A characteristic of DDS frequency generation is that other products are generated as well as the wanted component. These are normally removed by the filter, but in some cases can be used in preference to the normal one. The main component is the image response, a frequency equal to the clock frequency minus the normal commanded value. Thus for a 100 MHz clock, a programmed frequency of 12 MHz will also give an output at  $100-12 = 88 \text{ MHz}$ . Tuning will of course be in the wrong direction as  $F_{\text{out}} = F_{\text{clock}} - F_{\text{set}}$ . Bandpass filtering is obviously needed here as the lower 'normal' frequency has to be removed as well as the clock and other higher components and use in this way will normally be restricted to narrow band operation. Typically for a 100 MHz clock, frequencies from around 60 to 90 MHz should be realisable. There are another pair of frequencies available above the clock, given by  $F_{\text{out}} = F_{\text{clock}} + F_{\text{set}}$  and  $F_{\text{out}} = 2.F_{\text{clock}} - F_{\text{set}}$  and so on, giving access to 110 – 140 and 160 – 180 MHz, again for a 100 MHz clock. These higher frequency terms will progressively reduce in amplitude due to the  $\text{SIN}(X) / X$  envelope for sampled waveforms and the frequency response of the D/A converter within the DDS chip and have not been tested.

*Frequency response of the 5<sup>th</sup> order elliptic filter with  $F_c = 35\text{MHz}$ ,  $F_s = 70\text{MHz}$  and  $A_{\text{min}} = 60\text{dB}$  (Frequency scale logarithmic 20MHz to 250MHz)*

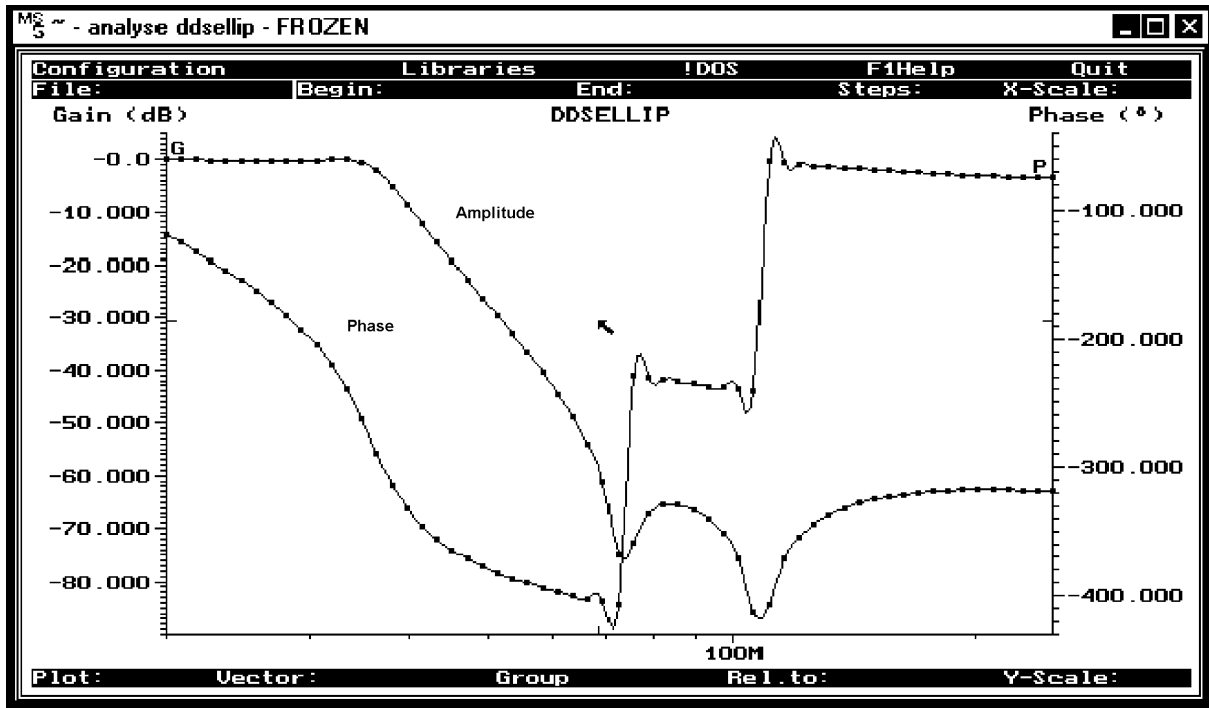


Figure 4. Plot of Filter response