AD9852 Direct Digital Synthesiser module and driver software

Andy Talbot G4JNT November 2004.

Overview

This unit is a complete PCB based module capable of generating frequencies from zero to about 100MHz with virtually zero channel step size. It is based around the Analog Devices AD9852 DDS chip which is a very versatile device. As well as the basic frequency generation function, the output can be stepped in phase to 14 bit resolution, automatic fast Frequency Shift keying is built in, as well as the possibility of pre-programmed chirp functions and output waveform shaping.

The AD9852 chip is controlled by pre-programming its internal registers via a serial or parallel bus input, then issuing an update pulse whereupon the pre-programmed changes take immediate effect. This module contains a PIC microcontroller to allow straightforward interfacing through an RS232 serial link with a Personal Computer running a simple ASCII terminal programme. Writing custom software for data modes, for example to control the frequency and phase of the DDS via this route becomes straightforward as only text now need be sent on the serial interface. A command is available to allow the update to be synchronised to an external timing edge such as that from GPS receiver.

Simple commands are included to set frequency, phase and amplitude, and to save these to non-volatile memory so the DDS starts with the same values when it is next turned on. Commands are also available to change each individual chip register to give full access to the device's capabilities. Full details of the command protocol are contained at the end of this document.

Before using this module, please obtain the AD9852 data sheet, available from www.analog.com or directly from myself, and read it thoroughly before use. When programming the registers, it is possible to overheat the device by clocking it much too fast (Having said that, I once accidentally operated the device at a clock frequency in excess of 500MHz and it was working correctly, but drawing 1 Amp and getting very hot. Such operation is not recommended!)

Module Design

The Circuit Diagram is shown in Figure 1. A single PCB contains the DDS chip with its peripheral components for correct operation, space for a crystal oscillator / TCXO module to form the clock (an external clock such as a 10MHz high stability reference can also be used), the PIC for the operating system and power supply / decoupling components.

The AD9852 chip requires a 3.3V power supply, and when operating at its maximum clock frequency of 300MHz consumes around 600mA, so an off board regulator mounted on a heatsink is essential. The PIC runs from a 5V supply, generated from a separate low power regulator.

Interfacing to the RS232 serial link uses the 'unofficial' 0/5V levels that appear to work with every serial port tested to date. This ensures the simplest connection possible to interface the PIC to the RS232 port - just a single resistor to limit current flowing into the pin from the computer Transmit data (TXD) connection.

The PIC has the connections for in circuit programming brought out to a separate header, so there is no need to mount this in a socket if reprogramming for new functionality or operating system update is foreseen.

The AD9852 can generate frequencies up to approximately 0.35 times the clock frequency; the clock can go up to 300MHz if the chip has had its base slug soldered to the PCB, so frequency generation up to about 105MHz is achievable. An on board Phase Locked Loop multiplier allows the clock frequency to be generated from a lower input reference. This PLL can be programmed to multiply by a factor between 4 to 20 times, so for a 300MHz reference a clock oscillator of 15MHz can suffice. The PLL can also be switched out and the clock supplied directly for optimum phase noise performance. The module design has biasing included on the clock input pin to permit AC coupling of any signal - subject to the limitations shown in the data sheet.

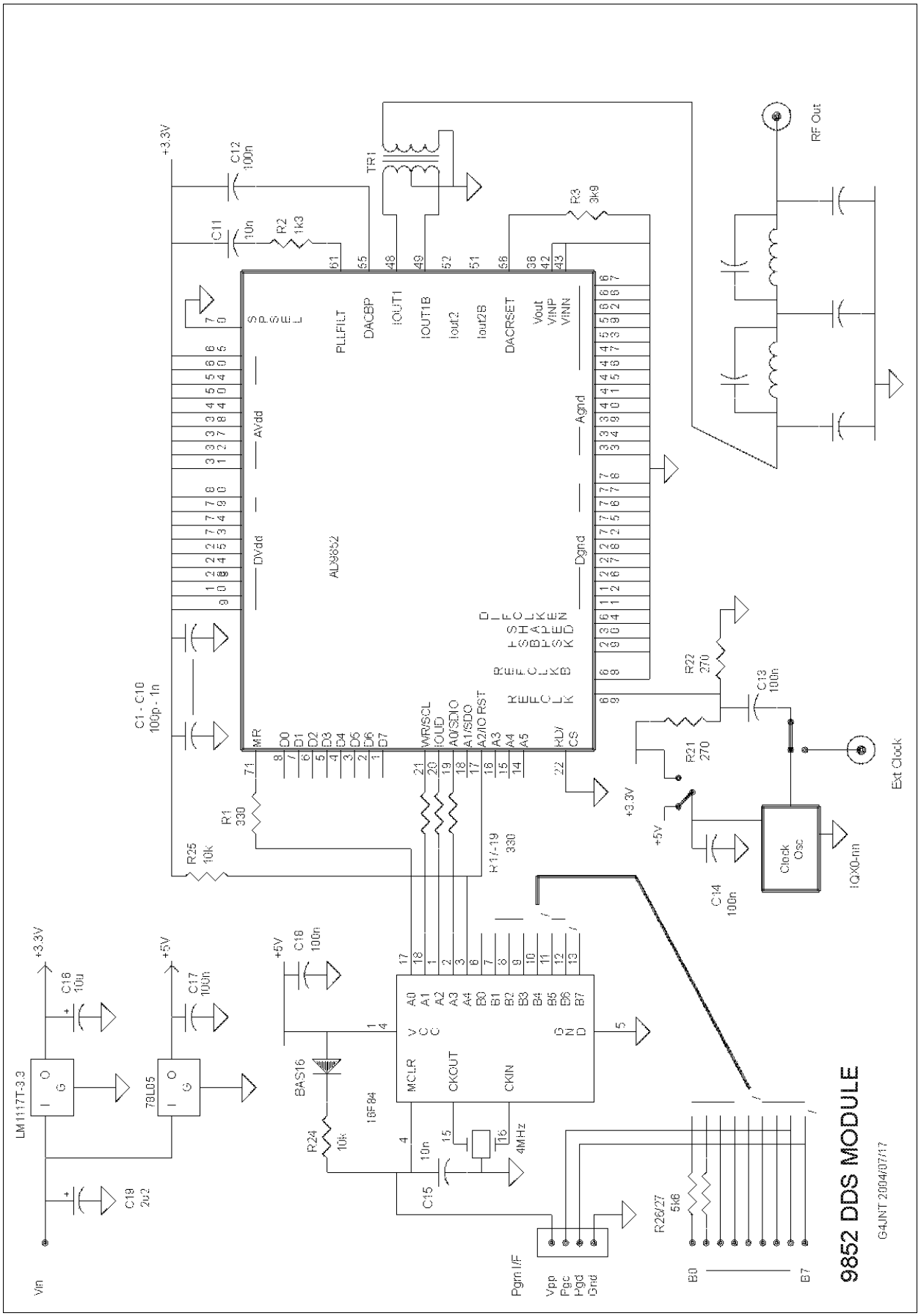

9852 DDS MODULE
G4JNT 2004/07/17
AD9852
PLLFILT
DACBP
IOUT1
IOUT1B
Iout2
Iout2B
DACRSET
Vout
VINP
VINN
Agnd
Dgnd
AVdd
DVdd
SPISEL
MR
D0
D1
D2
D3
D4
D5
D6
D7
WR/SCL
IOUD
A0/SDIO
A1/SDO
A2/IO RST
A3
A4
A5
RD/
CS
REFCLKB
REFCLK
DIFSCHLKBPEENKD
TR1
C12 100n
C11 10n
R2 1k3
R3 3k9
RF Out
C1 - C10 100p - 1n
R1 330
R25 10k
R11/-19 330
R22 270
R21 270
C13 100n
C14 100n
Clock Osc
IQX0-nin
Ext Clock
+3.3V
+5V
LM1117T-3.3
78L05
C16 10u
C17 100n
C18 100n
C19 2u2
Vin
BAS16
R24 10k
16F84
MCLR
CKOUT
CKIN
VCC
GND
A0
A1
A2
A3
A4
B0
B1
B2
B3
B4
B5
B6
B7
C15 10n
4MHz
Pgm I/F
Vpp
Pgc
Pgd
Gnd
R26/27 5k6
B0
B7

Figure 1 AD9852 DDS Module Circuit

Maximum output power from the AD9852 is obtained if both complementary outputs are used driving a transformer in push-pull. 6dB or less is achieved if one output is directly connected to the load resistor, both outputs need to have a DC connection to ground. The widest frequency range is achieved if a balun type winding with a trifilar winding is used in this position. With the right choice of ferrite material, a transformer frequency response from below 1MHz to 200MHz is possible.

With the specified value of 3.9K on pin 56, output power is typically +3dBm from this push-pull configuration. The value of this resistor can be lowered to increase power output, to a limit of 2k. See the data sheet for more details.

Output filter.

The design of a suitable filter will depend on the clock frequency and wanted output range. Pads have been provided on the board for a 5 section elliptic filter with some values for a range of cut off frequencies given in Table 1. A simulated response for the 60MHz version in shown in Figure 2 A different filtering solution, for example bandpass or Chebyshev response may offer advantages when operating over a restricted frequency band

Cut Off MHz	C1 pF (Shunt)	L1 nH	C2 pF (parallel)	C3 pF (shunt)	L2 nH	C4 pF (parallel)	C5 pF
100	43	94	4.6	62	76	13	36
80	53	117	5.7	76	95	16	45
60	71	156	7.6	102	127	21	60
40	107	234	11.4	153	190	32	91
20	142	312	15	204	253	43	121

Table 1 Values for 5th order elliptic anti-alias filter.

Figure 2 Frequency response of 60MHz filter from Table 1

Clock Source

A 40MHz packaged crystal oscillator has been supplied. This can be used as-is for a 40MHz clock, or the PLL can be programmed to generate a clock of 160, 200, 240 or 280MHz

The pads on the PCB fit other alternative packaged crystal oscillators or TCXOs. Connections to these are (usually) standardised so most types should fit. The pad layout is optimised for SMT types, but wire ended packages can be installed by bending their leads to fit. A wire link will need to be fitted to either the 3.3V or 5V power supply as desired.

An alternative is an external source, and space has been provided on the board for inputting this and coupling into the DDS chip. Consult the data sheet for allowable levels and frequencies, but basically anything from near DC to 300MHz should prove suitable with an input level of a between 0 to +10dBm. If using an external source at a high frequency, be very sure about the PLL programming. It is easy to overclock the device with the potential for overheating and damage.

Using a DDS and avoiding Spurii

A DDS is certainly not a final solution for ultra clean wide band frequency generation. All DDS chips generate spurious outputs and these have to be considered carefully. The raw DDS process generates a fundamental frequency component by a reverse sampling process, where the clock is the sampling frequency. The same rules about sampling and Nyquist apply here as to conventional sampling – the absolute maximum frequency that can possibly be generated is equal to half the clock. Alias products are also generated at frequencies equal to N.Fc +/- Fout where Fc is the clock, and N is an integer. So for a clock of 200MHz generating an output at 70MHz, there will also be signals at 200 – 70 = 130MHz, 2 * 200 – 70 = 330MHz, 470Mhz and so on. These roll off with a SIN(x) / x shape, but the first ones are still high enough to need filtering out. So here is the reason for the approximately 0.35.Fc upper limit. At this frequency, the first alias product falls at 0.65Fc, so an output filter with a cut off just above 0.35Fc has nearly an octave in which to reach the desired rejection at 0.65Fc. The 5th order elliptic illustrated can achieve 50dB rejection here.

However, alias products are not the only spurii generated. Imperfections in the Digital to Analogue conversion and sine look up tables within the DDS chip cause harmonics to be generated – but these are not just normal harmonics, they are themselves aliased. The result of harmonics that fall above the Nyquist frequency is that they are folded back into the lower frequency band and appear, often where they are not wanted. The AD9852 data sheet lists quite a comprehensive set of spurious levels, but in general a figure of around –50dBc is guaranteed. However, if a component at –50dBc falls in band it is probably going to be troublesome. The following example illustrates this.

Wanted Fout	=	50.02MHz		(for a 6m beacon ,say)
Fclock	=	200MHz		(ought to be ideal, generate from 10MHz ref. input))
Third harmonic	=	150.06MHz		
aliased product	=	200 – 150.06	=	49.94MHz

Which is very close, impossible to filter out and could be as high as –50dBc

The solution is to choose clock and output frequencies together. Since a DDS has virtually infinite frequency resolution, and the PLL clock multiplier can be programmed over a range of frequencies, alter the clock to avoid situations like this. An example might be to use a lower clock like 180MHz. Now the aliased third harmonic sits at 180 – 150 = 30MHz (approx), the aliased second harmonic at 180 – 50 = 130MHz so we should be OK.

For a simple CW generator where the wanted output is less than 10 - 20% of the clock, harmonic aliassing is rarely a problem. BUT, for microwave use where a fundamental frequency will be multiplied many times even very low levels of close in spurii can be disastrous. An effect of frequency multiplication is that spurious sidebands increase by the square of the frequency multiplication factor.

So as an example, take a 10GHz beacon source starting off with a raw DDS generated signal at 54MHz which is multiplied by 192 to 10368MHz.

Close in spurii will be magnified by 192^2 which is approximately 46dB. So that –50dBc specification has now become almost equal to the wanted signal. If by a careful choice of clock, you manage to avoid any terms that high, even a –80dBc spurious, undetectable by most spectrum analysers will still give a result approaching 50dBc. –80dBc spurii on the generated DDS output are numerous and there are probably tens within a close range – all of which will be multiplied up and intermodulate / mix with each other to give hundreds or even thousands of potential –40 to -50dBc terms, which could add up and manifest themselves as a 'mush' around the wanted output. This is not good!

The solution is to use a PLL to filter out just the wanted signal from among the close in products. For a microwave beacon a simple PLL based around a 108MHz VCXO works admirably (see reference 1) and allows all products more than a few hundreds of Hz away from the carrier to be removed, but even an LC oscillator can do a useful job for lower frequencies.

The real value of a DDS lies is in its very fine frequency resolution and low inherent phase noise, adding practically nothing extra to the reference clock used.

Using other alias products

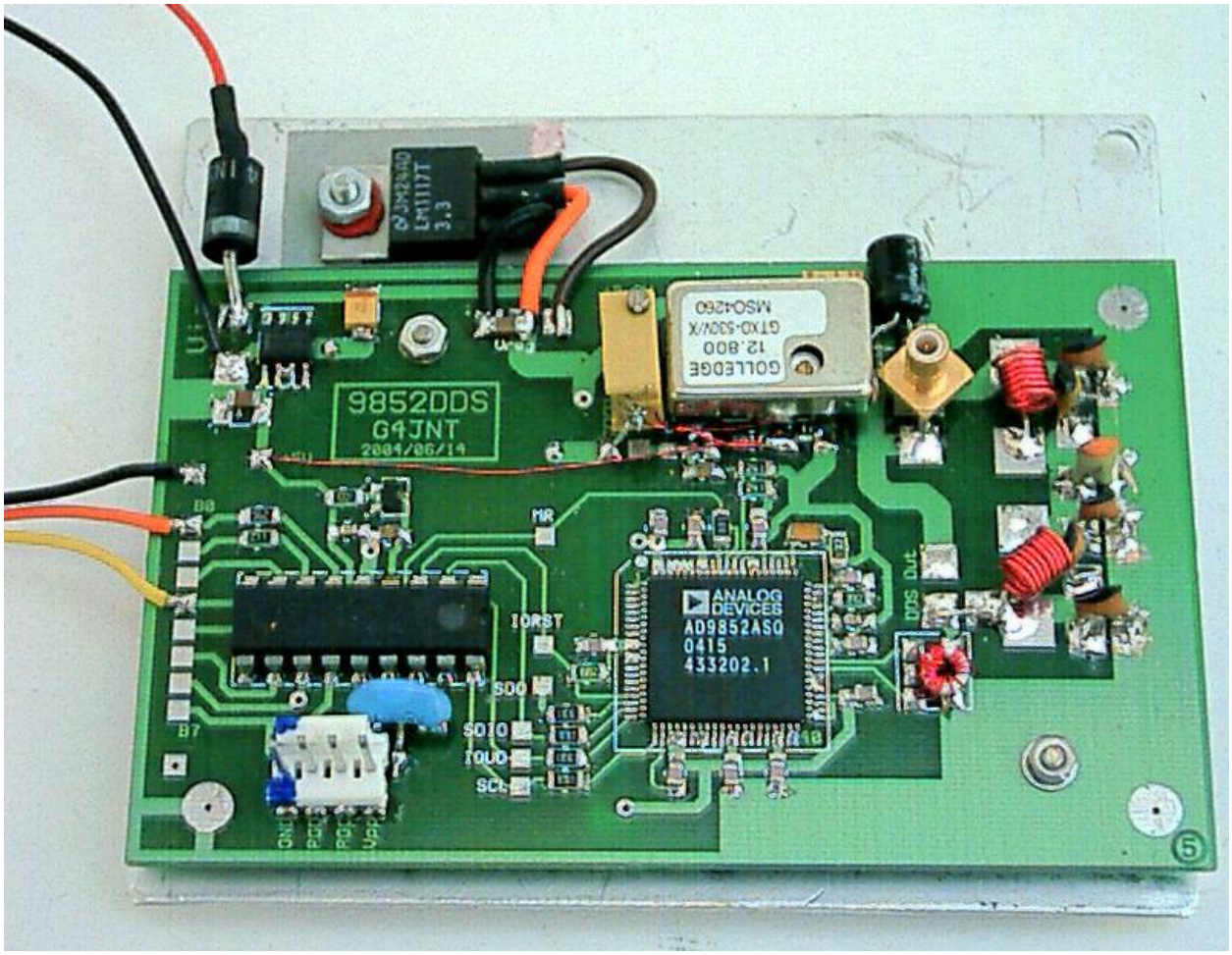
By a careful choice of clock and output filter, other alias products can be selected for a higher frequency output. Obviously a bandpass filter will be required here to eliminate products above and below the wanted one. An area to watch out for when using this scheme is the SIN(X)/X roll off. This means that the higher order products will have a significantly lower amplitude than the fundamental, making filtering of the wanted product even tighter. The following example for a personal 70MHz beacon using the older AD9851 DDS chip illustrates the idea.

Wanted F_{out}	70.029 MHz		
F_{clock}	60 MHz (From 10MHz reference, X6 in PLL)		
Fundamental	10.029MHz		0 dB (reference level)
1st Alias	60 - 10.029 =	49.971MHz	-14 dB
2nd Alias	60 + 10.029 =	70.029 MHz	-17 dB
3rd Alias	2.60 - 10.029 =	109.971MHz	-21 dB

So a bandpass filter is needed that can reduce the 1st and 3rd alias products to a suitable level.

Components List

1	R3	3k9
2	R24,R25	10k
1	R2	1k3
4	R1,R17-19	330
2	R21,R22	270
2	R26/27	5k6
1	C16	10u
1	C19	2u2
10	C1 - C10	100p - 1n
5	C12,C13,C14,C17,C18	100n
2	C11,C15	10n
1	D1	BAS16
1	U1	PIC 16F84
1	U2	AD9852ASQ
1	U3	78L05
1	U4	LM1117T-3.3
1	X1	4MHz Resonator
1	TR1	See text



One version of the finished module. The multiturn preset resistor and wire ended capacitor were added to provide fine frequency control of the TCXO. Note the idiot diode on the DC power input just in case.

AD9852 DDS Module controller software.

PIC S/W Reference 9852DDS (2004/07/22) and 9852DDSX (Nov 2004)

Overview

The DDS module is supplied with a PIC microcontroller that contains code to allow the AD9852 to be controlled via an RS232 serial link from an ASCII terminal such as a PC running Hypertrm software. A further input to the PIC can be used for an external trigger, so the updating of the DDS output can be synchronised to an external event such as a UTC pulse from a GPS receiver.

Setting up the serial link

Set your terminal programme to 19200 baud, 8 data bits, no parity and 1 stop bit (19200 N81). Turn off local echo and do not enable CR-LF translation on receive. Connections between the PC and the DDS module are defined in Table A1

Connection	9 Way D type Pin	PIC Connection
TXD	3	Port B0 (via resistor)
RXD	2	Port B3
Gnd	5	Gnd

Table A1 RS232 interface connections

Connect the serial link and switch on the DDS module. After about half a second delay, a display similar to that in Table A2 will appear.

```
9852 DDS Controller G4JNT
Qxxxxxxxxxxxx[cr]   Pxxxx[cr]  Mxxxx[cr] U  W  V G  K  R

0 = 00 00
1 = 00 00
2 = 40 00 00 00 00 00
3 = 00 00 00 00 00 00
4 = 00 00 00 00 00 00
5 = 00 00 00 40
6 = 00 00 00
7 = 10 64 00 60
8 = 0F FF
9 = 00 00
A = 00
B = 00 00

240000000
```

Table A2 Start up display

All the registers in the AD9852 chip are loaded with the values stored in EEPROM, either the default initial values, or any that have been subsequently changed by the G or W commands (see below). As supplied the default will result in an RF output at exactly 0.25 times the clock frequency, with no PLL multiplier in use and maximum output amplitude. All other registers are the same as the manufacturers default start up settings.

Controller Software

The AD9852 DDS chip has quite a comprehensive set of capabilities which are all controlled by writing appropriate values to its working registers then triggering (updating) the device. The controller software has two main functional capabilities.

For simple CW frequency generation , a straightforward command structure for quickly updating CW frequency, phase and amplitude is implemented, with separate commands to update the DDS with these changes either immediately, or on an external pulse edge. The current frequency and amplitude settings can optionally be written to non-volatile storage (internal EEPROM) for immediate start up next time the module is turned on or reset.

The other mode of operation allows any of the internal registers to be written individually, giving full access to the chips functionality. All register contents altered using these commands are stored in EEPROM and loaded in the next time the module is turned on.

A final option is available to allow users to store a string of up to 15 characters in EEPROM for reading back on the serial link; they perform no action on the DDS chip itself. This can be used, for example, to store the clock frequency so any DDS driver software can read this value back, and so be used with several different modules, each with their own clocks.

Commands

Immediate programming

These use commands Q, P, M, U, T and W

Qxxxxxxxxxxxx[cr] Sets a new frequency of the single carrier output. The format must be exactly as shown with xxxxxxxxxxxx replaced by hexadecimal ASCII characters. eg. Q0280000000AF[cr]. The new frequency is programmed into the AD9852 (although not into EEPROM) but does not take effect immediately.

If the command is recognised, the controller responds the code entered followed by Q[cr][lf] eg 0280000000AFQ[cr] *This command controls the contents of DDS register 2*

Pxxxx[cr] Sets the phase of the RF carrier output. The format must be exactly as shown with xxxx replaced by hexadecimal ASCII characters. eg. P1000[cr]. The new phase is programmed into the AD9852 (although not into EEPROM) but does not take effect immediately. The most significant two bits are ignored so P1000 has the same result as P9000 (but not the same as P0000)

The characters are not echoed back to the terminal, so when typing in by hand this has to be done blind. If the command is recognised, the controller responds with the code entered followed by P[cr][lf] *This command controls the contents of DDS register 0*

Mxxxx[cr] Sets the amplitude of the RF carrier output. The format is the same as for the P command, for example M0FC0[cr]. Only the least significant 12 bits are recognised, the first character is ignored so MFFFF gives the same result as M0FFF. Maximum amplitude is given by a value of FFF, and a value of 000 shuts off RF output completely. If the command is recognised, the controller responds with the code entered followed by M[cr][lf*] This command controls the contents of DDS register 8*

U Updates the DDS chip with the new P, Q or M values set in the above commands, or updates registers that have been changed with the G command. When complete, the controller responds by sending Z[cr][lf] No carriage return is needed after any single letter commands

T Triggers the controller to wait for a positive edge on the external timing input (Port B1) before updating the values from the P, Q or M commands. 'Z'[cr][lf] is returned when the update is complete. Note that while waiting for the positive timing edge, the controller will be deaf to any further serial commands and may have appeared to 'hang'. This situation has to be checked for in any driver software by looking for the acknowledgement Z before issuing any further commands. There is no way out of this hang-up, apart from forcing a positive edge on B1 or by resetting the processor.

W Writes the current frequency and / or amplitude entered from the last Q or M command to EEPROM. This command will only be accepted after a Q or M command has been issued. If OK, the controller responds with a 'Z'

Register Programming

These use commands G and V, and also require the U command to transfer data to the DDS itself

The individual AD9852 registers can be updated one at a time and the values are always stored to EEPROM for immediate start up. The AD9852's registers have different lengths depending on their function, from one to six bytes in length. There is plenty of scope for incorrect operation and unexpected results, particularly when programming the control register. No error checking is performed within the controller so read the data sheet carefully before changing registers! If you get into a serious mess, change all the registers to a known working state, such as those shown in Table 1 and start again.

The **G** command is used to update an individual register:

Type **G** and the controller responds with '*Reg No*' No [cr] is needed after the G

Enter a value from **0** to **9**, **A** or **B**. There is no need to enter a carriage return.

The controller responds with '*x bytes of Reg. Data*' , where x is a number from 1 to 6.

Enter the data in hexadecimal (with no [cr]), and as soon as the final character of the requisite number is entered, the controller will respond with '*Z*'. (Note, one byte of data requires two characters, 6 bytes requires 12 characters).

The new value is immediately written to both the AD9852 register and the appropriate EEPROM register, but the AD9852 is not updated, so the changed value will not have any immediate effect.

After register changes are completed, a **U** command then has to be sent to update and transfer the register to the AD9852 internals to activate the change

The **V** command is used to dump the entire EEPROM contents , in the format shown in the central portion of Table A2.. It also loads and updates the AD9852 with the data. Be aware that using this command overwrites any values changed by the Q, P or M commands, unless a W command has been issued to save them.

User Data

These two commands allow up to 15 characters of user data to be stored and read back from EEPROM.

The **K** command allows data to be entered.

Type **K** and the controller responds with '*Enter < 15 chars. of user data*'

Enter the characters required, followed by a [cr] to terminate. Any ASCII character is accepted, and all letters are converted to upper case. If an attempt to enter more than 15 characters is made, the controller responds with '*Overflow*' and the first 15 entered are accepted.

In the interests of conformity, if this data is used for clock frequency it is suggested this takes the form of, for example, '240000000.00HZ' or '240.00MHZ' so the readout is meaningful and can be easily read and interpreted by driver software.

The **R** command reads back the user data

Issue **R** (with no carriage return) for the controller to respond with the stored string

Later versions of the controller software may include extra commands. Command letters will always be chosen so that there can be no ambiguity with hexadecimal data, hence commands A-F will not be used in any future versions.

9852CALC.EXE Frequency Setting Software.

A simple Windows 98 programme is available for controlling the frequency of the module, although it can also be used just to calculate the hex code, given clock and wanted frequency. The software has been written in Visual Basic and the complete installation is too big to fit on a floppy disk – contact me direct for a copy of the full installation suite

To use the programme, connect the module to the RS232 interface and start the programme. If the module cannot be found on the interface, a message window pops up to inform you of this. The software automatically reads the user data from the PIC and, provided this has a valid number followed by the text "MHZ" in it (eg. 160.000MHZ), the value is interpreted as a clock frequency. Any other value for the clock can be entered in the box.

The frequency that is programmed into the DDS can then be updated by typing it into the appropriate box, checking HZ/kHz/MHz as appropriate and clicking the update button. If the automatic update is checked, the frequency from the device will change as soon as it is entered in the window. The up-down arrows can be used to change the frequency in steps of the value entered in the step box.

The frequency code can be written to the non-volatile EPROM storage by clicking the appropriate command button.

At this stage no other options are available with this programme, and to change any other registers such as phase, amplitude, PLL mulitplier etc, the ASCII terminal route will have to be used.

This programme is an embryonic piece of software (my first venture into VB programming) and updates will rapidly follow. Once the full 1.6meg is installed, any future updates will only require the single .9852CALCEXE file to be updated.