SERIAL (RS232 Format) INTERFACE FOR THE 'Ebay SPECIAL' AD9850 DDS MODULE

Andy Talbot G4JNT 4 March 2014

The code ***9850_Ser*** for the PIC 16F628 allows a AD9850/9851 DDS chip to be set to generate any frequency by entering the actual value in decimal using ASCII characters on a serial COM port interface. This contrasts with my original AD9850 control code (***DDSPC*** and its derivatives) that required the register values to be entered in hexadecimal format with a complex addressable handshaking protocol. Thus this new version is of more general use, and is now directly compatible with the synthesiser drive commands delivered by Murray, ZL1BPU's 's WSQ software (see the section on WSQ at the end)

This simpler ASCII text based format means that to set the module to generate, say, 475450.4 Hz you simply send it the string FRQ475450.4[cr] on a 9600 baud RS232 (COM Port) interface.

A 16F628 PIC is used with a 4MHz crystal or ceramic resonator to define the timing. Unfortunately, the internal RC oscillator in this chip is uncalibrated, so its use for RS232 baud rate timing is not possible. Figure 1 shows the circuit diagram and connections to the DDS module. A higher resolution version can be found in file ***SER_SPI.GIF***

The PIC code is contained in ***9850_Ser.ASM*** [1] which contains default settings for the 125MHz DDS Clock as supplied with the Ebay modules. This is the value stored in the file ***9850_Ser.HEX*** supplied for directly blowing into a PIC device. For any other value of DDS clock frequency, the value of the Frequency Constant stored in EE memory will have to be changed and the modified .ASM file reassembled.

For convenience, a startup frequency stored in EE is loaded into the DDS at boot up, before any valid serial commands are received. The default is 137000Hz and can only be changed by modifying the .ASM file and reassembling.

The applicable lines of code in the .ASM file are :

```
FREQCONST   de 0x00,0x00,0x57,0xF5,0xFF,0x86    ;Fstep/Fclock * 2^64    (125MHz clock, 0.01Hz steps  = 1475739526)
STARTFREQ   de 0x00,0x47,0xD3,0xD4              ;DDS registers for 137000Hz
```

Code Functionality

Input data in Hz, with an optional two digits following a decimal point are received from the serial interface, aligned and converted to a binary representation of the desired output frequency. The frequency is maintained inside the firmware in units of 0.01Hz with the count kept in a 32 bit register. This means the maximum frequency possible is 2^{32} * 0.01 = 42.949MHz; just about the maximum allowable from an AD9850 device. Alignment of the received data is done by identifying the position of the [cr] and the decimal point.

To convert this wanted frequency to values to send to the DDS chip, the frequency is multiplied by a constant derived from the wanted step (0.01Hz in this case) and the DDS clock. Since integer arithmetic is used throughout, values need to be scaled appropriately to get the required accuracy.

The constant used is calculated from : Step size / DDS Clock * 2^{64}

So for the default values of 0.01Hz and 125MHz, the constant is :

0.01 / 125MHz * 2^{64} = 1475739526 or in hexadecimal 0x57F5FF86.

This is stored as a 48 bit number in EE, and can be seen at the start of the .ASM file against the label '*FREQCONST*' A modified 32 x 48 bit multiplication routine is used, with the factor of 2^{64} then removed by taking only the highest 6 bytes for a 48 bit result. Since the AD9850 has only a 32 bit frequency setting

register, the lowest two bytes of the result are further discarded to give a 32 bit value to send to the DDS chip.

Accuracy is dependent on two factors. Most critical is the actual setting resolution of the DDS. With a 125MHz clock the resolution of the 32 bit AD9850 is 125MHz / 2^{32} = 0.029Hz (which is greater than the programming resolution / step of 0.01Hz). The use of a constant scaled to 64 bit accuracy leads to a cumulative error such that at the highest programmed frequency, the error could amount to around 0.01Hz. At lower frequency setting values, the cumulative error is below the 0.01Hz setting resolution.

With a lower DDS clock, frequency errors are proportionately reduced.

Frequency Setting Commands

Use a terminal emulator programme, or some other means of generating ASCII text on an RS232 interface. Terminal parameters are 9600 baud, 8 bit data, 2 stop bits, no parity, no handshaking. 9600,N,8,2

The DDS then just needs commands starting with the keyword **FRQ** followed by the frequency in Hz and terminated by a carriage return [cr] The keyword can by in upper or lower case. Spaces are permitted (*) between the keyword and following numerical data. Up to two decimal places of frequency are used, any further decimal place values are ignored. The use of a decimal point is optional, frequencies can be entered as integer values of Hz. Spaces are NOT PERMITTED after the last digit ands before the [cr]

No local echo is used, so if you want to see characters as they are typed, turn “Local Echo On” in your terminal programme.
All of the following frequency setting commands are valid :

FRQ137456[cr]
Frq 10700000.0[cr]
FRQ 475500.01[cr]

If the command is accepted, the entered string is echoed back in it its entirety. If the command is not accepted, only a single [cr] will be echoed back.

The DDS will immediately be updated with the desired frequency

And that’s it – Simples !

Possible Future Enhancements

Adding a STOre option to save the current frequency to EE for startup next time the unit is turned on

Allowing frequency to be entered in MHz and kHz using M and K in the data string

Transferring to a PIC with calibrated RC internal clock to remove the need for Xtal / resonator

(*) The only reason spaces are permitted between the FRQ header and any following numbers is that a space is interpreted as a zero within the PIC firmware. Any other character appearing here will more than likely be interpreted as a decimal digit with some other value than zero. Spaces appearing as zeros also show why they must not appear between the numerical data and the carriage return

Use with the ZL1BPU / ZL2AFP Weak Signal QSO Software

```
WSQ2 setup for serial comms and PTT

1
9600
N
8
2
FRQ475200.0
FRQ475202.0
FRQ475203.9
FRQ475205.9
FRQ475207.8
FRQ475209.8
FRQ475211.7
FRQ475213.7
FRQ475215.6
FRQ475217.6
FRQ475219.5
FRQ475221.5
FRQ475223.4
FRQ475225.4
FRQ475227.3
FRQ475229.3
FRQ475231.3
FRQ475233.2
FRQ475235.2
FRQ475237.1
FRQ475239.1
FRQ475241.0
FRQ475243.0
FRQ475244.9
FRQ475246.9
FRQ475248.8
FRQ475250.8
FRQ475252.7
FRQ475254.7
FRQ475256.6
FRQ475258.6
FRQ475260.5
FRQ475262.5
FRQ475264.5
FRQ475266.4
FRQ475268.4
FRQ475270.3




Description of parameters above -
-comport
-baudrate
-parity
-bits
-stopbits
-remaining lines are synthesizer commands
```

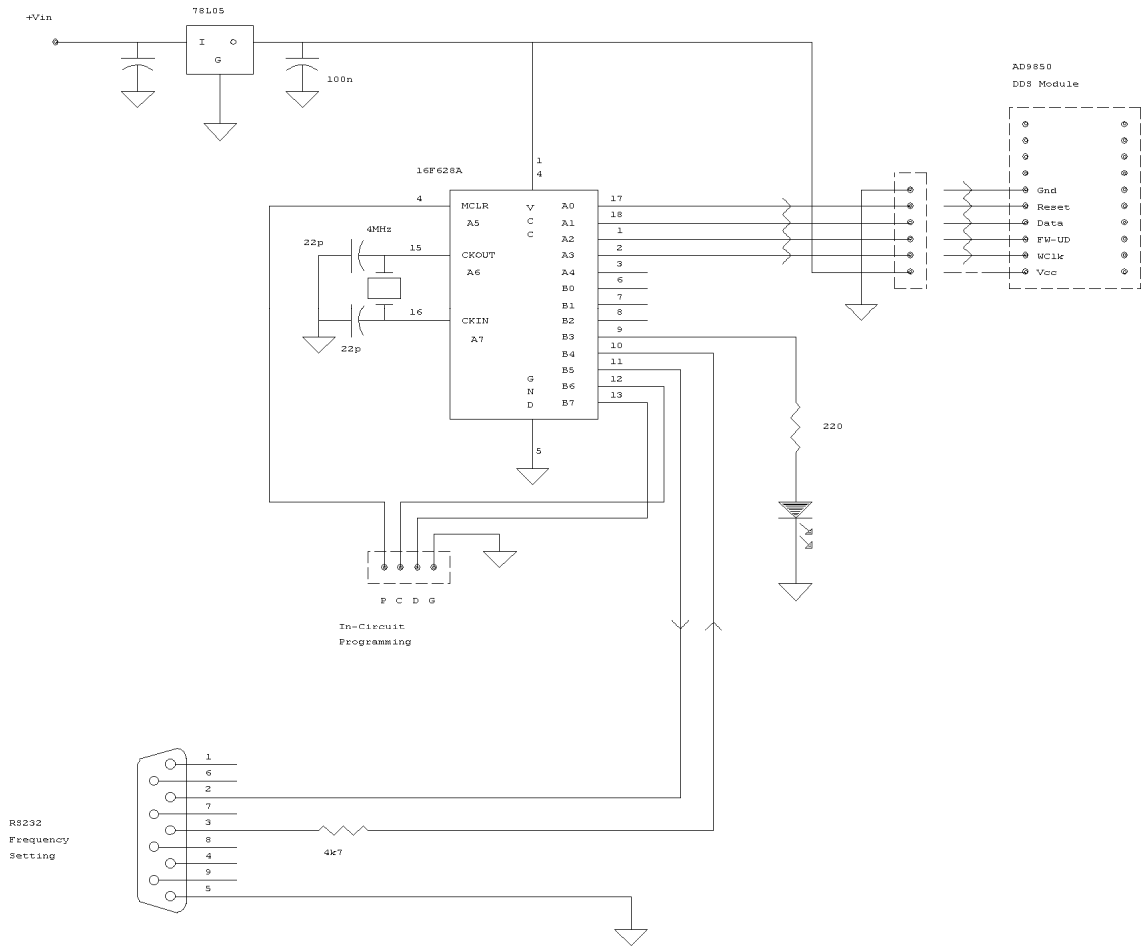
WSQ has the facility to drive a synthesiser via a COM port. Read the WSQ documentation thoroughly , particularly the section on Synthesizer Installation in the *WSQhelp.htm* file. The WSQ setup file. *Setup.txt* needs to be written, with the 33 tone frequencies specified in the format needed for this synthesiser control. A typical example is shown here

Murray's spreadsheet *FEI Calculator.xls* can be used, but do not use the hex codes highlighted in yellow. Instead use the frequency values listed in column B. Copy and paste these into Set.txt then add the keyword 'FRQ' at the start of every line. The spreadsheet could be easily modified to do this for you. Ideally the frequencies should also be formatted

The WSQ software has been tested with this firmware using the setup.txt file listed here.

References

[1] PIC Code and design utilities http://www.g4jnt.com/AD9850_Serial.zip


+Vin
78L05
I
O
G
100n
AD9850
DDS Module
Gnd
Reset
Data
FW-UD
WClk
Vcc
16F628A
MCLR
A5
CKOUT
A6
CKIN
A7
VCC
GND
A0
A1
A2
A3
A4
B0
B1
B2
B3
B4
B5
B6
B7
4MHz
22p
22p
220
P C D G
In-Circuit
Programming
RS232
Frequency
Setting
4k7