Serial Control for the VertiCom MTS1500 Synthesizer

Andy Talbot G4JNT July 2009

A little while ago I bought a VertiCom MTS 1500-151-01 12GHz Synthesizer module from Dave Robinson, G4FRE/WW2R. Dave supplied a write up and had already worked out the programming codes needed to be sent as a serial word to the module to set it onto any frequency within its operating range. All details are available from [1] and will not be repeated here.

Dave already supplied PIC code for blowing into a 12F629 to set the unit onto either of two frequencies [2] – but to change these necessitated getting into the PIC assembly listing and hard coding several register values. I wanted to be able to update the values on-the-fly with a simple RS232 connection using ASCII commands from *Hyperterminal*.

Having just written some PIC code to control the LMX23x6 synthesizer in the Bridgewave Synthesizer module [3] this was a good starting point. Fortunately Dave's PIC / module interface used the same PIC device, and almost the same pin connections (just two swapped) as mine, so I decided to make the VertiCom interface fully compatible with his existing PIC module, so the only changes needed would be addition of the RS232 interface (a resistor and three wires to a 9 way D connector) and, of course, reprogramming with the new firmware.

The RS232 interface is arranged to share the same pins on the 12F629 as those used for clock and data from the device programmer, and a convenient way to manage both in-circuit programming and RS232 control is to install a four way header as shown in Figure 1 (see *Verticom.gif* for a higher quality version) A resistor is needed in the RS232 interface to limit the negaive current (unfortunately this resistor can't be mounted on the PCB as it prevents the PIC programmer form working). The same serial interface is used for my Bridgewave controller [3], LMX1500 Synth module [4] and [5] and programmable Beacon Keyer [6].

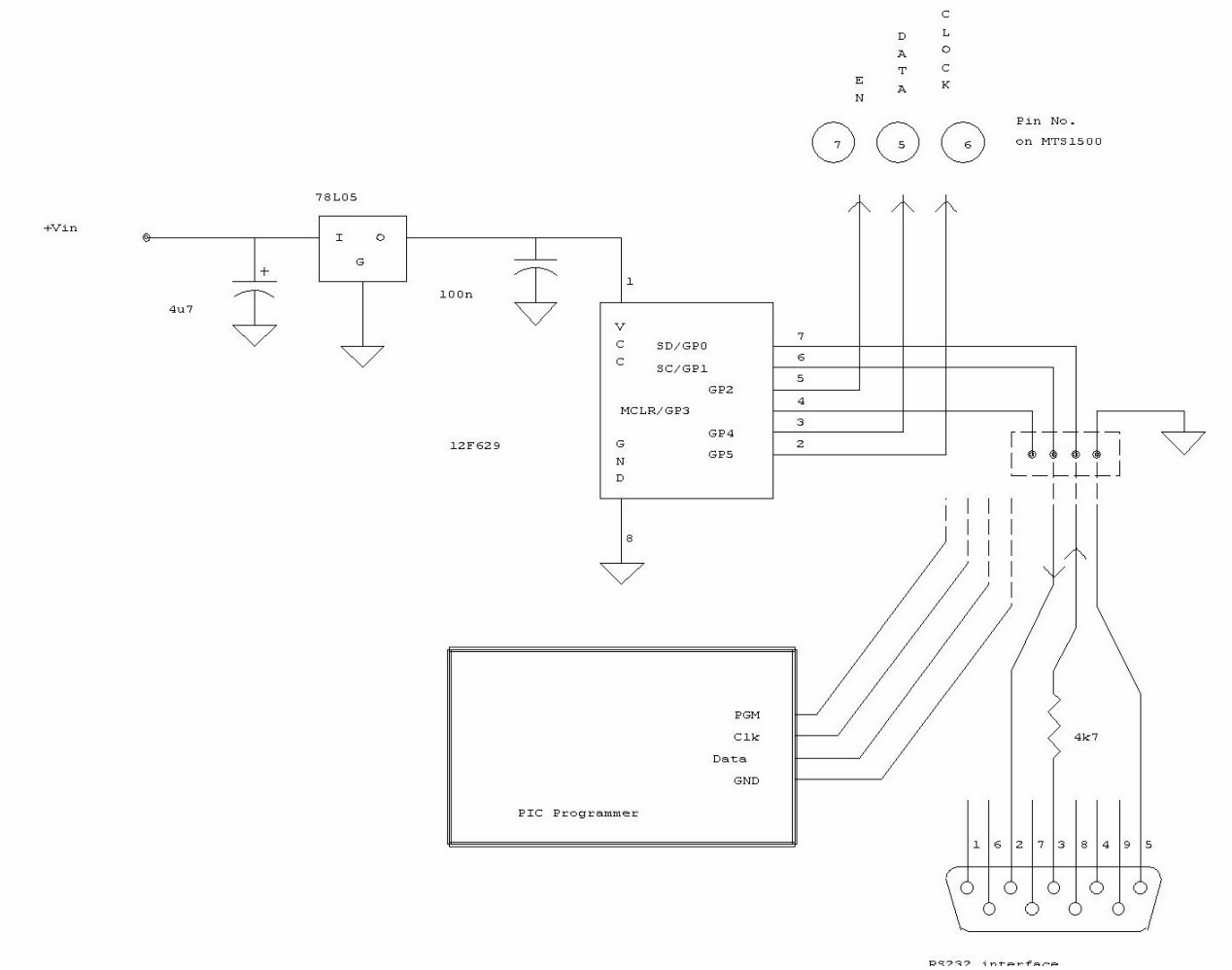

Figure 1 Interfacing the PIC

Updating and interfacing the PIC controller

1) Download MTS15CTL.ZIP from [7] , extract the PIC firmware MTS15CTL.HEX and (re)programme the 12F629. The .ASM source code is included for reference.

2) Make up an interface lead with a 9 Way D Female connector and 4k7 resistor as shown in Figure 1, and connect to your computer's COM port.

3) Run ***Hyperterminal***, with the parameters set to 1200 Baud, 8 Bit data, No parity, 2 stop bits and no handshaking. 1200 N81

The Synthesizer is controlled with ASCII commands. All commands must be terminated with a Carriage Return , shown here as [cr]

In the exmples shown below, data you type is shown underlined such as **N12345[cr]** , and responses from the controller shown in italic ***N – 12345***

Do not put any extraneous spaces into commands – in the examples below a space is shown before the [cr] purely for clarity. If a command has the wrong syntax, no response will be received from the controller. Upper or lower case letters are accepted.

Connect the interface and turn on the synthesizer. A display similar to this should appear and shows the values stored in non-volatile memory with a brief description of the command protocol to change them.

```
Verticom Synth Control
G4JNT/WW2R

Commands:
Rxxxx
Pxxx
Sxx
Cxxx
Gxx
[U]pdate
[W]rite EE

R 003F
P 061
S 10
C 19C
G 05
```

Each of the five registers R, P, S, C and G can be set individually. The data is entered as hexadecimal ASCII values with either two, three or four hex digits following the register name. A modified version of Dave's spreadsheet for generating the values, *MTS1500_JNT.xls* included within this archive gives the hex data needed.

The values are entered one at a time but are not immediately sent to the MTS1500 module. For example to set P = (hex) 45, S = 12 and R = 0123 Enter :

P045[cr]
The controller will respond with ***P-045***
Then type **S12[cr]** Generating the response ***S-12***
Then **R0123[cr]** Generating the response ***R-0123***

When all values have been successfully entered, press **U[cr]** to Update the module and start generating the new frequency. The controller will respond with :

```
R 0123F
P 045
S 12
C 19C
G 05
Updated
```

To save these to non-volatile memory so the controller will boot with the new values next time it is turned on, enter **W[cr]** to give the response

```
R 0123
P 045
S 12
C 19C
G 05
Written
```

To avoid confusion…

Note that the hex values that have to be entered are not the same as in Dave's original spreadsheet *MTS1500.xls* due to bit and byte alignment issues in the way I send data across the interface.

To make comparisons and help with checking, a debug facility has been built-in to the PIC code to show the actual string of data sent to the module (I only needed this originally for getting the code operational, but decided to leave it in for convenience).

Type **D[cr]** to toggle the Debug mode on or off.
Each time it is invoked the controller will respond with ***DEBUG 0*** or ***DEBUG 1*** for on or off respectively.

If Debug mode is ON, then whenever a **U** or **W** command is issued the contents of the two control words CW1 and CW2 are shown in binary, and can be checked against those in MTS1500.xls.

```
R 003F
P 061
S 10
C 19C
G 05
CW1 10000000011111110000000000000000000000000
CW2 00001100001001000000000011001110000000000101
Updated
```

MTS1500_JNT.xls shows the hex values.

References:
[1] http://www.g4fre.com/mts1500.pdf
[2] http://www.g4fre.com/mts1500.zip
[3] http://www.g4jnt.com/BridgeWave_Synth.pdf
[4] http://www.g4jnt.com/synthblb.asm
[5] http://www.g4jnt.com/synthblb.gif
[6] http://www.g4jnt.com/BCNKEYER.zip
[7] http://www.g4jnt.com/MTS15CTL.ZIP