Controlling the Bridgewave Communications type Microwave Synthesizers

Andy Talbot G4JNT July 2009

At the RAL Roundtable, Paul M0EYT handed my a flat grey module *Photo 1*, said it was a 12GHz synthesizer, and "could I do anything with it"

On opening up, the module was found to contain a synthesizer based around an LMX2326 chip and a Zcom CLV1525E VCO, followed by a conventional times 8 multiplier chain and splitter for two 12GHz outputs. Several internal regulator chips are 5V output devices, and the DC power input was stated by Paul as being +8V.

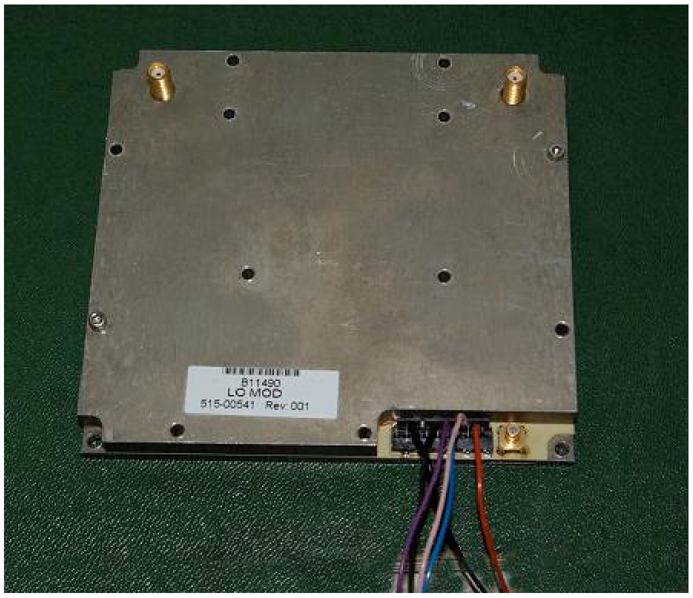

Photo 1 - The Bridgewave Communications 12GHz Synthesizer Module

There was no inbuilt micro or any logic for control, and the three Microwire programming pins from the synthesizer chip were brought to the outside world on a 10 way connector. Connections for this were traced out and are shown in *Figure 1* :

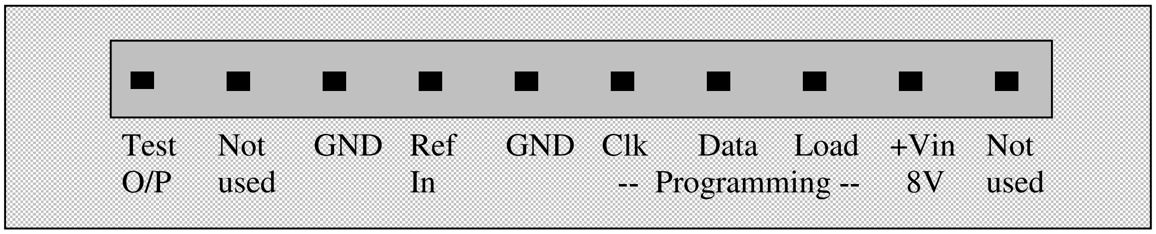

Figure 1 *Bridgewave Communications Synthesizer interface*

There is no internal reference source for the PLL and this has to be supplied externally Pin 4 for supplying the reference input has a track leading to pads for a PCB mounted coax connector adjacent to the 10 way connector so an SMC socket was installed here for convenience as shown in Photo 1. Applying +8V power (350mA drawn) resulted in an unlocked 12GHz signal appearing on the two output SMA connectors. So all appeared to be working.

Programming.

I already had a PIC utility for controlling LMX1501 devices from a serial RS232 interface. See http://www.g4jnt.com/synthblb.asm This was built around the tiny 8 pin PIC 12F629 . The firmware was modified to send the commands in the format needed by the LMX2326 which wants 21 bit words consisting of 19 bits of Data for each of the **R**eference and **N** divider values plus the **F**unction or control register, and two address bits to define the register being programmed.

The connections for the interface are shown in *Figure 2*. The four pin connector serves a dual purpose. It serves as the in-circuit programmer for the PIC and also as a direct interface to a serial or RS232 port (with an additional resistor). This allows a simple external interface for routine programming of frequency and setup information, as well as being able to update the PIC code as the need arises. The serial control allows values to be programmed into non volatile memory so it will boot up with the correct frequency at turn-on.

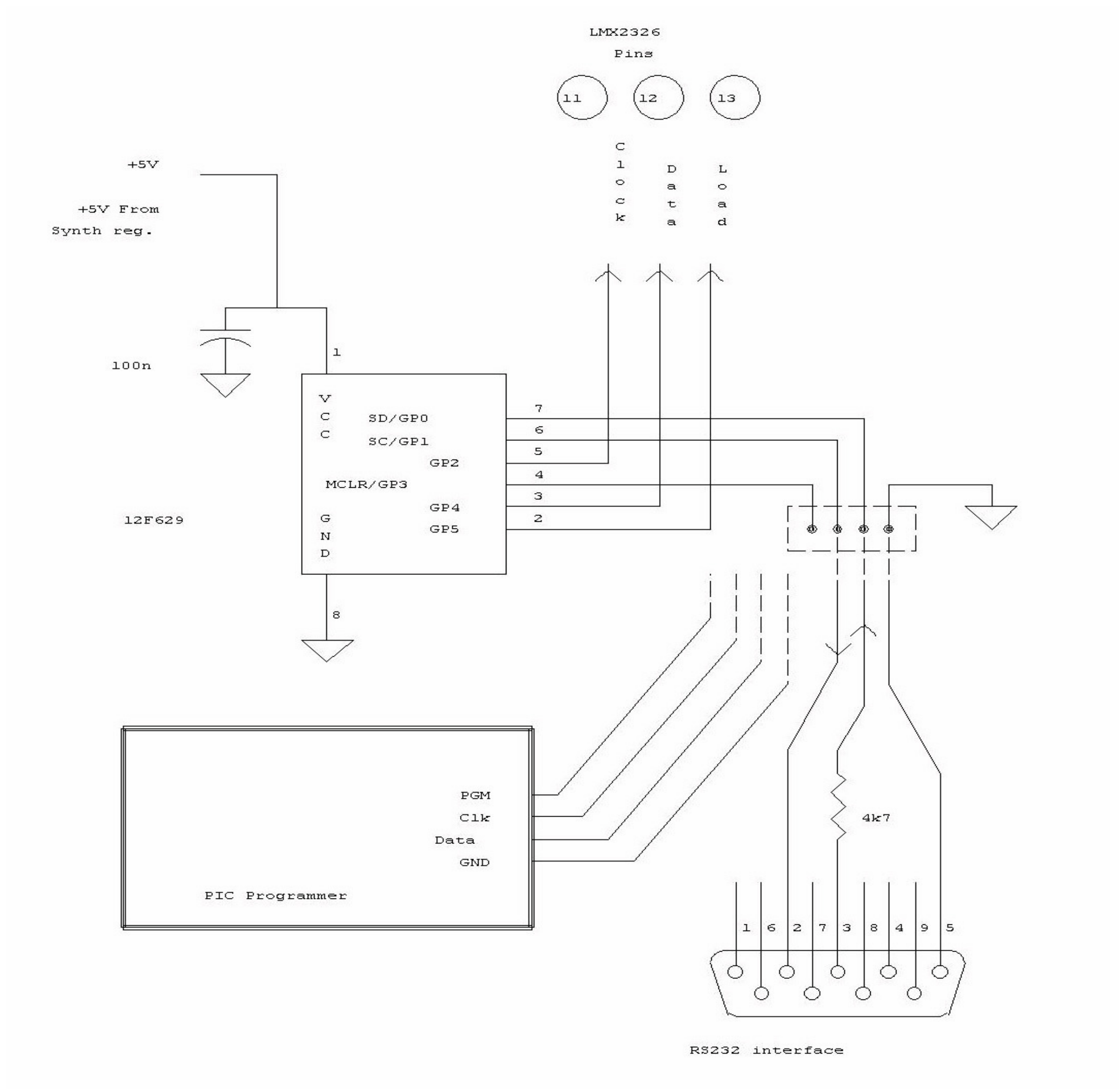

Figure 2 *PIC Controller Interface for LMX Synthesizer chips*

Synth Control

Close examination of the LMX23x6 data sheet available from http://www.national.com/mpf/LM/LMX2326.html gave the minimum requirements for the Function register – the only bit that actually mattered to the PLL operation was the VCO polarity bit which has to be set to 1 for positive tuning direction. All other bits can be zero. There is a convenient test-pin on the LMX2326 Pin 14, *Fo/LD*, which outputs one of several internal signals from the PLL depending on Function register programming. This is brought out to Pin 1 of the external interface and for initial testing was programmed to give the divided-down N divider output. When looked-at on a scope the waveform should consist of reference spikes at the comparison frequency and will show whether there is any PLL instability, or if it is even locked!

I used a 10MHz reference input and initially guessed the PLL might be set up for a comparison frequency in the region of 200kHz. So programmed in values of R = 0x32 (decimal 50 for the hexadecimal-challenged ☺) and N = 0x1DC9 - (go on work it out) for the middle of the VCO's specified range. The spectrum analyser showed a nice clean signal at 12GHz. Then a brainwave struck; a comparison frequency of 125kHz results in 1MHz steps at the output, and N = output Frequency in MHz . (R = 0x50 and N = 0x2EF8 gives 12024MHz = 24048 / 2)

Although the VCO is only specified for 1500 – 1550MHz it locked happily down to at least 1450MHz which is good! This gives 11600MHz output, and when used as an LO for 24GHz, is 23200MHz, meaning that for a 24048 receiver it will allow any IF below 800MHz. It may even go lower, but I suspect 11376MHz for a 23cm IF is a bit too much.

Shoehorning it in

As there is plenty of space on the Synthesizer's PCB, it was decided to mount the PIC inside the case and apply the serial programming to pins on the existing 10 way connector. There are not enough spare pins for a complete interface as three are needed if PIC device programming is to be available as well as the RS232 interface. Fortunately, the LMX programming pins have zero-ohm links in their tracks to the 10-way connector so these were removed and the three external programming pins now became my interface.

The 8 pin PIC was glued onto the PCB adjacent to the nearest ground track I could find that wouldn't have part of the upper casing lying on top of it when assembled. Refer to *Photo 2*. A dab of superglue was used to firmly fix the chip in place which was installed the right way up with the pins lifted to clear. I've had unpleasent experiences with wrong pin connections and blown chips using upside-down dead-bug construction - perhaps this ought to be called live bug construction. The location shown is not the neatest place for it, but was the only place I could find where a direct connection to ground was available and resulted in six relatively long signal links going to the pads of the now-removed zero ohm links. Note the 100nF decoupling cap on the Vdd pin, also connected directly to ground. All totally OTT, and a much nicer job could have been made by installing the chip straddling the three signal lines, resulting in six short signal links and two longer PSU / ground ones. If the decoupling cap was connected directly across the PIC pins 1 and 8 it would still have been more than adequate.

The PIC Code as well as copies of the figures can be found at http://g4jnt.com/Bridgewave.zip

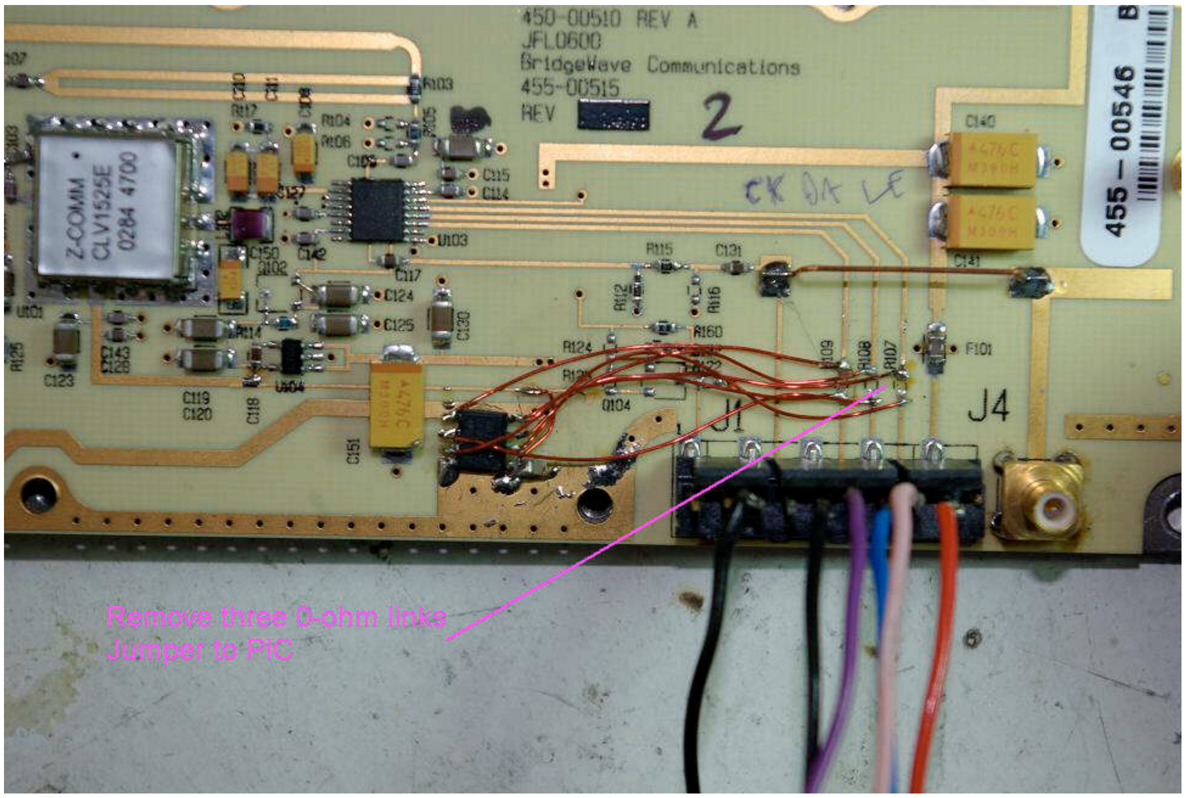

Photo 2   The PIC interface mounted onto the Synthesizer PCB (in a non-optimum position)

Serial Interface Programming Instructions

Commands are sent to the Synth Controller with ASCII commands.   All commands must be terminated with a Carriage Return , shown as   [cr]

In the exmples below, data you type is shown underlined such as **N12345[cr]** and responses from the controller shown in italic ***N – 12345*** .   Upper or lower case letters are accepted and typed characters are not echoed back

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.

Make up an interface lead with a 9 Way D Female connector and 4k7 resistor as shown in Figure 2, and connect to your computer's COM port.   Run ***Hyperterminal***, or ant similar serial driver programme with the parameters set to 1200 Baud, 8 Bit data, No parity, 2 stop bits and no handshaking : *1200 N81*

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

```
LMX23x6 Control    G4JNT
Commands:
Rxxxx
Nxxxxx
Fxxxxx
[U]pdate
[W]rite EE

R 0050
N 02EB0
F 00030
```

The three registers are updated by entering the letter R, N or F followed by four or five hexadecimal characters then [cr]. T o set a new value of N, work out the value needed for the programmable divider, convert to a five digit hexadecimal number and type, for example **N02EB1 [cr]** If the data has been accepted the controller will respond with ***N-02EB1***

Do the same for R if desired, but note that this only takes a four digit value, for example **R0050 [cr]** will respond with ***R-0050***

Unless you really need to, and have studied the data sheet, leave the F value alone.

The values are not immediately written to the synthesizer, so all R, N and F can be set and all sent together. When all three have been succesfully entered, press **U [cr]** to send these values to the LMX2326 chip.
If everything is OK, the frequency will jump to its new correct value and the controller will respond with :

```
R 0050
N 02EB1
F 00030
Updated
```

Pressing **W [cr]** instead causes the values to be written to non-volatile memory as well as updating the device, and will be loaded next time the unit is turned on.
The response in this case is :

```
R 0050
N 02EB1
F 00030
Written
```

Note that as it stands, there is no scope for setting the LD Precision bit – the MSB of the R register - via the serial interface. If you need to set this it will have to be changed in the PIC firmware.