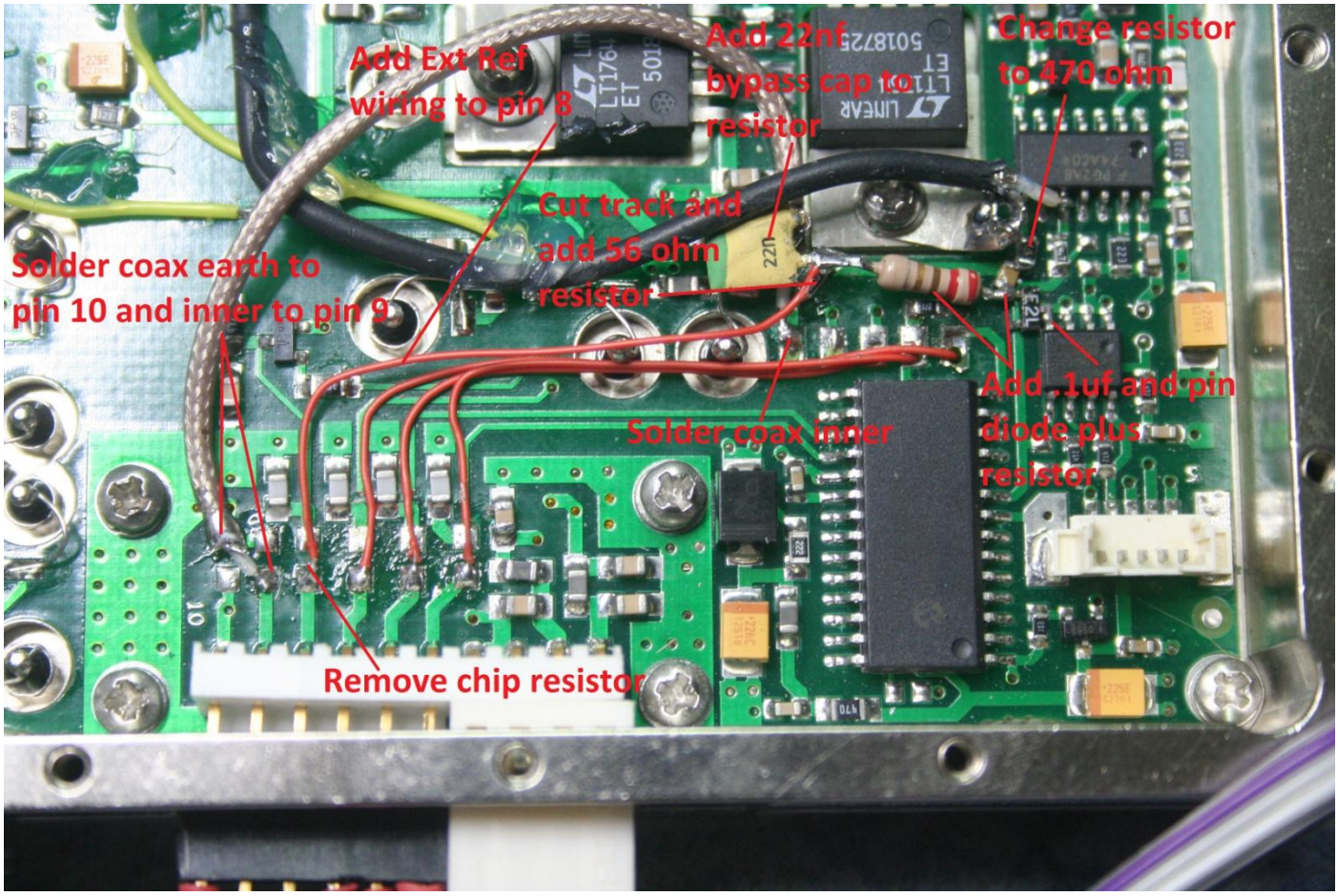
Reference Internal – External switch modification

To allow the selection of internal or external reference selection some modifications have to be made. The internal oscillator runs several functions with-in the unit. One signal is used for the 26 volt supply to drive the PLL, another derives the clock for the internal micro, which is no longer used, and thirdly it provides the negative bias for the GASFet amplifiers. I decided not to touch this oscillator as it could inadvertently cause loss of bias for the fet amplifiers and their possible destruction.

Pin 8 of the 74AC04 buffer feeds the PLL reference via a 47 ohm resistor. This resistor was removed and replaced by a 470 ohm resistor. A .1uf capacitor and a pin diode were added to the PLL side of the 470 ohm resistor.

In addition, the track was cut near where the signal goes into the PLL reference link and a 56 ohm resistor was added. Pin 9 of the interface connector has a piece of small coax (RG179) added to connect this pin to the PLL reference input. There is a 220 ohm resistor added to the junction of the .1uf and pin diode to provide bias for the pin diode so it can isolate the internal reference signal to the PLL. The DC bias control line is connected to pin 8 of the interface connector and a smd resistor is removed from the pin 8 position as per the attached photo.



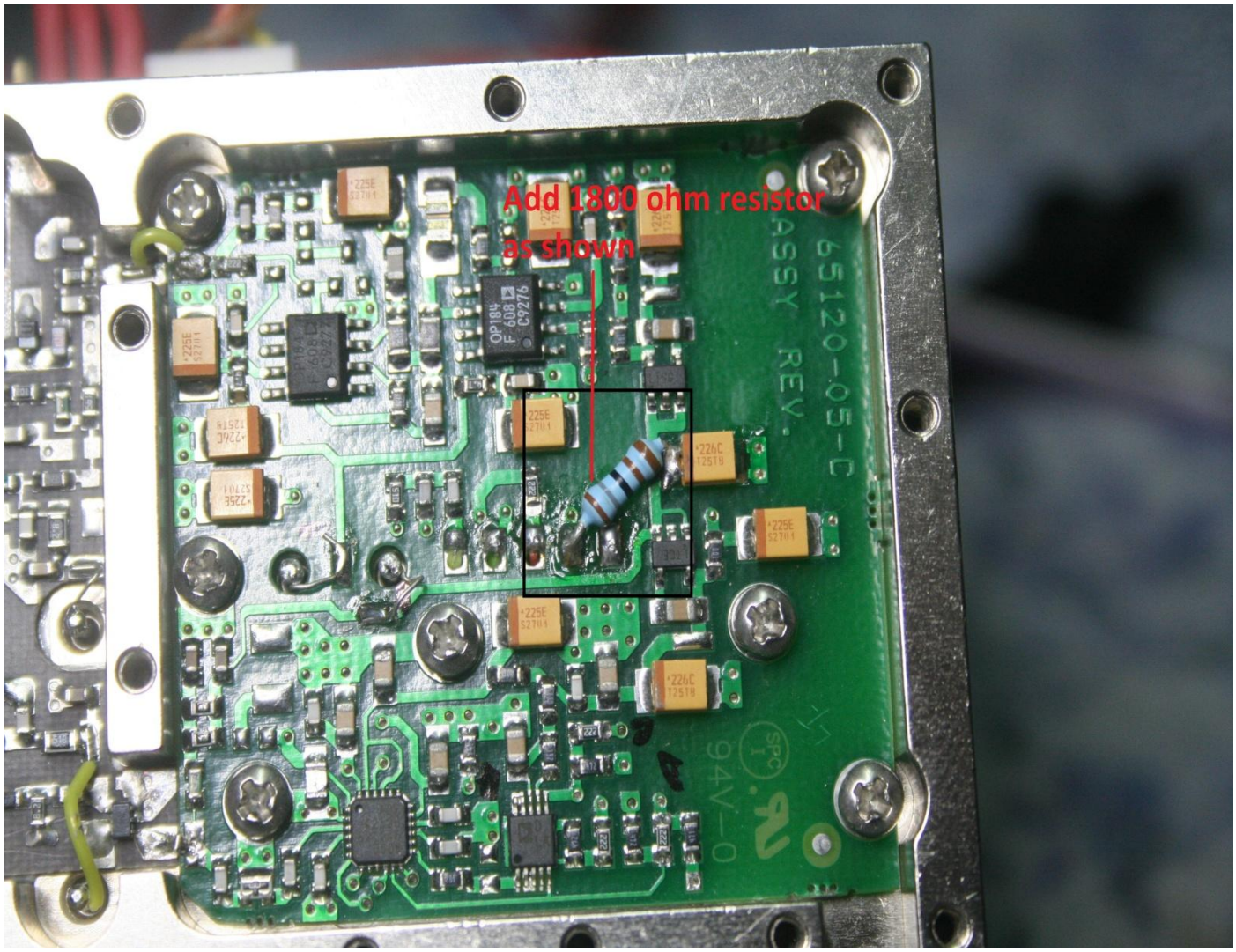
When +5 volts is applied to pin 8 of the interface connector the pin diode is biased on and effectively shorts the internal reference signal going to the PLL (leaving all other oscillator functions running) and provides a termination to the external reference via the 56 ohm resistor.

Connecting an external reference will then operate the PLL. Any signal from -10dBm to +10dBm will function correctly. An external reference CANNOT be connected without power applied to pin 8 of the connector otherwise beating of the two signals will occur.

Out of Lock detection

The initial out of lock detect on the unit will no longer function with the modifications listed above. This is due to the initial way the system was designed, the O/L function from the PLL was multiplexed with the serial data line to the microprocessor.

The external microcontroller has been programmed to either load data on a write cycle or read the O/L function when not writing data to the PLL. The added resistor ensures the correct voltage levels are available for the external controller to detect the O/L signal.



18F2520 Controller PCB

A universal controller board was designed based on a Microchip 18F2520 microprocessor.

The board communicates with the Elcom synthesiser via the SPI bus interface to either load the frequency data or to read the O/L status from the synthesiser.

Port B.0 provides the out of lock signal. The pin is high on out of lock and low when locked.

This pin is 5 volt TTL from the microcontroller so caution should be exercised if connecting to external circuits.

The ICSP header is used to program the microprocessor.

The 12 volt supply header, SPI bus connector and PortB.0 are the only connections used. With modifications to the software the other pins can be used to select multiple frequencies from the unit.

Interface Connector pin out

Pin 1	+8 volts
Pin 2	+12 volts
Pin 3	Earth
Pin 4	Earth
Pin 5	Serial data (SDO) connection
Pin 6	Serial clock (SCK) connection
Pin 7	Latch enable (EN)
Pin 8	Internal/External Reference select
Pin 9	External 10Mhz reference input
Pin 10	Earth

Interconnection cable from controller to Elcom

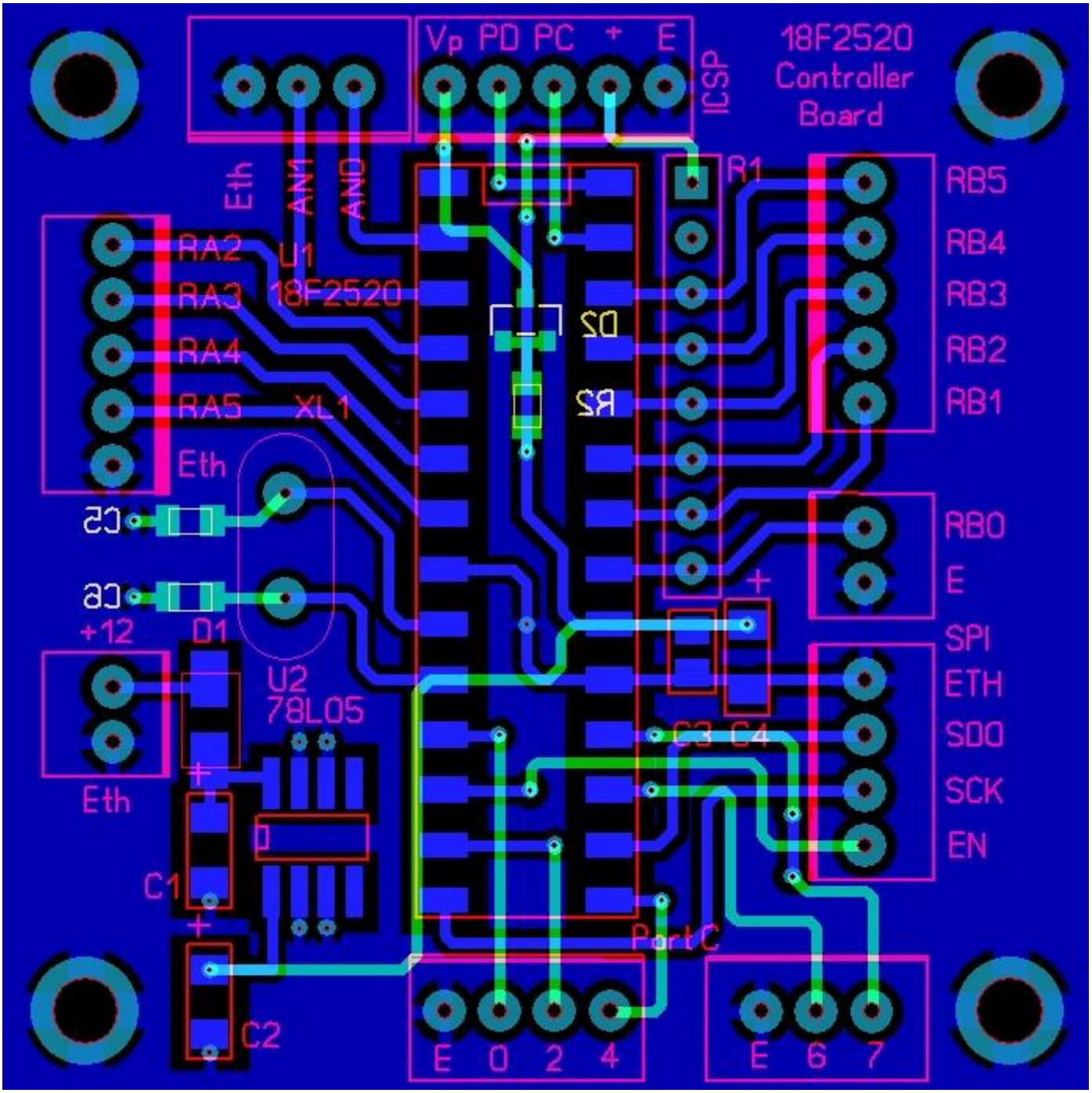
Controller SPI connector Elcom Connector

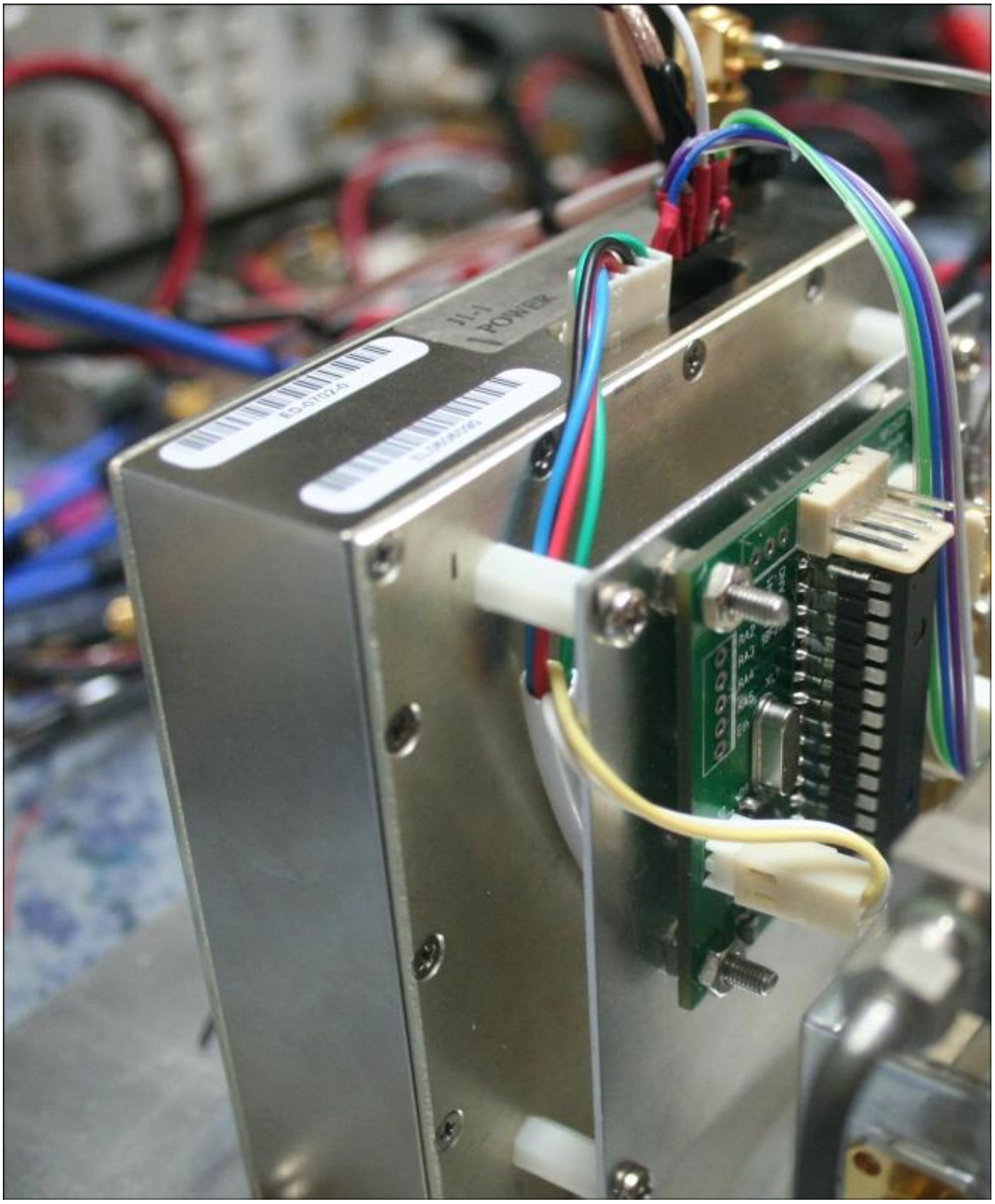
ETH ----------Pin 10

SDO ----------Pin 5

SCK ----------Pin 6

EN ----------Pin 7


18F2520
Controller
Board
ICSP
Vp PD PC + E
Eth
AN1
AN0
U1
18F2520
RA2
RA3
RA4
RA5
XL1
Eth
C5
C6
+12
D1
U2
78L05
C1
C2
Eth
R1
D2
R2
RB5
RB4
RB3
RB2
RB1
RB0
E
SPI
ETH
SDO
SCK
EN
C3
C4
PortC
E 0 2 4
E 6 7


J1-1
POWER

The software is written in MikroBasic from MikroE.

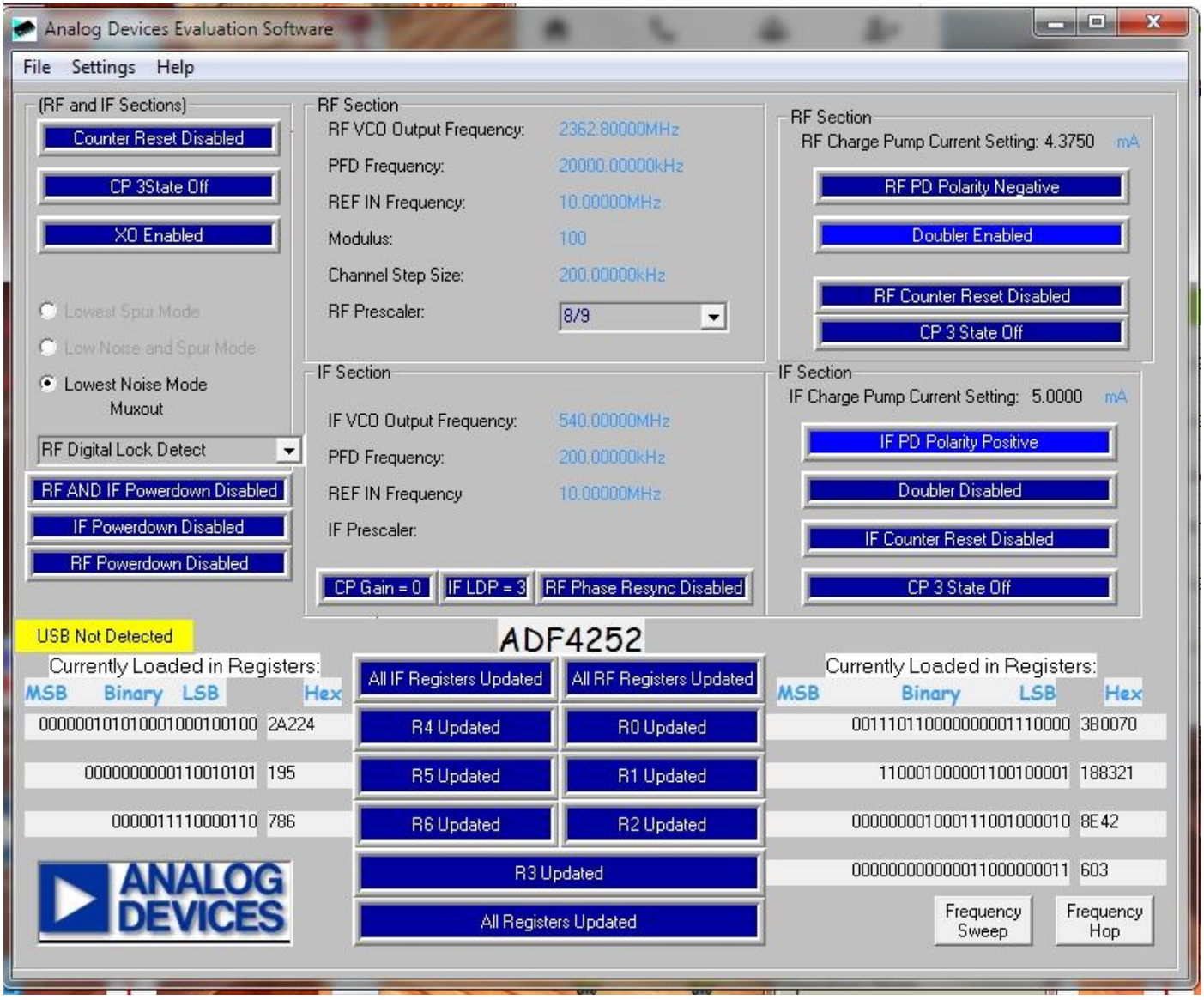
The free version of the compiler is available from here : http://www.mikroe.com/mikrobasic/pic/

The calculation of the data for frequency selection is easily done from the Analogue Devices application available from here :

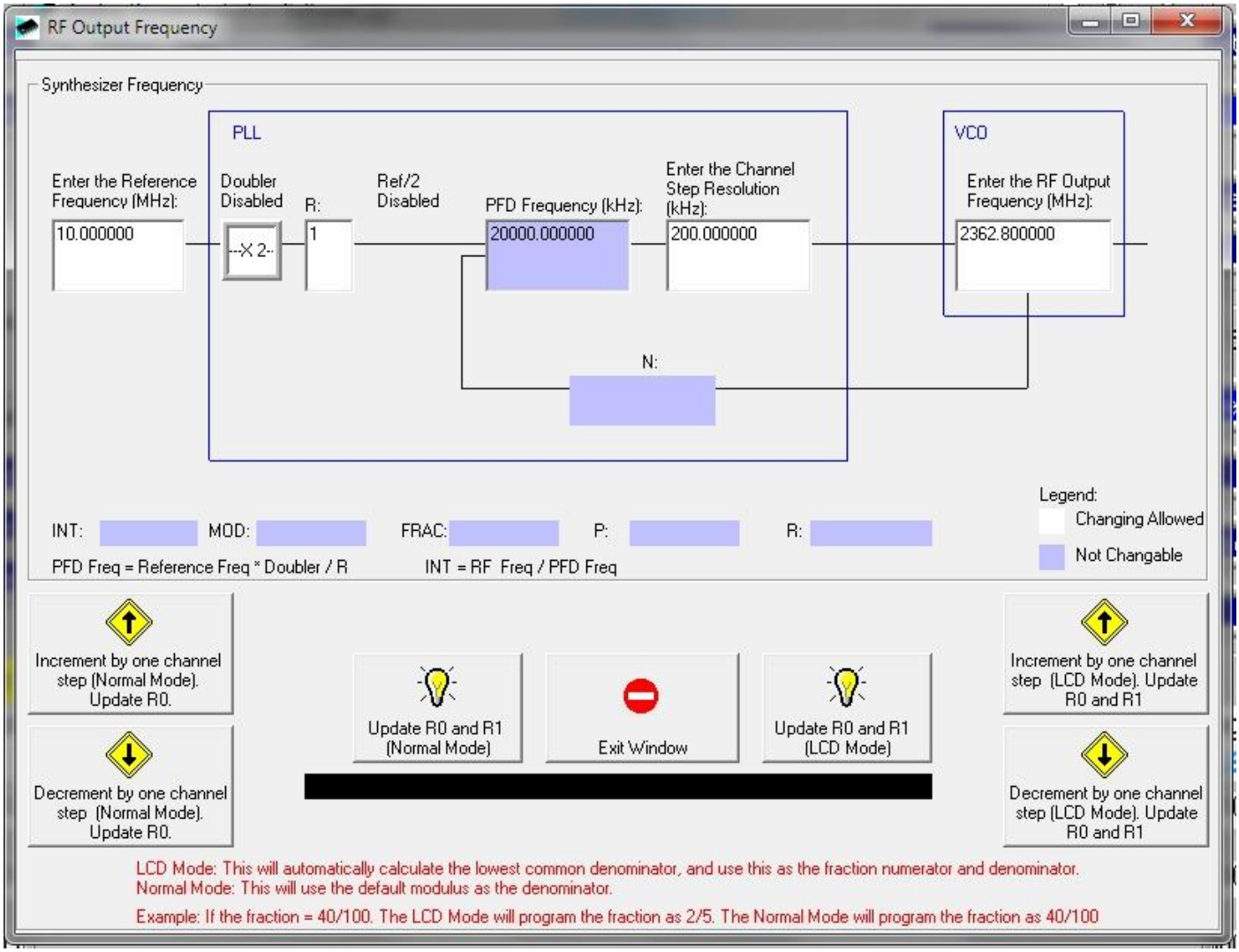
http://www.analog.com/en/rfif-components/pll-synthesizersvcos/adf4252/products/EVAL-ADF4252/eb.html

Download and install the application. When it loads you will have to select the ADF4252 synthesiser and ignore the warning about no USB found. This software is written for their development board

Screen shot from software showing hex data for the values used in the program.



Screen shot showing PLL setup parameters



```
program Elcom_syntherizer

'Completed 14/08/2013
'This programs the Elcom synthesiser at 200Khz channel increments
'Monitors the LOCK status and alarms if out of lock
'Detector polarity is negative on the PLL
'One channel implemented in this software version
‘Implemented on 18F2520 microcontroller


Sub Procedure Channel_1()
     TRISC = $00                    Set PortC to outputs
     SPI1_Init()

     Delay_ms(200)

     SetBit (PortC, 1)
        Delay_ms(1000)
     ClearBit (PortC, 1)
        Delay_ms(10)

        SPI1_Write ($3B)       '3B          'program the RF "N" register
        SPI1_Write ($00)       '00          'Set output freq to 11814Mhz
        SPI1_Write ($70)       '70          ‘VCO frequency of 2362.8Mhz

    SetBit (PortC, 1)
        Delay_ms(10)
    ClearBit (PortC, 1)
        Delay_ms(10)

        SPI1_Write ($18)       '18          'program the RF "R" register
        SPI1_Write ($83)       '83
        SPI1_Write ($21)       '21

    SetBit (PortC, 1)
        Delay_ms(10)
    ClearBit (PortC, 1)
        Delay_ms(10)

        SPI1_Write ($8E)       '8E          'program the RF "control" register
        SPI1_Write ($42)       '42


    SetBit (PortC, 1)
        Delay_ms(10)
    ClearBit (PortC, 1)
```

```
        Delay_ms(10)

        SPI1_Write ($06)      '06        'program the Master Register register
        SPI1_Write ($03)      '03
        Delay_ms(10)

    SetBit (PortC, 1)

    TRISC = $FF                    ‘return PortC to inputs

    Delay_ms(2000)

end sub

main:

  TRISC = $FF                      'Set Port C to inputs
  TRISB = $00                     ‘Set PortB to outputs
  Delay_ms(200)
                                 'At startup select channel not used in this version
 ' if PortD.4 = 1 then
  Channel_1
 ' end if

 ' if PortD.5 = 1 then
 ' Channel_2
 ' end if

 ' if PortD.6 = 1 then
 ' Channel_3
 ' end if

 ' if PortD.7 = 1 then
 ' Channel_4
 ' end if

run:                               ' check PLL lock status

  while true
    if Button(PORTC, 5, 10, 0) then
      PortB.0 = 1
    end if

    if Button(PORTC, 5, 10, 1) then          'PortB.0 = 0 when phase locked
      PORTB.0 = 0
    end if
wend
end.
```