**1** **Purpose**

**2** **Related Documentation 3**

**3** **Hardware Schematic 3**

**4** **User Interface (Hyper Terminal) 4**

*4.1* *Opening HyperTerminal 4*

*4.2* *Setting Session parameters HyperTerminal 5*

*4.3* *Help command ‘?’- HyperTerminal 5*

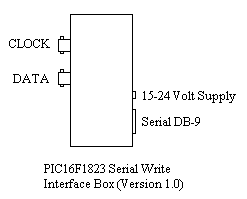
*4.4* *Send Data: ML = Set Data 7*

*4.5* *SVXX = Set Vout (default = 3.435 V) 8*

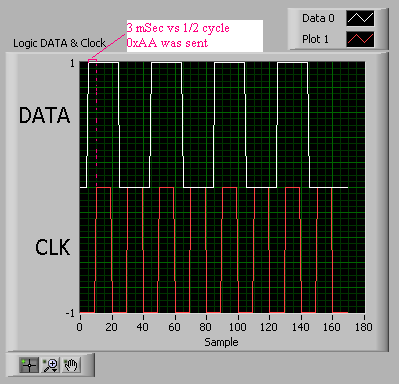
*4.6* *SFX = Set Frequency (default = 10 Hz) 10*

*4.7* *What if I send a Command not already covered? 11*

**1 Purpose**



The purpose of this device is to create an instrument capable of driving a shift register by way of a serial interface. It must also not require the use of Labview or require a user interface to be written. The clock frequency and data/clock amplitudes must be programmable. All function supported by the LV NI-DAQ version must be support but none of the environment elements of that structure must be required (No LV, NO NI DRIVERS, No NI HW). This HW will appear to operate slightly different then the LV NI-DAQ version. The data line will only be transitioned 3 mSec prior to the clock edge. The NI-DAQ version transitioned much sooner. This difference has no impact on the performance.



It is also the intent of this design to allow for an interrupt driven hardware test architecture that can be modified easily using C and/or assembly languages and includes inexpensive model and simulation environments. Manufacturer component models can be used for all ICs keeping the development cost of this instrument down. This development must be functional within hours and not days of work. This was done by using an LF design built to support a HV sip and copying the serial interface code used to support that design.

A simple PIC16F1823 design was used to support the programmable needs. HyperTerminal was selected to support the user interface requirements.

**2 Related Documentation**

The following documents were used to create and should be referenced when reviewing this document:

1. PIC12F/LF1822/PIC16F/LF1823 Data Sheet

2. Maxim +5V-Powered, Multichannel RS-232 Drivers/Receivers

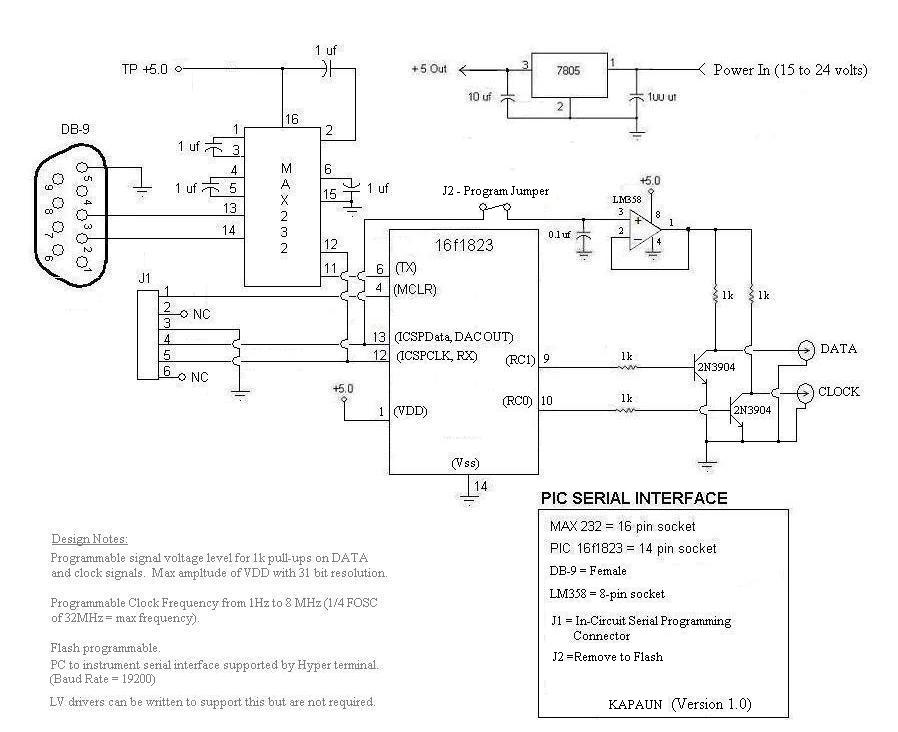
3. HI-TECH Software PICC Lite Ansi C Compiler

4. MPLAB IDE v8.56

5. PICKit 2 v2.61

6. Serial\_Write.mcw (This projects workspace.)

**3 Hardware Schematic**

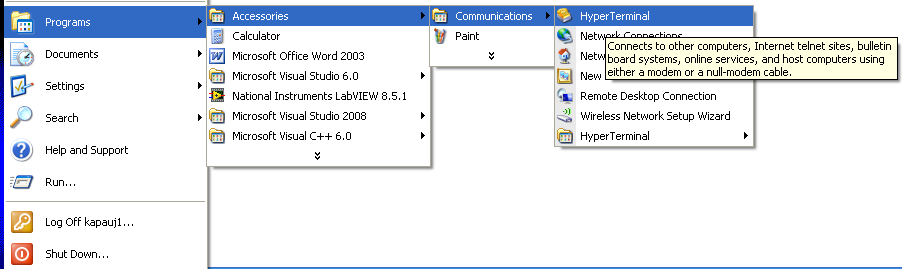


**4 User Interface (Hyper Terminal)**

The instrument can be exectuted using Hyper Terminal. Labview is not required which allows this tool to be used on lab and IT (desktop or laptop) pc configurations.

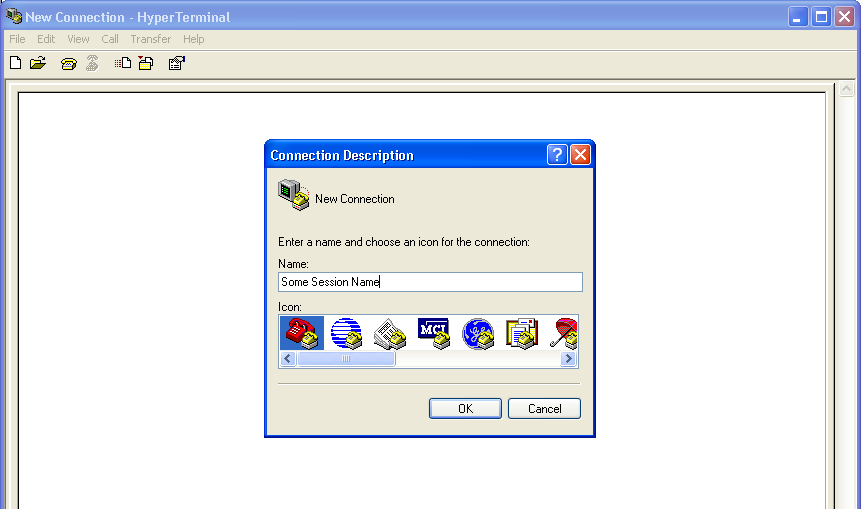
***4.1 Opening HyperTerminal***

HyperTerminal is located under ‘Communications’ as shown in the following diagram. Click on it to open a new session (it may look different on Windows 7 but will be located in the same place.) This tool is available on all lab and IT machines.

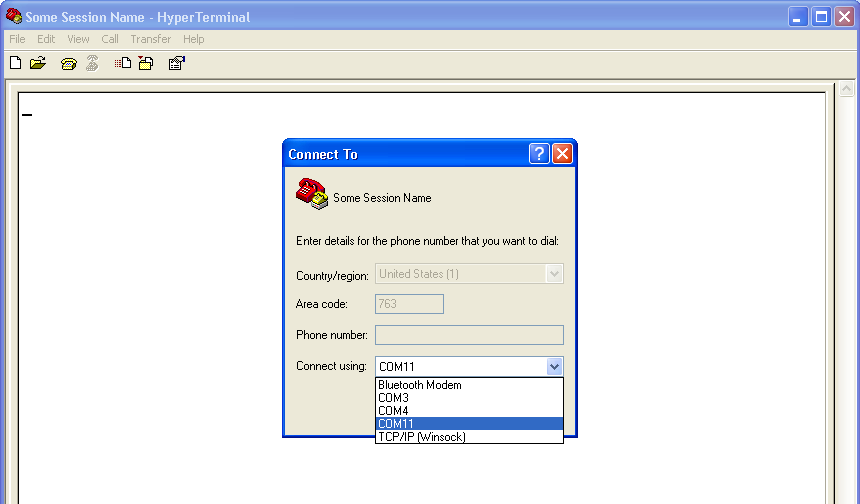


***4.2 Setting Session parameters HyperTerminal***

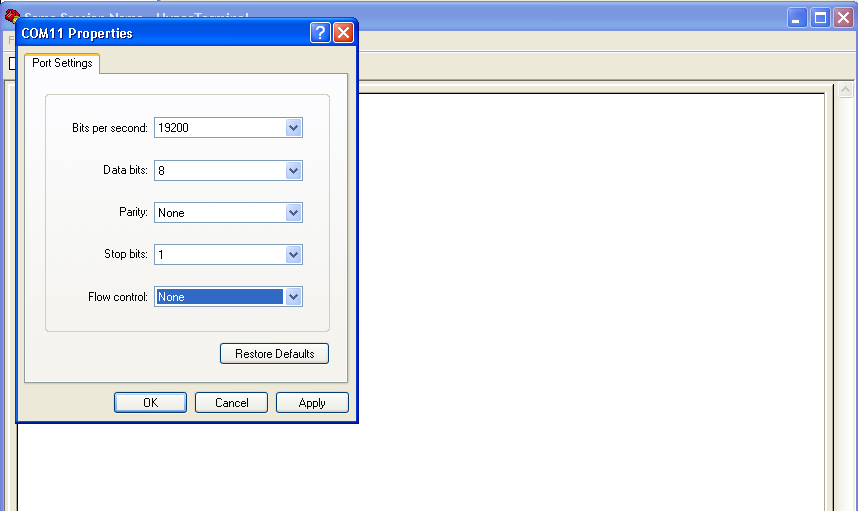
When HyperTerminal opens, you will be prompted to name your new connection as shown below. This will allow you to save the connection configuration which allows you to avoid having to go through these steps each time you open a session with the device. Enter a name and press ‘Okay’.



Next you will be asked ‘What you will be using to connect’. Pull down the menu under ‘Connecting using’ and select the Com Port you are using. In the picture below, we are using Com11. When you select it, the phone number items will become un-selectable as shown. Press ‘OK’ to continue.



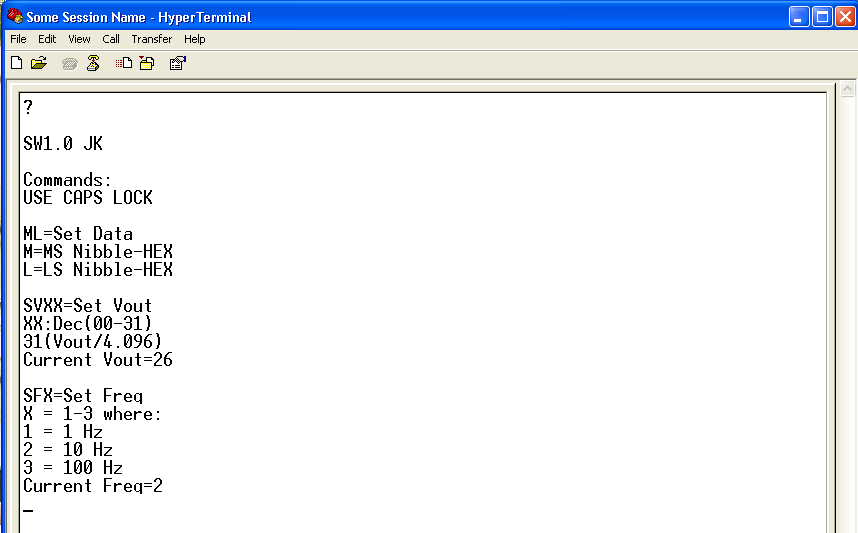
The communication port property settings will then be request. This instrument is configured to run at 19200 bps with no flow control. Select the port settings as follows and press ‘Apply’. Then press ‘OK’.



You are now able to communicate with the instrument. Turn ‘CAPS LOCK’ on.

***4.3 Help command ‘?’- HyperTerminal***

Now that the session has been established you can get the FW version information, command help and current instrument setting by entering ‘?’ as shown below:



Use ‘?’ to keep track of your current Vout (supplied to 1 k pull-ups) and Frequency settings. Modifying these values will only remain while the instrument remains powered up. These default powered on values can be modified by changing the following definitions contained in utility.c of the project:

// This defines the power on values for Vout and Frequency

volatile unsigned char Vout\_ASCII\_Hi = 0x32; // Vout for User Interface (Upper Char, 2X)

volatile unsigned char Vout\_ASCII\_Lo = 0x36; // Vout for User Interface (Lower Char, X6)

volatile unsigned char CLOCK\_FREQUENCY = 2; // Clock frequency

volatile unsigned char CLK\_ON\_TIME = 50; // 50 mSec On and 50 off for 100 mSec (10 Hz)

Three functions are shown:

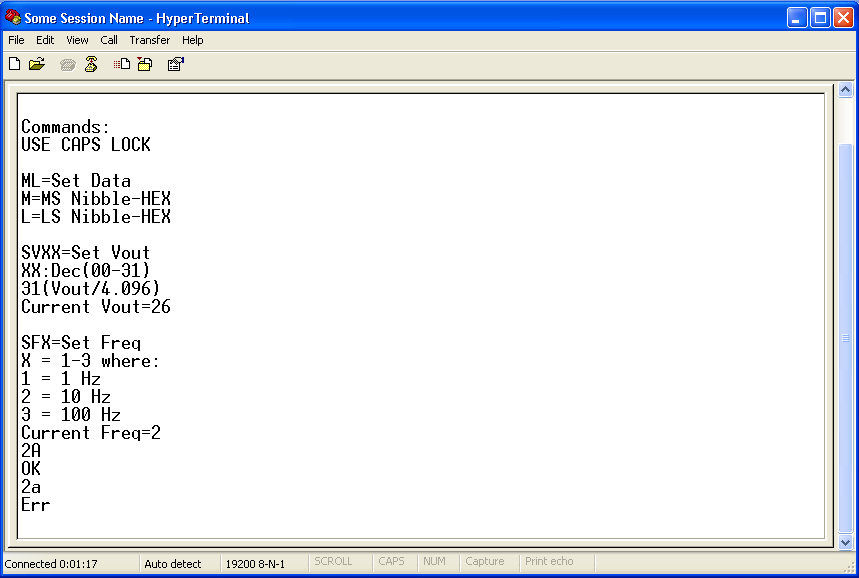
1. ML = Set Data (Writes data)

2. SVXX = Set Vout (default = 3.435 V)

3. SFX = Set Frequency (default = 10 Hz)

***4.4 Send Data: ML = Set Data***

This is the most used function. All that is required is for the Hex value to be typed in. Use CAPS Lock set to On. Hyper Terminal sends the ASCII characters to the instrument and the instrument does no casting to deal with lower case values. This means that 2A is represented differently then 2a. An error (Err) will be reported if an invalid value is sent and an ‘OK’ will be sent upon the completion of programming a valid setting. The following shows a ‘2A’ followed by a ‘2a’ being sent.



In the case that an error occurs, re-send the entire byte. This look up could be modified by changing the ASCII lookup table defined in set\_data.c.

unsigned char ASCII[17] = {'0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F', '\0'};

***4.5 SVXX = Set Vout (default = 3.435 V)***

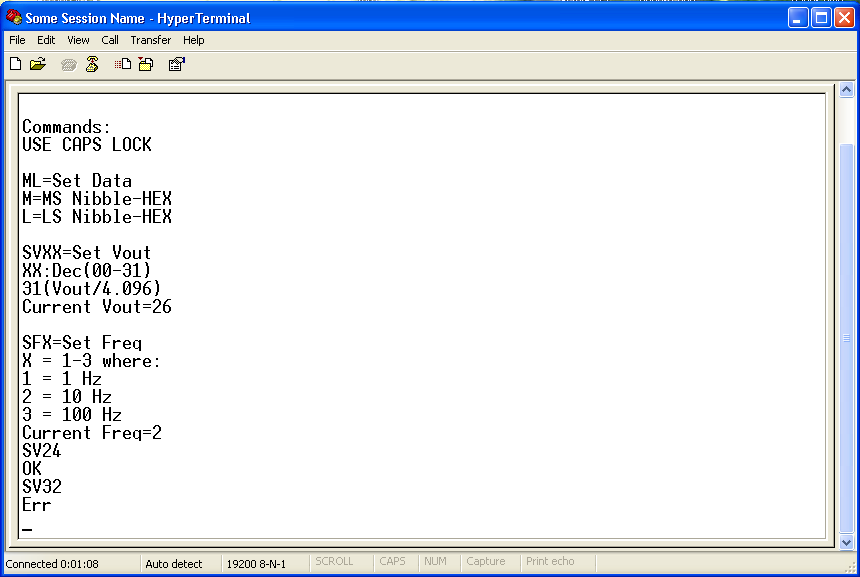
This sets the new Vout values. DACOUT was configured in main.c to use a fixed voltage reference. DACOUT was also enabled so changes will simply adjust the value being outputted to the 1k pull-up resisters. With the current configuration, the vout supplied can be calculated as follows:

DACCON1 (~25.7 dec) = Volts Want (3.4V) / Reference Used (4.096V) \* Scale (31)

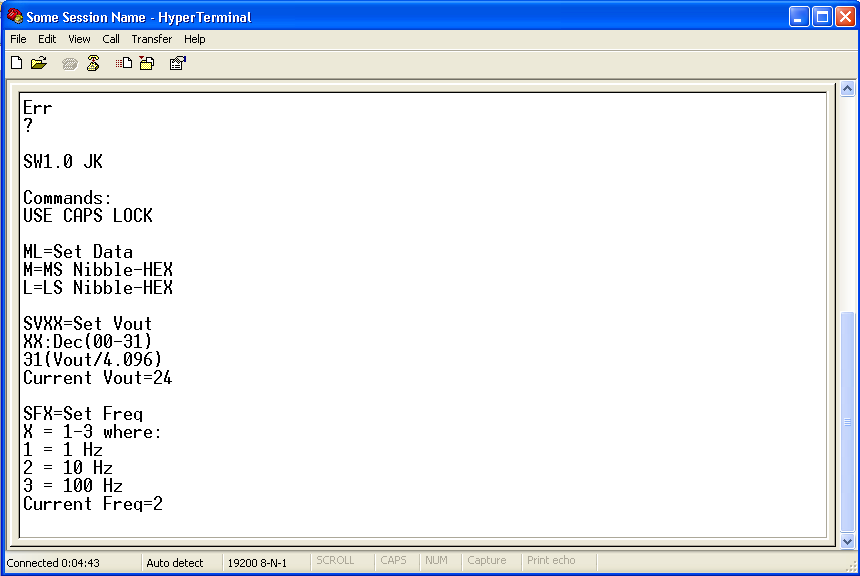
The max value could be set to VDD (5 volts) but requires a change to FVRCON in main.c.

To change this setting type ‘SV’ followed by the decimal value you calculated. If the value is valid (between 00 and 31) an ‘OK’ message will be returned after the change completes. If the value is not valid an ‘Err’ message will be returned. Vout will be set to the last valid setting. This can be checked by typing ‘?’ and reviewing the Current Vout value.

In the following example, a valid value of 24 is sent followed by an invalid value of 32.



‘?’ is now sent to show that the last valid value of ‘24’ is currently set (see Current Vout =24).



The last point to be made about this is that values between 0 and 9 must be entered as two characters such as: SV00, SV01, SV02….

***4.6 SFX = Set Frequency (default = 10 Hz)***

This sets the clock/data frequency. Currently this is all timed off the global ms timer because the rates required are very slow. There are currently 3 supported settings as follows:

1 - 1Hz

2 - 10Hz

3 - 100Hz

This means that there are only 3 valid commands that can be sent. They are:

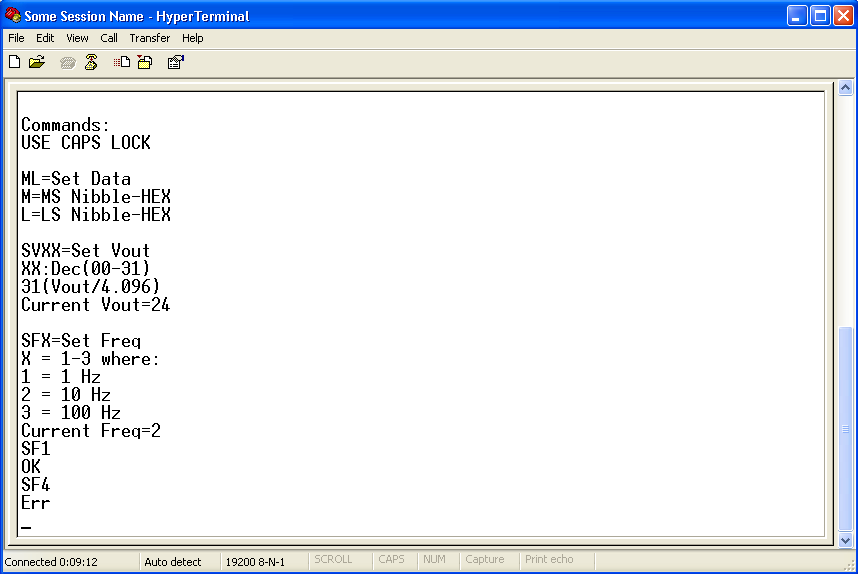
1. SF1 (set frequency to 1Hz)

2. SF2 (set frequency to 10Hz)

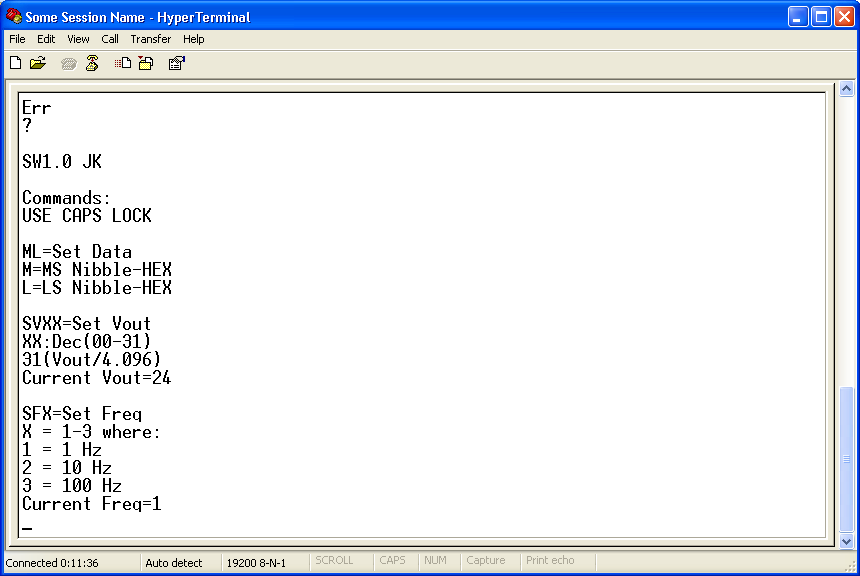
3. SF3 (set frequency to 100Hz)

If a valid setting command is sent an ‘OK’ message will be returned. If a value is sent other then was described above, and ‘Err’ will be returned and the frequency will be set to the last valid setting provided.

The following show a valid SF1 being sent followed by an invalid SF4.



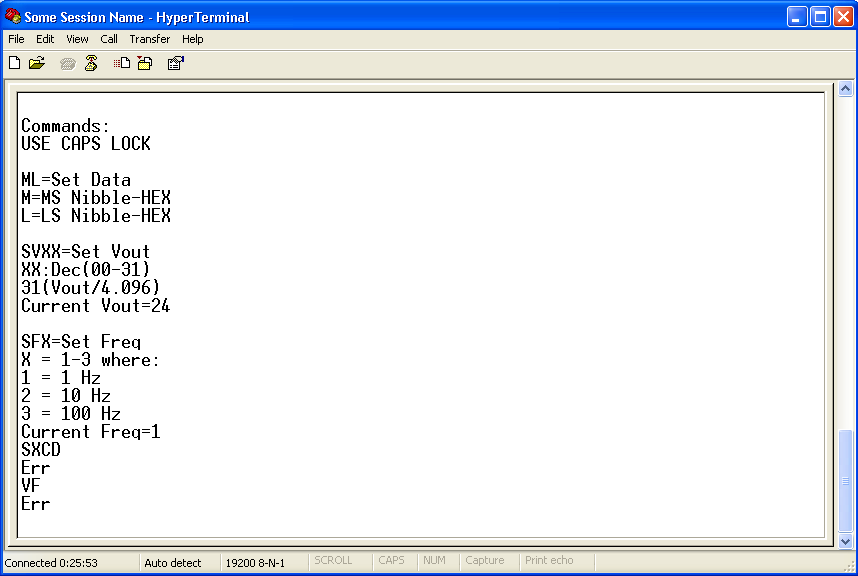
‘?’ is now sent to show that the last valid value of ‘SF1’ is currently set (see Current Freq =1).



The last point to be made about this function is that these very slow frequency setting could be changed in flash with the highest frequency available be equal to FOSC/4 or 8MHz.

***4.7 What if I send a Command not already covered?***

The output of the device will only be modified if the command sent is valid. If you start your command with something other then a hex value (in upper case) or ‘S’ then continue entering characters until an ‘Err’ message is returned. This will take 2 characters to be entered or 4 if you started the command with a set instruction (‘S’). The Rx buffer will then be cleared and you can start over. The following shows a 4 character error instruction followed by a two character error instruction:



Pressing ‘?’ will always display the current instrument settings.