Application Note YYY

3Flex 3500: Collecting, Accessing, and Using Transient Data

Andrew D. D'Amico and Bill Conklin

The 3Flex 3500 has the added capability to collect, access, and use on-demand transient data (pressures, temperatures, quantities adsorbed, etc.) during adsorption and desorption measurements. Data are collected every half second. This application note is a tutorial on how to collect the data being transmitted over the Ethernet cable used for instrument communication using a client such as PuTTY or HyperTerminal. Also, it will be shown how the captured data can easily be handled using scripts created in languages such as Python.

INSTALLING THE PuTTY CLIENT

PuTTY is an open source (free to download) SSH and telnet client that can easily be used to access all of the transient data output by the 3FLEX 3500 instrument. Since PuTTY is open source and easy to download, it will be the client discussed in this tutorial. No program installation is required. The .exe file runs directly from a Windows desktop. The software can be downloaded online (only save the putty.exe file):

http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

Figure 1 shows how the PuTTY icon should look on the desktop.

Figure 1. PuTTY client logo as shown on the desktop.

ACCESSING THE DATA

In order to connect to the instrument, the following items have to be done (shown in Figure 2):

1. Input 3FLEX IP address, found in the Unit Configuration, into the PuTTY software
2. Input 54000 in the 'Port' field
3. Select 'Raw' connection type
4. Save session settings (optional)
5. To save the data, go to the 'Logging' settings (see Figure 3).
    a. Select 'All session output'
    b. Select destination (using 'Browse…') to save file and create a .txt file (helpful to name this file the same as your sample file to be analyzed)
6. Click 'Open' to start collecting the data (note: this program can be started before the 3FLEX starts collecting data—it will be 'waiting' for when data output from the 3FLEX begins).

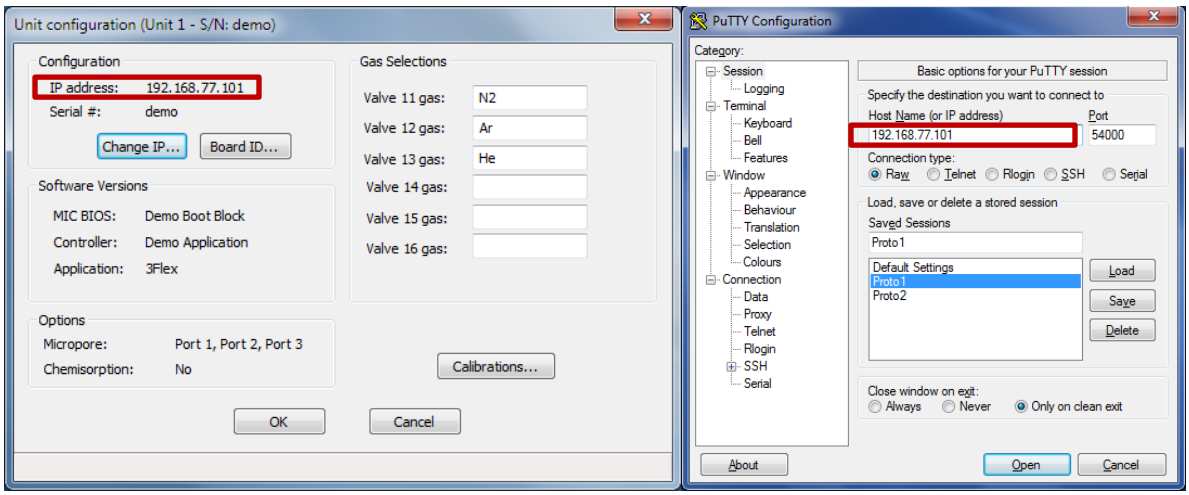


Figure 2. Obtaining the 3FLEX IP address from the Unit Configuration dialog box (left) and corresponding settings in PuTTY (right) in order to access the transient data.

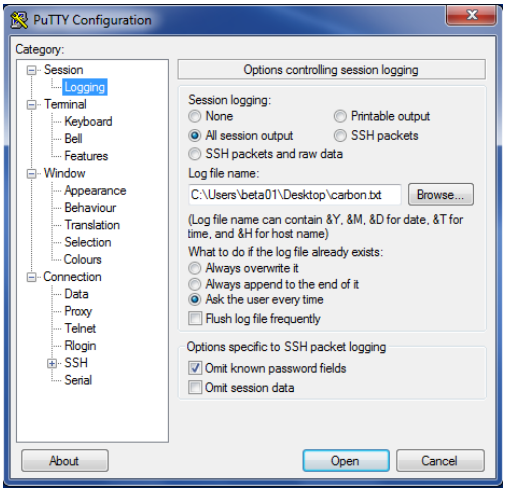


Figure 3. Proper settings to save the transient data to a file named 'carbon.txt'.

USING THE DATA

The text file containing the transient data can be accessed using any means that accepts tab separated values. Possibilities include Notepad, Microsoft Excel (or other spreadsheet software), or programming languages that read .txt or .xls files. Within programming languages, MATLAB®, Octave, and Python may be a few that come to mind. MATLAB® has the `xlsread()` function and Octave has the `textread()` function which are easy to use. However, the use of Python is chosen as the focus of this application note since Python is free to use, well supported in terms of online documentation / user forums, and the use of python code is also possible within the 3FLEX software (sample files). Please refer to Appendix I for a description of all of the data contained within the saved file. Refer to Appendix II for installing the proper components of Python, and refer to Appendix III for example code and how to execute the use of Python to view transient data on demand.

APPENDIX I: DESCRIPTION OF ALL COLUMNS IN DATA FILE

The data file generated during this collection process has 58 columns of tab-delimited, or separated, values. When viewed in Microsoft Excel or other spreadsheet program, the data occupy columns 'A' though 'BF'. The identity of each of these columns of data is given below:

a= 'manifold pressure (torr)';
b = 'manifold volume (cc)';
c = 'manifold temperature (K)';
d = 'time (ms)';

e = 'port 1 pressure (torr)';
f = 'port 1 volume (cc)';
g = 'port 1 temperature (K)';
h = 'port 1 ambient temperature (K)';
i = 'port 1 analysis temperature (K)';
j = 'port 1 warm freespace (cc STP)';
k = 'port 1 cold freespace (cc STP)';
l = 'port 1 quantity dosed (cc STP)';
m = 'port 1 quantity adsorbed (cc STP)';
n = 'port 1 has manifold ?';
o = 'port 1 valve open ?';
p = 'port 1 data points taken';
q = 'port 1 current pressure table index';
r = 'port 1 last data point elapsed time (ms)';
s = 'port 1 last data point pressure (torr)';
t = 'port 1 last data point quantity dosed (cc STP)';
u = 'port 1 last data point quantity adsorbed (cc STP)';
v = 'port 1 last data point Po (torr)';

w = 'port 2 pressure (torr)';
x = 'port 2 volume (cc)';
y = 'port 2 temperature (K)';
z = 'port 2 ambient temperature (K)';
aa = 'port 2 analysis temperature (K)';
ab = 'port 2 warm freespace (cc STP)';
ac = 'port 2 cold freespace (cc STP)';
ad = 'port 2 quantity dosed (cc STP)';
ae = 'port 2 quantity adsorbed (cc STP)';
af = 'port 2 has manifold ?';
ag = 'port 2 valve open ?';
ah = 'port 2 data points taken';
ai = 'port 2 current pressure table index';
aj = 'port 2 last data point elapsed time (ms)';
ak = 'port 2 last data point pressure (torr)';
al = 'port 2 last data point quantity dosed (cc STP)';
am = 'port 2 last data point quantity adsorbed (cc STP)';
an = 'port 2 last data point Po (torr)';

ao = 'port 3 pressure (torr)';
ap = 'port 3 volume (cc)';
aq = 'port 3 temperature (K)';
ar = 'port 3 ambient temperature (K)';
as = 'port 3 analysis temperature (K)';
at = 'port 3 warm freespace (cc STP)';
au = 'port 3 cold freespace (cc STP)';
av = 'port 3 quantity dosed (cc STP)';
aw = 'port 3 quantity adsorbed (cc STP)';
ax = 'port 3 has manifold ?';
ay = 'port 3 valve open ?';
az = 'port 3 data points taken';
ba = 'port 3 current pressure table index';
bb = 'port 3 last data point elapsed time (ms)';
bc = 'port 3 last data point pressure (torr)';
bd = 'port 3 last data point quantity dosed (cc STP)';
be = 'port 3 last data point quantity adsorbed (cc STP)';
bf = 'port 3 last data point Po (torr)';

APPENDIX II: PYTHON INSTALLATION COMPONENTS

In order to operate Python in a way that on-demand data can be viewed, 3 components need to be installed: Python 3.2, NumPY (for numerical functions), and MatPlotLIB (for graphical display of the data).

INSTALLING PYTHON 3.2

Installation of Python needs to be done first. Python version 3.2.3 can be downloaded from python.org:

http://www.python.org/getit/releases/3.2.3/

Depending on your computer system, appropriate versions may vary, but the following file should be compatible with most, if not all, computers that operate the 3FLEX instrument:

Windows x86 MSI Installer (python-3.2.3.msi)

INSTALLING NumPY

NumPY can be install next and contains numerous numerical functions and libraries which are needed in order to properly handle the transient data extracted from the .txt file collected during an analysis. NumPY version 1.6.2 should be download to be compatible with the installed version of Python. One download location is sourceforge.net:

http://sourceforge.net/projects/numpy/files/NumPy/1.6.2/

The corresponding file to download is:

numpy-1.6.2-win32-superpack-python3.2.exe

INSTALLING MatPlotLIB

The last necessary component of the installation is MatPlotLIB. Version 1.2.1 is compatible with the previously mentioned / installed software and can be downloaded:

http://www.lfd.uci.edu/~gohlke/pythonlibs/#matplotlib

The proper file to download is:

matplotlib-1.2.1rc1.win32-py3.2.exe

APPENDIX III: SAMPLE PYTHON CODE

The below is example code that was used to produce the data shown in Figure A3.1. This is the most basis code need to see transient data, but the flexibility exists to script a more detailed file—allowing for flexibility such as comparing pressure or quantity adsorbed between ports, generating a separate figure for each port, etc.

```
import numpy as np
import matplotlib.pyplot as plt

myFile = 'carbon.txt'
data = np.genfromtxt(myFile, skip_header=2, skip_footer = 1)

# Time (minutes)
x=data[:,3]
x = x/1000/60

# Sample pressure PORT 1(torr)
y=data[:,4]

plt.plot(x,y,'ko')
plt.ylabel('Pressure (torr)')
plt.xlabel('time (minutes)')
plt.show()
```

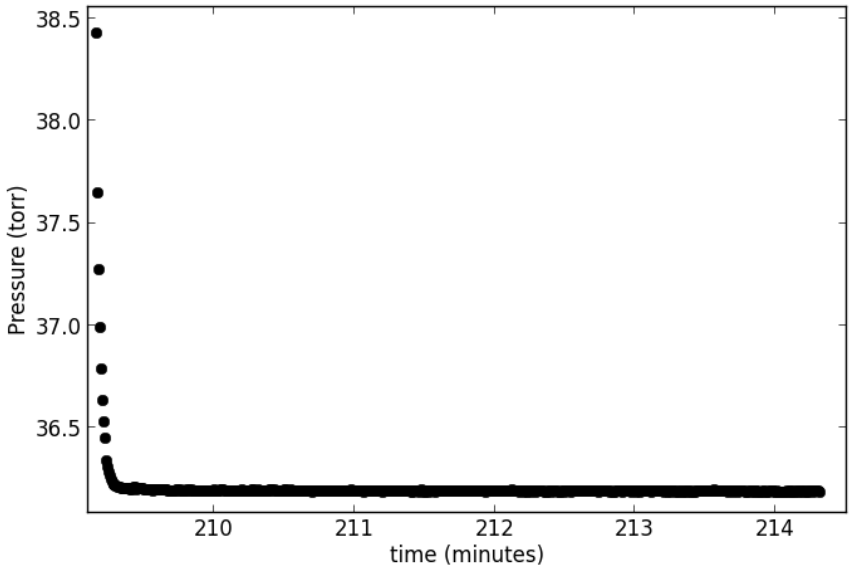


Figure A3.1. Example pressure data obtained from transient data recording.

In order to take advantage of the power of Python programming, understanding of the above sample code is needed.

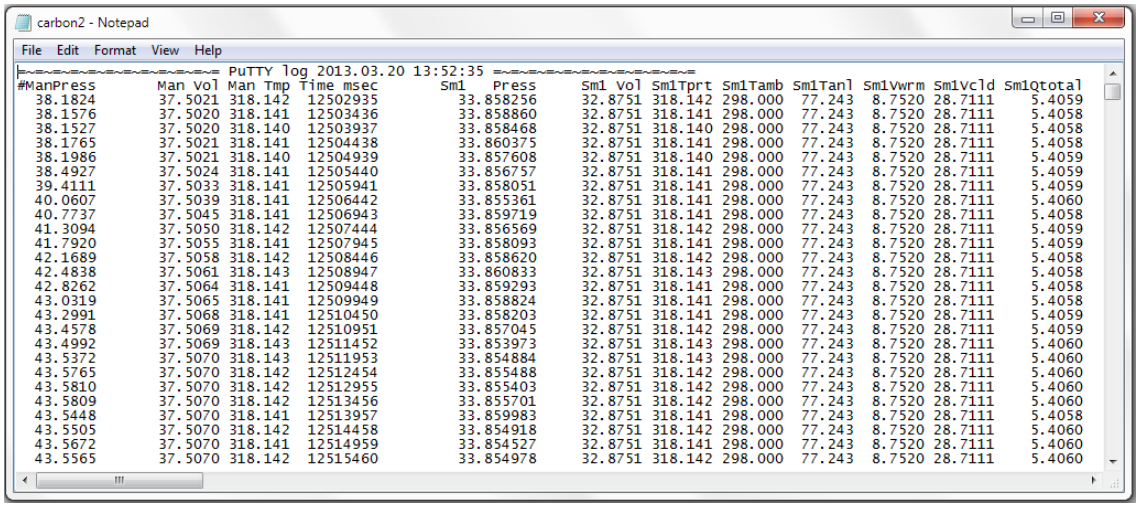
```
import numpy as np
import matplotlib.pyplot as plt
```

Import is needed to load the modules containing functions to be used—in this case, numerical and plotting functions. Coding is easier if abbreviations such as np and plt are used.

```
myFile = 'carbon.txt'
data = np.genfromtxt(myFile, skip_header=2, skip_footer = 1)
```

In this code, 'MyFile' is a variable name for the .txt file to be read. As long as the .txt file and the script file are in the same directory (eg on the desktop), the script file can run automatically by double-clicking on the script file after the correct file has been hard-coded once.

The genfromtxt() function contained in NumPY allows for the text file to be converted into a numerical matrix. The skip_header setting allows for the first 2 rows of data to be excluded from the matrix. As seen in Figure A3.2, the first row of data is ASCII "art" created by the PuTTY program, and the second row of data are the column labels—neither of which are numerical data. The skip_footer setting is needed because, when the .txt file is read on demand, the entire last row of data may not be complete. This causes problems for creating the matrix and needs to be excluded. This means that the last half-second of data is not displayed when the data are viewed on demand.


carbon2 - Notepad

File Edit Format View Help

PuTTY log 2013.03.20 13:52:35

#ManPress	Man Vol	Man Tmp	Time msec	Sm1	Press	Sm1 Vol	Sm1Tprt	Sm1Tamb	Sm1Tanl	Sm1Vwrm	Sm1Vcld	Sm1Qtotal
38.1824	37.5021	318.142	12502935		33.858256	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4059
38.1576	37.5020	318.141	12503436		33.858860	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
38.1527	37.5020	318.140	12503937		33.858468	32.8751	318.140	298.000	77.243	8.7520	28.7111	5.4058
38.1765	37.5021	318.141	12504438		33.860375	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
38.1986	37.5021	318.140	12504939		33.857608	32.8751	318.140	298.000	77.243	8.7520	28.7111	5.4059
38.4927	37.5024	318.141	12505440		33.856757	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4059
39.4111	37.5033	318.141	12505941		33.858051	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4059
40.0607	37.5039	318.141	12506442		33.855361	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4060
40.7737	37.5045	318.141	12506943		33.859719	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
41.3094	37.5050	318.142	12507444		33.856569	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4059
41.7920	37.5055	318.141	12507945		33.858093	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4059
42.1689	37.5058	318.142	12508446		33.858620	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4058
42.4838	37.5061	318.143	12508947		33.860833	32.8751	318.143	298.000	77.243	8.7520	28.7111	5.4058
42.8262	37.5064	318.141	12509448		33.859293	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
43.0319	37.5065	318.141	12509949		33.858824	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
43.2991	37.5068	318.141	12510450		33.858203	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4059
43.4578	37.5069	318.142	12510951		33.857045	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4059
43.4992	37.5069	318.143	12511452		33.853973	32.8751	318.143	298.000	77.243	8.7520	28.7111	5.4060
43.5372	37.5070	318.143	12511953		33.854884	32.8751	318.143	298.000	77.243	8.7520	28.7111	5.4060
43.5765	37.5070	318.142	12512454		33.855488	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4060
43.5810	37.5070	318.142	12512955		33.855403	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4060
43.5809	37.5070	318.142	12513456		33.855701	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4060
43.5448	37.5070	318.141	12513957		33.859983	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4058
43.5505	37.5070	318.142	12514458		33.854918	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4060
43.5672	37.5070	318.141	12514959		33.854527	32.8751	318.141	298.000	77.243	8.7520	28.7111	5.4060
43.5565	37.5070	318.142	12515460		33.854978	32.8751	318.142	298.000	77.243	8.7520	28.7111	5.4060

Figure A3.2. Sample of transient data displayed in a text editor. Note the ASCII art and labels of the first two rows of data—which need to be excluded from the numerical data.

```
# Time (minutes)
x=data[:,3]
x = x/1000/60
```

Appendix I of this document identifies the columns of data recorded. The fourth column of data contains time in milliseconds. However, the first index in Python is 0. Therefore, `data[:,3]` captures all of the rows (:) in the fourth column.

```
plt.plot(x,y,'ko')
plt.ylabel('Pressure (torr)')
plt.xlabel('time (minutes)')
plt.show()
```

The above code shows the syntax for plotting. The label 'ko' means filled black circles.

While this code is very basic and only begins to explore the possible script configurations, scripts with greater power and flexibility can readily be created. Micromeritics' applications specialists would be glad to discuss and support creating custom scripts for individual applications.