

# Model 378x-PC Water-based Condensation Particle Counter

## Access and Use the PC/104 Single Board Computer (SBC)

### 1. Overview

A Water-based Condensation Particle Counter (Model 3782/3785/3786 WCPC) with the -PC option is assembled with an internal PC/104-style Single Board Computer (SBC). This optional PC/104 SBC offers internal data logging, real-time clock, and remote data access capabilities for the WCPCs.

The PC/104 SBC is connected internally to the WCPC as shown in Fig. 1. Two of the four connectors on the back panel of the WCPC (PC/104 Serial and PC/104 Ethernet) provide access to the SBC. The other two connectors (Serial and USB) on the back panel directly connect to the WCPC. Communications to the WCPC should occur either through the internal serial connection to the SBC or to an external device connected to the Serial or USB port, but not both. Attempting simultaneous commutations from both interfaces will result in loss of data.

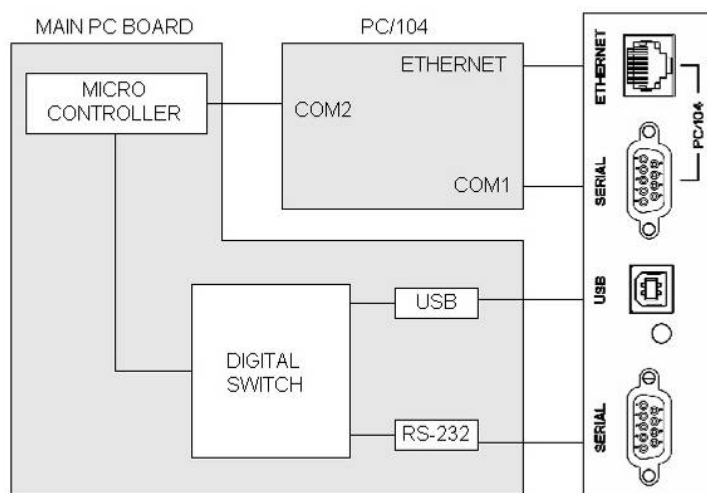


Figure 1. WCPC data communications diagram

Currently, the SBC used within the WCPC is a PCM-3347 or a PCM-3348 manufactured by EMAC, Inc. (<http://www.emacinc.com>). The SBC is loaded with the Linux operating system. It is expected that the user has a working knowledge of Linux and some programming experience in order to use the optional SBC.

## 2. Making a connection

### 2.1 Access

In order to configure the SBC to control the WCPC and/or log data, the user must access the SBC through a Linux console. Connection to a Linux console (bash shell) can be made through either the PC/104 Serial Port or through the Ethernet interface. Using the Linux console, users can modify configurations, establish terminal access to the WCPC, and execute programs.

### 2.2 Serial interface using terminal

The PC/104 Serial Port is set up as a terminal server for the PC/104 SBC. A terminal or computer running terminal emulation software can access the SBC through this port. A Null Modem (crossover) cable is needed to connect this port to a personal computer. The default serial settings for the PC/104 terminal server are:

```
Baud Rate ..... 9600
Bits/Character ..... 8
Stop bits ..... 1
Parity ..... None
```

Once a connection is made, a login prompt will be provided and the user may login using the username and password provided in section 2.4.

### 2.3 Ethernet

#### 2.3.1 Finding IP address

Before a connection can be made to the SBC through the Ethernet port, its IP (Internet Protocol) address must be known. The SBC Ethernet port (eth0) is initially configured to gather an IP address from an external DHCP (Dynamic Host Configuration Protocol) server. The IP address is provided in a dotted-quad format, for example: 192.168.10.5. For the SBC to actually acquire an IP address, it must be connected to a network with an active DHCP server BEFORE powering up the WCPC. The connection between the Ethernet port and the network must be made with a shielded twisted pair 10-BaseT network cable. A cable equipped with a ferrite interference filter should be used with the 3785-PC to reduce electromagnetic interference. Determining the IP address assigned by the DHCP server to the SBC can be a challenge. Some possible ways to find the assigned IP address include:

- Query the leases distributed by the DHCP server
- Trial and error attempts using ping or telnet to find the IP addresses if the local subnet is small.
- Use of an nmap utility to scan for IP addresses on the local network.
- Querying the SBC directly through the terminal server on its serial port, using the console command ipconfig.

The Ethernet configuration may be changed to provide a static IP address using the start up configuration routine provided once an initial login with the SBC is achieved.

### 2.3.2 Telnet

Once the SBC IP address is determined, the instrument can be remotely accessed using a telnet utility. To telnet from Windows, choose **Run** from main **Start** menu. Enter **cmd** and select **OK** to open a command window. Enter **telnet [IP address]** and a prompt for user name and password will come up for the user to log in (see section 2.4).

## 2.4 Login

The default user name and password for the SBC are:

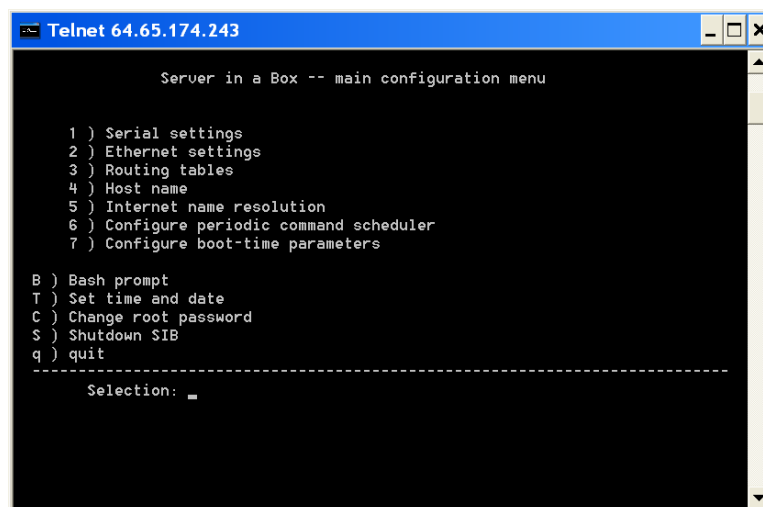
```
User Name:    root
Password:    emacs_inc
```

This login provides root level access to the SBC with complete access to all files and devices. The user may want to create non-root level user accounts that provide for more secure operation of the SBC.

## 3. Data collection

### 3.1 Main configuration menu

After user login, a main configuration menu will show up (see Fig. 2). Settings that can be configured through this menu include serial ports, Ethernet ports, time, and password etc. The details of this menu are listed in Table 1 in the Appendix. There is no need to change the settings before starting the data collection. Press **q** to exit the main configuration menu and a prompt **SIBx24.emacinc.com:** shows up.



```
Telnet 64.65.174.243
Server in a Box -- main configuration menu

1 ) Serial settings
2 ) Ethernet settings
3 ) Routing tables
4 ) Host name
5 ) Internet name resolution
6 ) Configure periodic command scheduler
7 ) Configure boot-time parameters

B ) Bash prompt
T ) Set time and date
C ) Change root password
S ) Shutdown SIB
q ) quit
-----
Selection: _
```

Figure 2. Main configuration menu

### 3.2 Data logging using WCPCLog

#### 3.2.1 Start/Stop data logging

A demonstration C program named “WCPCLog” has been included in the directory `/home/WCPC`. This program captures the data (D) records returned by the WCPC, timestamps them, and stores them. To enter the `/home/WCPC` directory, type in `cd /home/WCPC`. To start the program, enter “`./WCPCLog [sample interval in`

**deciseconds] [start\_minute] [start\_hour]”**. For example, “./WCPCLog 60 00 17” will log data once every six seconds starting at 17:00. The commands are case sensitive. To stop logging data, press **Ctrl+C**. Once WCPCLog is running, it will set the SBC’s file system to ‘read-only’ when it logs its first data record. The WCPCLog program must be stopped and the file system set to ‘read-write’ before files can be deleted or modified (see section 4.2)

### 3.2.2 Data storage

The logged data shown in the window is also stored into a file named with the current date, e.g., data collected on March 17, 2005 is stored in a file named 20050317. Each line of data record has a date stamp in the format of yyyy/mm/dd and a time stamp in the format of hh:mm:ss (Fig. 3).

Data collected during separate time periods on the same day will be stored in the same data file. Data collected later during the day will append to the end of the file. A new data file is created each day for data logging processes that run pass midnight.

Data is stored in a Compact Flash card, typically 128 Megabyte in size. Each data record takes about 60 bytes. A flash card of this size is capable of storing over 3 years of data with the sample interval set at 1 minute.

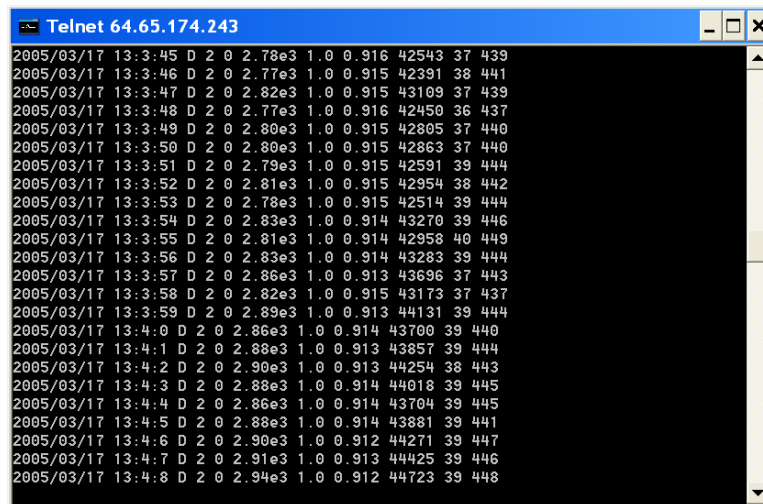


Figure 3. One-second interval data records

### 3.2.3 Logging data in background

The data logging process can be running in background by adding an ‘&’ to the end of the command, e.g. **./WCPCLog [sample interval in deciseconds] [start\_minute] [start\_hour] &**. Running the data logging process in background allows the user to perform other operations such as accessing other files while collecting data. This background data logging can only be stopped by the command **kill [process ID]**, or **killall WCPCLog**. The process ID can be looked up by using the **ps** command as shown in Fig. 4. In the figure, the ID for the WCPCLog process is 1646. This process can be stopped by typing in “kill 1646”.

```

Telnet 64.65.174.243
ps
PID  Uid  Stat  Command
1    root  S     init [2]
2    root  S     [keventd]
3    root  S     [ksoftirqd_CPU0]
4    root  S     [kswapd]
5    root  S     [bdflush]
6    root  S     [kupdated]
72   root  S     /sbin/pump -i eth0
75   root  S     [eth0]
101  nobody S     /usr/bin/boa
105  root  S     /usr/sbin/inetd
109  root  S     /sbin/syslogd
112  root  S     /usr/sbin/cron
118  root  S     /sbin/getty 38400 tty1
119  root  S     /sbin/getty 38400 tty2
120  root  S     /sbin/getty 38400 tty3
121  root  S     /sbin/getty -L ttyS0 9600 vt100
122  root  S     in.telnetd: Fredq-linux.powerengmfg.com
123  root  S     -bash
145  root  S     in.telnetd: 206.165.115.194
146  root  S     -bash
1646 root  S     ./WCPCLog 20 54 13
6915 root  R     ps
SIBx24.emacinc.com#:2005/03/17 15:21:44 D 2 0 6.43e3 2.0 1.659 178056 191 648

```

Figure 4. List of running processes

### 3.3 Remote data access using an Internet web browser

A symbolic link to the /home/WCPC directory is included in /home/www directory so files created by the WCPCLog program can be accessed via the internet web browser by typing in the address **http://[IP Address]/WCPC/**. This web page shows a list of file names in the /home/WCPC directory that can be accessed. Double click the name of a data file shows the data stored in the file. The data can then be copied and pasted into another application such as a spreadsheet program. Data pasted into a Microsoft Excel worksheet can be easily expanded into column data by selecting the **Data** menu and then **Text to Columns**.

## 4. Information for advanced users

### 4.1 Serial communications program – minicom

The terminal emulation program **minicom** has been configured to use one of serial ports (/dev/ttyS1) to communicate with the WCPC through an internal connection. This program may be used to test operation with the WCPC and even to capture data (Fig. 5). The program can be run from the Linux console by typing **minicom**. The warning about running minicom as root can be ignored. The local echo of the commands entered can be turned on by pressing **Ctrl+A** and then **E (Ctrl+A+E)**. The appending of linefeeds should also be enabled by pressing **Ctrl+A** and then **A (Ctrl+A+A)**. It should be noted that the SBC is setup with a “read-only” file system. This insures that when the WCPC is turned off and the SBC is brought down “hard” and the SBC will boot cleanly on the next power up. This also means that for data to be saved to the SBC it must be either written to the “/var” directory which is mounted read-write in RAM, or the file system must be mounted read-write during the moments when data is to be written.

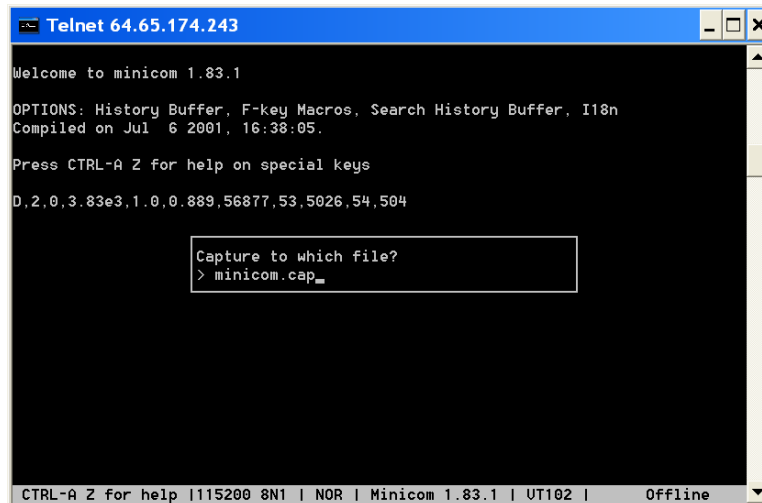


Figure 5. Using Minicom to capture data

Use **Ctrl+A+L** to start/stop capturing data with minicom. A prompt will show up to ask for the file name. Enter **/var/[file name]** to make sure the data is captured into the “/var” directory. Notice that the file stored in /var directory will be lost if the WCPC is turned off and the /var directory has only a limited capacity for data storage. To store the data into other directories, you can run the minicom program under the **Bash Prompt** of the main configuration menu while the file system is mounted read-write. It should also be noted that data captured with minicom is not time-stamped. Press **Ctrl+A+Q** to quit the program.

The WPCLog program is written specifically to log data and is the recommended choice for collecting data as described in Section 3.

## 4.2 Configuration menu

The main configuration menu is invoked on login and can also be started by entering **/root/SIBconfig.sh**. Upon execution the menu program sets the Linux file system to ‘read-write’ and back to ‘read-only’ when exited. The bash shell option allows access to a console while with ‘read-write’ access to the file system. It should be noted that if the WPCLog program is running, it will reset the file system to ‘read-only’. To edit, move or delete files on the SBC, the WPCLog program must be stopped before running the configuration menu program and the related console shell.

## 4.3 FTP access

In order to upload files to the WCPC from a remote computer via ftp as a root user, the “/etc/ftpusers” needs to be changed. While in a ‘read-write’ files system, edit the file (**edit /etc/ftpusers**) and place a ‘#’ in front of ‘root’ to comment out this line. This allows the user ‘root’ to use ftp for file transfers. For security, edit the ftpusers file back to turn off ftp access by user root once file transfers are complete. Alternatively, create a new user account with which to do file transfer via ftp.

Use **ftp [IP Address]** to access the WCPC. Login with the User Name and Password in section 2.4. Use the command **get [file name]** to transfer files from the WCPC to the remote computer and use **put [file name]** to transfer files from the computer to the WCPC.

#### 4.4 WCPCLog time synchronization

The WCPCLog program collects WCPC data and saves it with a time stamp from the SBC system clock. Since the WCPC clock drifts over time, a short script (`/home/WCPC/WCPC_start.sh`) has been created that is run by the SBC scheduler (cron) at a minute before midnight each day. The script kills any running WCPCLog processes, synchronizes the system clock using an `rdate` command, and then restarts WCPCLog to begin logging data at midnight. The script is executed each day, but to enable the WCPCLog program, the script must be edited to remove the comment symbol from the line that calls the program.

In a similar manner, the WCPCLog program can be made to run on power-up by uncommenting the call to the program in the start-up script `/etc/init.d/wcpc_start`.

### 5. Appendix

#### 5.1 Settings in main configuration menu

Table 1. Settings in Main Configuration Menu

Server in a Box - - main configuration menu			
	Main menu	Sub menu	Notes
1	Serial settings	1) Configure SLIP port 2) Configure terminal port 3) Configure dial-in terminal port 4) Configure dial-in PPP port 5) Configure dial-out PPP connection 6) Configure standard serial port 7) View serial setting	Port <code>/dev/ttys</code> is configured as a serial terminal server with Line speed = 9600 Terminal emulation = vt100
2	Ethernet settings	1) Configure Ethernet device 2) View network device setting	Currently, device <code>eth0</code> is configured as: Local IP address = DHCP; It can be changed to fixed IP address for the WCPC
3	Routing tables	1) Add routing entry 2) Delete routing entry 3) Set default gateway 4) View routing data	
4	Host name	Host Name Configuration	Current host name: <code>SIBx24.emacinc.com</code>
5	Internet name resolution	1) Add nameserver 2) Delete nameserver 3) Add static host entry 4) Delete static host entry 5) View name resolution configuration	
6	Configure periodic command scheduler		This is now handled by <code>/home/WCPC/WCPC_start.sh</code>
7	Configure boot-time parameters	1) Set root filesystem READ-ONLY 2) Set root filesystem READ-WRITE 3) Configure loadable modules / drivers	Default setting is READ-ONLY
B	Bash prompt	-	Choose this option to enter Linux commands
T	Set time and date		

Server in a Box - - main configuration menu			
	Main menu	Sub menu	Notes
C	Change root password		
S	Shutdown SIB		SIB: Server In a Box
q	quit		

## 5.2 Useful Linux commands

The following is a list of commonly used LINUX commands which may be of value. Remember that LINUX is case sensitive. Options which can be used with a command are placed in <>. The <> are not part of the command and should not be included in the command that you enter. A complete list of supported commands is available on the SBC web page ([http://\[IP Address\]](http://[IP Address])) under the busybox.html link.

logout	logs you out of a Telnet session
cat	concatenate and display
cd [directory]	change directory
cd ..	moves you backwards to the next higher subdirectory level
cd /	moves you to the highest directory level
clear	clears the screen
cp [oldfiles] [newfiles]	copies a file; this leaves the old file intact and makes a new copy with a new filename
date	tells you the current date and time
df <-h>	displays how much space on the disks is free
edit	starts an editor program
ftp	starts a file transfer program
get [remote-file]	ftp command to download the <i>remote-file</i> and store it on the local machine
hwclock	read or set the hardware clock
ifconfig	TCP/IP command to assign an address to a network interface and/or configure network interface parameters
kill [processID]	kills a process that has the ID of <i>processID</i>
ls <-l>	lists the files in a directory; <b>-l</b> displays detailed information about each file and directory
mkdir [new_directory]	makes a new subdirectory
more	Display a text file one page at a time
mv <-i> [oldname] [newname]	renames a file or moves it from one directory to another; the <b>-i</b> option tells mv to prompt you before it replaces an existing filename
ps	displays information about your processes/jobs/programs which are running on the server
put [local-file]	A telnet command to store a local file on the remote machine with the same file name
quit	ftp command to close the connection
rm <-i> [filenames]	removes or deletes files; the <b>-i</b> option asks you to confirm that you want to delete each file
rmdir <-i> [directory]	removes a directory; the <b>-i</b> option asks you to confirm that you want to delete the directory
telnet	open a connection to a remote system using Telnet protocol

### 5.3 Files of interest

**/etc/rc2.d/S99wpcp\_start** - Symbolic link pointing to script /etc/init.d/wpcp\_start used to run WPCPLog on start up

**/etc/localtime** - Symbolic line pointing to the zoneinfo file used for the local time zone.

**/etc/crontab** - Scheduler file with entry to run /home/WCPC/WCPC\_start.sh at one-minute before midnight.

**/etc/ftpusers** - File containing a list of user which may NOT access ftp server.

**/home/WCPC/WCPCLog.c** - Source file containing C code for the WPCPLog program.

**/home/www/WCPC** - Symbolic link to /home/WCPC to allow the WCPC directory to be viewed from the SBC web server

### 5.4 Listing of WPCPLog source code

```
// Copywrite (C) 2003 2005 Quant Technologies LLC

// Provided under the GPL (GNU Public License)
// WPCPLog is a program that acquires data from a water-based CPC
// WPCP-400/5/10 or TSI 3782/3785/3786
// The program Time stamps the data and writes it to a file.
// New files are created daily
// Version 0.80      28-Dec-03
// Converted from C++
// by Jacob Quant   05-Jan-05
// changes added arguments for sample time and start time
// timestamp by start of sample
#define VER "1.0"

#include <stdio.h>           // for sprintf
#include <stdlib.h>         // for system calls
#include <string.h>         // for str functions
#include <unistd.h>        // for read, write, sleep
#include <termios.h>
#include <time.h>
#include <fcntl.h>

int main( int argc, char **argv ) {
    int i;                  // general
    char buff[80];         // working buffer
    char buff2[20];
    FILE* filestream;     // handle to file for saving data
    char send_buff[100];  // buffer to build up data record for saving to file
    int fd;               // file device representing serial port
    struct termios options; // structure holding serial port option settings
    time_t now;           // used to hold current time queries
    struct tm* timestamp; // structure holding different parts of time format
    char rec_buff[80];    // buffer holding data received from serial port
    char* buffptr;       // pointer for filling rec_buff
    int nbytes;          // number of bytes in serial input buffer
    char date[12];       // date timestamp string YYYY/MM/DD
    char filename[12];   // log file name created from current date YYYYMMDD
    char sm_str[12]={"SM,2,60\r"}; // String used to set CPC acquire mode (SM command) 6 second
    default
    int start_min, start_hour; // Start min and hour used to time the sample start

    // Get command line arguments
    printf("WPCPLog Ver. %s \n",VER);
    printf("Usage: WPCPLog [sample interval in deciseconds] [start_minute] [start_hour]\n");
    if(argc>1){ // Sample interval argument (deciseconds)
        i=atoi(&argv[1][0]);
    }
}
```

```

if( i>= 1 && i<36000){
    strcpy(sm_str,"SM,2,"); // Build 'set mode' mode command
    strcat(sm_str, &argv[1][0]); // append sample interval time
    strcat(sm_str, "\r");
}
else{
    printf("Bad sample interval argument\n");
    return;
}
}
else
    strcpy(sm_str,"SM,2,60\r"); // Default sampling mode is 6 seconds

if(argc>2){ // Start minute argument
    start_min = atoi(&argv[2][0]);
    if(start_min<0 || start_min>59){
        printf("Bad start minute argument\n");
        return;
    }
}

if(argc>3){ // Start minute argument
    start_hour = atoi(&argv[3][0]);
    if(start_hour<0 || start_hour>23){
        printf("Bad start hour argument\n");
        return;
    }
}

// Open serial port for reading and writing
fd = open("/dev/ttyS1", O_RDWR | O_NOCTTY | O_NDELAY);
if(fd==-1)
    perror("Unable to open communications port");
else
    fcntl(fd, F_SETFL, FNDELAY);
tcgetattr(fd, &options);
cfsetispeed(&options, B115200);
cfsetospeed(&options, B115200);
options.c_cflag |= (CLOCAL | CREAD);
options.c_lflag &= ~(ICANON | ECHO | ECHOE | ECHONL | ISIG);
options.c_iflag = (IGNPAR | ISTRIP | ICRNL );
tcsetattr(fd, TCSANOW, &options);
tcgetattr(fd, &options);

// Initialize WCPC
write(fd, "SM,0\r",5); // Put WCPC in idle state (if its connected)
usleep(100000); // 100msec delay for WCPC response
while(read(fd,rec_buff,1)>0) // Flush any characters in serial buffer
;
write(fd, "RV\r",3); // Collect WCPC type
usleep(100000); // 100ms delay for WCPC response
buffptr = rec_buff;
rec_buff[0]=0;
while(read(fd,buffptr,1)>0){
    if(*buffptr == '\n')
        *buffptr = 0;
    else
        buffptr++;
}
if(strlen(rec_buff)>0)
    printf("Found %s\n",rec_buff);
else
    printf("WCPC not connected\n");

// Begin sampling now or at provided start time
now = time(NULL);
timestamp = localtime( &now );
if(argc==3){ // only start minute is given
    start_hour = timestamp->tm_hour;
}
if(argc>2){ // wait for starting minute and hour
    printf("Sample interval is %d.%d Starting time is: %d:%d\n",
        atoi(&argv[1][0])/10, atoi(&argv[1][0])%10, start_hour, start_min);
    do{
        usleep(50000); // wait for start time
    }
}

```

```

        now = time(NULL);
        timestamp = localtime( &now );
    }while(timestamp->tm_sec!=0 || timestamp->tm_min!=start_min || timestamp->tm_hour!=start_hour);
    printf("Sampling begun\n");
}
else
    printf("Sample interval is %d.%d Sampling has begun\n", atoi(&argv[1][0])/10,
atoi(&argv[1][0])%10);

// Data collection section
buffptr = rec_buff; // Init buffer pointer to start of receive buffer
write(fd, sm_str,strlen(sm_str)); // Start process by giving WCPC 'set mode command
(SM)

while(1){ // forever grab data and write out to a file
    usleep(100000); // pause 100ms before checking if any data has
arrived
    do { // do stuff if data is in buffer
        nbytes = read(fd, buffptr, 1);
        if( nbytes > 0 ) { // grab a character from serial port
            if( *buffptr != '\n' ){ // here we just collect characters till end-of-line
                if(*buffptr == ','){
                    *buffptr = ' '; // change commas to spaces
                }
                buffptr++; // prepare for next character
            }
        } else{ // if CR then process the record
            *buffptr = '\0' ; // we want to stop at the newline
            if(strncmp(rec_buff,"OK",2)==0) { // drop the 'OK from the SM command at the beginning
                buffptr =rec_buff;
                break;
            }

            // use time stamp and do some pretty formatting
            sprintf( date, "%d", timestamp->tm_year+1900 );
            strcpy( filename, date );
            strcat( date, "/" );
            if( timestamp->tm_mon+1 < 10 ){
                strcat( date, "0" );
                strcat( filename, "0" );
            }
            sprintf( buff2, "%d",timestamp->tm_mon+1 );
            strcat( date,buff2 );
            strcat( filename,buff2 );
            strcat( date,"/" );
            if( timestamp->tm_mday < 10 ) {
                strcat( date,"0" );
                strcat( filename,"0" );
            }
            sprintf( buff2,"%d",timestamp->tm_mday );
            strcat( date,buff2 );
            strcat( filename,buff2 ); // file name is just date in YYYYMMDD format

            system( "mount -o remount -o rw /" ); // make file system RW so we can write data

            // open a file with the date as its name for appending
            filestream = fopen( filename, "a" );
            strcpy( send_buff, date ); // copy date time stamp into working buffer
            strcat( send_buff, " " );
            sprintf( buff2, "%i", timestamp->tm_hour );
            strcat( send_buff, buff2); // add hour
            strcat( send_buff, ":" );
            sprintf( buff2, "%i", timestamp->tm_min );
            strcat( send_buff, buff2); // and minute
            strcat( send_buff, ":" );
            sprintf( buff2, "%i", timestamp->tm_sec );
            strcat( send_buff, buff2); // and second
            strcat( send_buff, " " );
            strcat( send_buff, rec_buff ); // append WCPC data record (everything space
delimited)
            strcat( send_buff, "\n" ); // append end of line
            printf("%s",send_buff); // echo to console (if there is one)

            // write out the data in the send_buff

```

```
        fwrite(send_buff, 1, strlen(send_buff), filestream);
        fclose( filestream );           // close file

        system("mount -o remount -o ro /"); // make file system read-only in case of power
outage
        now = time(NULL);                // grab current time for next timestamp
        timestamp = localtime( &now );  // load time into structure
        buffptr = rec_buff;              // reset rec_buff pointer for next record
    } // end processing a record
}
} while( nbytes > 0 );                  // end of character processing loop
} // end of forever loop
} // end of program
```



**TSI Incorporated**

**UK** Tel: +44 1494 459200 **E-mail:** [tsiuk@tsi.com](mailto:tsiuk@tsi.com) **France** Tel: +33 491 95 21 90 **E-mail:** [tsifrance@tsi.com](mailto:tsifrance@tsi.com)  
**Germany** Tel: +49 241 523030 **E-mail:** [tsiqmbh@tsi.com](mailto:tsiqmbh@tsi.com) **Sweden** Tel: +46 8 595 13230 **E-mail:** [tsiab@tsi.com](mailto:tsiab@tsi.com)

