ESL-TR-01/11-01

INTERNET POLLING DEVELOPMENT REPORT

LoanSTAR deliverable report

Final Report

Peter Klima
Dan Lockhart
Jeff S. Haberl, Ph.D., P.E.

November 2001

PREFACE

This is one of a series of final reports which document the development of the LoanSTAR and Technical Assistance Development Deliverables. The developments are broken down into two divisions, Task C and Task D. The following two tables itemize the deliverables for each Task.

Task C Deliverables	Description
1.Access and Display LoanSTAR Data via the Internet for Interested Agencies	Energy Systems Lab's web-based interface to the LoanSTAR Energy Databases, referred to as the Energy and Environmental Data System (EEDS).
2.Report Print/ Viewing Options (i.e., PDF format)	Effort to move reporting to Adobe Inc.'s Portable Document Format.
3.Online Feedback Ability (automated email to ESL)	Online feedback form built into the ESL's EEDS.
4.Internet Data Logging and Display a.Develop Internet-Based Real-time Display Software b.Buy/Develop and Test a Date Logger with Integrated TCP/IP Connectivity c.Buy/ Develop and Test Software to Poll a TCP/IP Logger	Research and development of internet based power measurement and energy data logging techniques.

Task D Deliverables	Description
1.Automated Email Polled Data Problem Alerts. (ESL internal)	The automatic notification via email of the sites that failed polling for a given polling job.
2.Automate IPNs Review Process to an 'Exception Only' Basis (ESL internal)	Streamlining the IPN review process by improving problem tracking and reporting.
3.Convert Ipns plots to PDF format (ESL internal)	Effort to move IPN reporting to Adobe Inc.'s Portable Document Format.
4.Automated the Data Analysis/Quality Verification Process to 'Exception Only' (ESL internal)	A series of checks that monitor raw files loaded into the database for low level data quality analysis.
5.Move IPNs, MECRs, AECRs to an Internet Base	Effort to combine reporting in Adobe Inc.'s Portable Document Format with a secure Web-based interface.

EXECUTIVE SUMMARY

The papers included in this report represent the LoanSTAR and Technical Assistance Development Deliverable Task C-Number 4, "Internet Data Logging and Display". The paper included in this report represent the work of several individuals at Texas A&M University's Energy Systems Lab: Peter Klima, Dan Lockhart and Jeff S. Haberl, Ph.D., P.E.

ABSTRACT

This is the final report which documents the development of Internet-based data logger polling. The project consists of two main tasks: the development of automated polling procedures that can be launched remotely with no operator input, and the development of a polling engine to launch the above procedures and pass the data on to the database server.

This final report describes the automated polling procedures that have been developed for Synergistic, Highland and ABB loggers. This report also describes the universal polling engine, which, in addition to launching polling and transferring data, is able to convert ABB data into Synergistic format (the format required to load the data into the LoanSTAR Informix database). The results of testing with Synergistic, Highland and ABB loggers are also described, as well as the real-world production use with ABB power meter loggers at West Texas A&M University.

This final report presents the system design, testing results and real-world production usage of the system. The copyrighted source code of all the programs is also included.

ACKNOWLEDGEMENTS

This project would not have been possible without the support that was provided by the following persons and/or the agencies or companies for which they work: Ms. Theresa Sifuentes and Mr. Dub Taylor at the Texas State Energy Conservation Office, Mr. Mike Thorpe at ABB USA, Mr. Ron Gumina at Synergistic Control Systems, Mr. John Larkin at Highland Technologies.

This project would not have been possible without the support that was provided by the following persons that work at the Energy Systems Lab: Mr. Carlos Ortiz, Ms. Kim Carlson, Mr. Kelly Milligan, Mr. John Bryant, and Mr. Stephen O'Neal.

TABLE OF CONTENTS

1.0 INTRODUCTION

The ESL has developed and tested an automated Internet polling capability for Synergistic (Synergistic 2001), Highland (Highland 2001) and ABB (ABB 2001) data loggers. These procedures can be used to eliminate phone lines at those facilities where Internet access is available at the data logger, as well as reduce the man-hours required for polling. Two ABB A1R+ power meters/loggers are currently being polled at West Texas A&M University using the system described below.

2.0 SYSTEM DESIGN

The entire polling system was designed to allow data from the loggers (which are only equipped with RS-232 serial or modem connections when supplied by the manufacturers) to be polled across and Internet connection.

2.1 Polling hardware

As none of the logger types supports a TCP/IP connection, the loggers cannot be connected directly to the Internet. These loggers only support RS-232 and telephone connections. Therefore, for Internet polling to avoid telephone costs, the logger must be connected to a PC by an RS-232 serial connection. That PC, in turn, can be connected to the Internet. This computer is henceforth referred to as the polling server. A second PC, located at the ESL, is required to control the polling server, download data, and place it in the ESL database. That computer is henceforth referred to as the polling client. This configuration is shown in Figure 1.

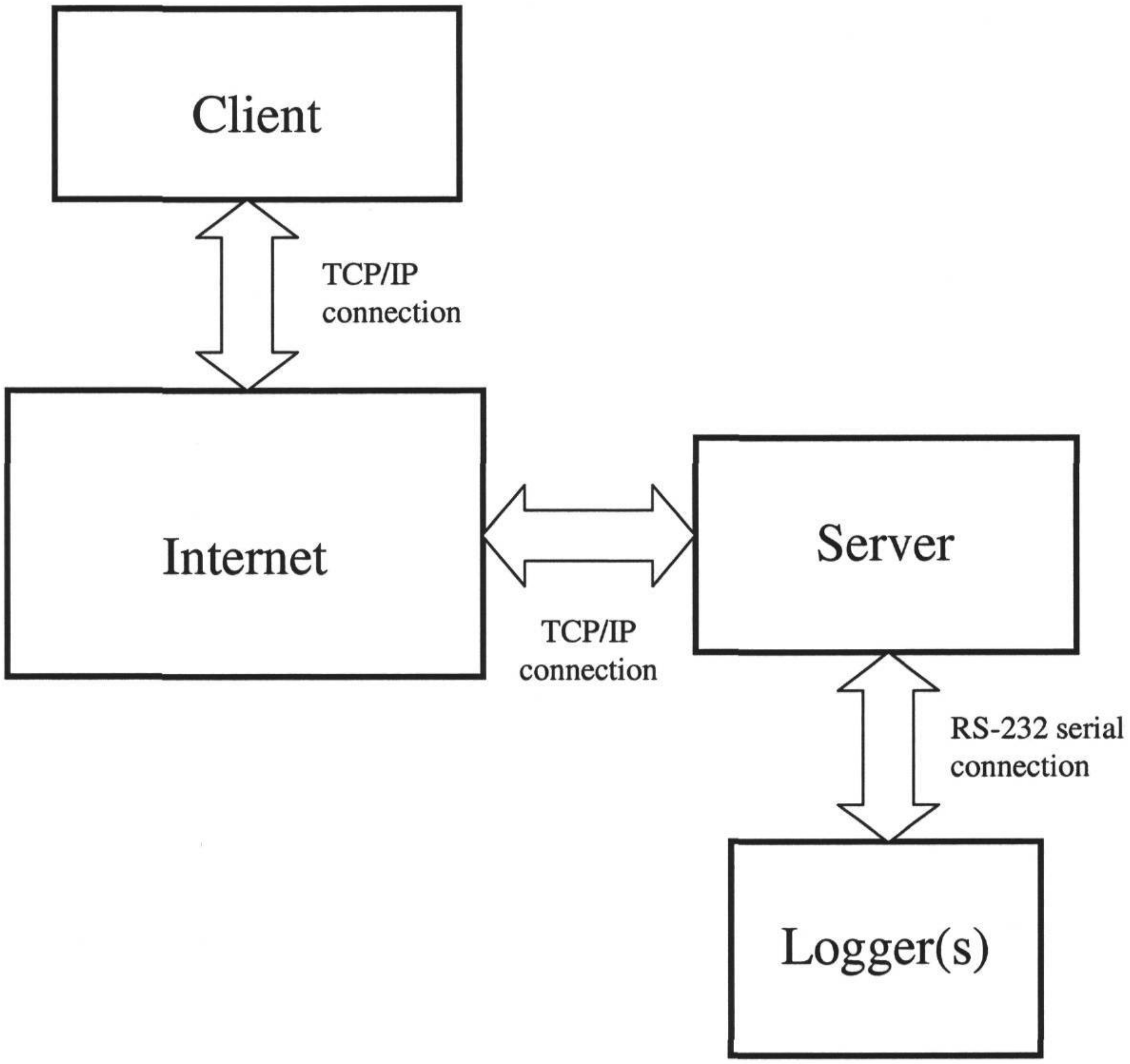

Figure 1. Polling hardware configuration. This figure is a diagram of the equipment used in this demonstration project.

2.2 Polling software

Rather than develop new software for communicating with the loggers, the ESL chose to use the software provided by the data logger manufacturers. This was done for reasons of support and reliability. However, the manufacturers' software requires user input to read the data from the logger, and such input cannot be manually provided at the remote server side.

2.2.1 Software control method and scripts

As the software for all three logger types being tested is not true Microsoft Windows software and runs in a MS-DOS window, it cannot be remotely controlled using MS Windows API calls. This problem was solved by using a keystroke simulation program to automatically provide predetermined input sequences to the MS-DOS programs. After testing several freeware keystroke simulation programs, Scancode 5.90 (Johnson 2000) was selected for its reliability and its ability to react to text appearing on the screen. Waiting for specific text to appear on the screen to indicate when an action is finished is more reliable and faster than using simple delays to give the software time to finish an action before feeding more keystrokes. It also allows special keystroke sequences to be activated in the case of action failure, thus allowing the program to exit and place an error message in a log file. The Scancode sequences are activated by MS-DOS batch files. Different batch files are required for each logger. Examples for all logger types can be found in the Appendix.

2.2.2 Server-side polling engine software

The batch files are run by the polling server program (originally written by Dan Lockhart and included in the Appendix), which is started by Windows when the server PC boots. The server resides in a perpetual 'waiting to connect' state which is changed to 'connected' when a 'start polling' command is received from the client. The server is divided into two primary functions. One function communicates with the client. The other function activates the batch files that poll the loggers. When polling is completed for a logger, a 'polling result status message' is sent to the client. If all sites have been polled, the server 'disconnects' from the client and re-enters the 'waiting to connect' state. The server also records its activity in a status file. All polled data is placed in a repository file on the ESL network's common share for reformatting (not needed for Synergistic logger files) and transfer to the ESL database loading routine.

2.2.3 Client-side polling engine software

The client (also written by Dan Lockhart and included in the Appendix) is started by the Windows NT Scheduler when polling is to commence. When started, the client verifies that the last polling activity was completed without errors. It then contacts the server program which runs on the Logger Polling PC (Fig. 1). After connection is established, the client sends a 'start polling' command, which include the sites to be polled, to the server(s) and waits for the arrival of a polling result message. This activity continues until all sites to be polled by the server have been polled as directed by the client. All communication between client and server is written to a status file on the client PC for later analysis if necessary. The client is capable of providing realtime polling status information, though this capability is still being developed. When all sites have been polled the client activates a data reformatting program for any non-Synergistic loggers (the program for ABB loggers written by Dan Lockhart is included in the Appendix) and terminates. The reformatting program converts the polled data to Synergistic format, and the converted file is stored in a repository file on the network common share by the polling client, into the format required by the ESL database load routine. The data are then sent to the UNIX database system using the FTP feature of the Internet Transfer ActiveX Control. After the FTP process completes, the reformatting program invokes the MAPI ActiveX Controls to send an email message to the database administrator informing about the load data transfer. The program then terminates.

2.3 System overview

The programs, the computer on which they run and their relationship to each other are shown in Figure 2. Although the detailed relationships are quite complex, the entire system can be automated as long as the task scheduler is configured to run the polling client program at regular intervals, and the polling server is launched on the polling computer by default (i.e. it is placed in the startup program group). These programs will then launch and control the others.

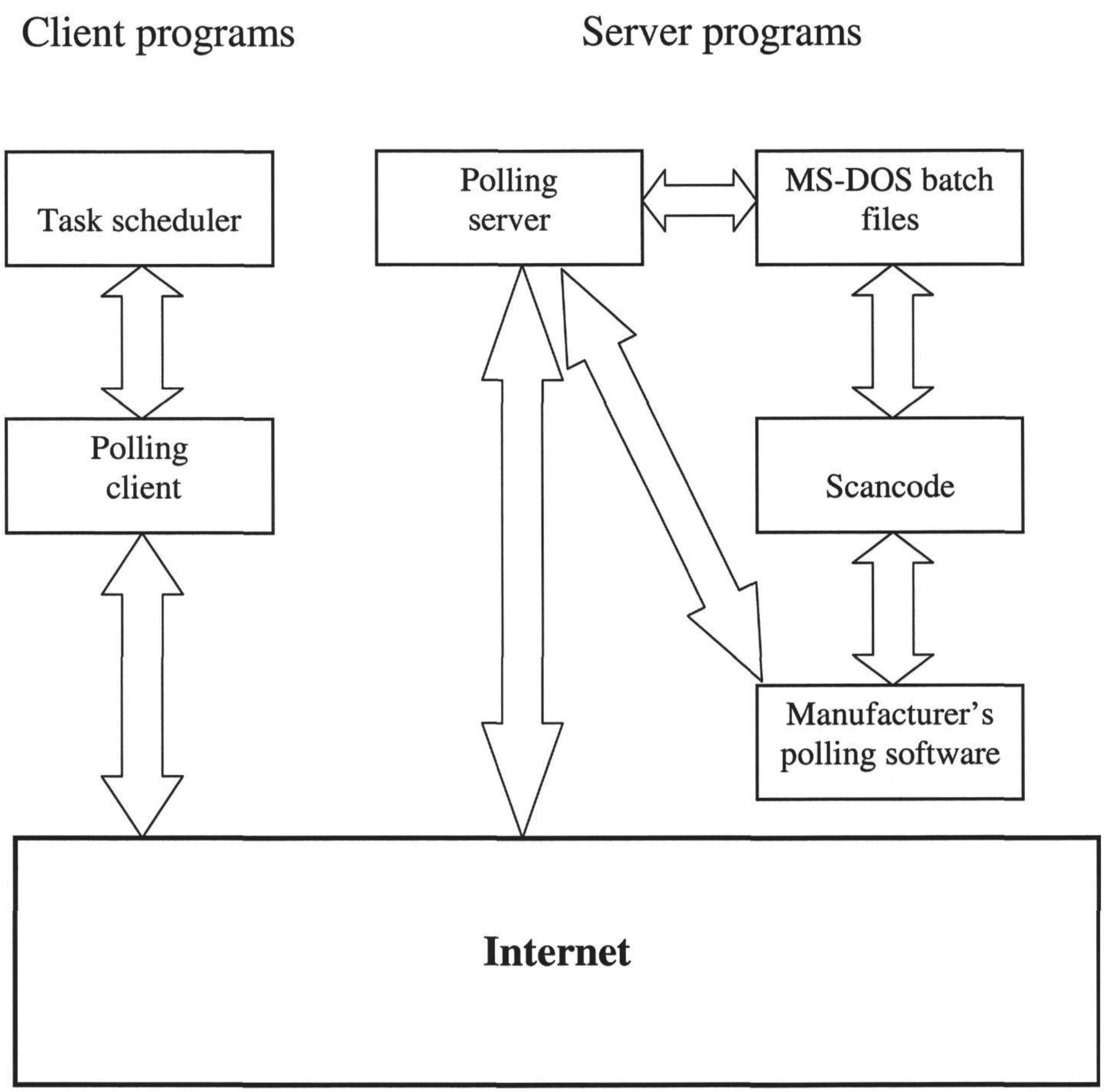

Figure 2. Polling programs and their relationships

3.0 TESTING RESULTS

The above system has been successfully tested with various models of directly connected Synergistic loggers (with all versions of Parset in use at ESL), various models of Synergistic loggers connected via phone line, and two ABB A1R+ power meter/loggers connected via phone line (no ABB logger was locally available for testing, so the ESL's loggers at remote sites were used). Automated keystroke feeding was also tested with one logger (a directly connected Highland logger), although the data format conversion program for those loggers has not yet been fully developed.

Two ABB loggers are currently being polled automatically on a weekly basis by the ESL using the above system. Although this connection currently uses a telephone line, the system is used to automate polling and save time. This demonstrates the dependability of the system in a real-world production setting. However, it has not been tested for polling large quantities of loggers with a single server, nor for using multiple servers with one client.

4.0 SUMMARY

Automated Internet polling capabilty has been developed for two logger types, and partially developed for a third. The basic principles may be applicable to several other logger types as well. It has also been demonstrated to be a reliable system for production use.

5.0 REFERENCES

ABB USA. (2001). P.O. Box 5308, 501 Merritt 7, Norwalk CT 06856-5308, USA

Highland Technologies, Inc. (2001). 320 Judah Street, San Francisco, CA 94122, USA.

Johnson, Bret (2000). Scancode 5.90 users guide. File provided with the freeware program available from the Internet at http://members.aol.com/bretjohn.

Synergistic Control Systems, Inc. (2001). 2700 Lake Villa Drive, Suite 180, Metairie, LA 70002, USA.

APPENDIX A: POLLING SCRIPT BATCH FILES

A.1 Scripts for Highland loggers (Ktools program)

Two batch files are required to complete polling. The first is the main polling sequence and must be altered appropriately for each individual logger. The second file only launches the first file and after that exist, updates the status file to indicate the main sequence is finished.

A.1.1 ktools.bat

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem this batch file causes kt to automatically poll whatever TSRs haven't
rem been polled yet
c:
cd \ktools
rem
rem uninstall any previous scancodes, just in case
scancode u
rem
rem change the status file to indicate that polling is now active
echo Status: Polling >status.txt
rem
rem here come the scancodes
rem
rem this one runs ktools2.bat which runs kt, then feeds kt keystrokes which
rem open a logger, connect to it, and go to read TSRs
rem this selects the second logger on the list (which our test logger happens
rem to be) - in order to select the first logger, remove one GDn (visible
rem below), or add as many GDns as needed to select a logger further down
rem on the list
rem at the end, it presses the F1 and F2 keys, triggering the next scancodes
scancode "ktools2" 28,d 4, Alt-f Alt-24 GDn GDn GDn 28, d 1, Alt-24, d 1, 1 Alt-c Alt-24, d 39,
1, d 1, 1 Alt-24 Alt-t f1 d2 f2
rem
rem this one just moves the mouse cursor down - a dumb workaround to get to
rem the button to read TSRs
rem it will not work correctly if NumLock isn't on, but it will not work
rem with the grey arrow keys, period
scancode k f1 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2
Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2
rem
rem this one is more of the same - the movement needed is so far that one
rem line isn't enough
scancode k f2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2
Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2 Sft-N2
rem
rem after waiting long enough to ensure the mouse cursor has been moved,
rem press shift-enter to simulate a mouse click, causing TSRs to begin
rem loading
scancode d 64 Sft-Enter
rem
rem scancode can't read kt's screen, because it's in graphics mode, so we
rem have to wait till the screen interrupt doesn't do anything for a while
rem another silly workaround
scancode waitforidle 72, Sft-N2 28
rem
rem two really dumb workarounds in one - wait for things to be done, then
rem start moving the mouse cursor to the right
scancode waitforidle 74, Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-
N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6
rem
rem move the mouse cursor some more three seconds later
scancode waitforidle 77, Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6 Sft-
N6 Sft-N6 Sft-N6 Sft-N6 Sft-N6
rem
rem after the mouse cursor is where it should be (over the Cancel button),
rem simulate a mouse click and exit the program
scancode waitforidle 80, Sft-Enter d 1, 1 alt-f alt-x
rem
rem this program doesn't have to change the status.txt file to say it's done,
rem uninstall all those scancodes or exit the DOS window, since ktools2.bat
rem takes care of those things
```

A.1.2 ktools2.bat

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem batch file called by ktools.bat as a kludge to get scancode to feed
rem keystrokes to kt but still have the batch file do stuff afterwards
kt
echo Status: Polling Complete >status.txt
rem exit
```

A.2 Scripts for Synergistic loggers (Parset program)

Three batch files are used. One runs the main polling sequence, and then (depending on whether polling succeeded or failed, as determined by Scancode reading information from the screen) one of the others is launched to update the status file accordingly. The main polling sequence file is slightly different for each version of Parset, and must be altered appropriately to work with each individual logger.

A.2.1 Main file for Parset 1.06

```
@echo off
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem automatically poll using Scancode to feed Parset keystrokes
rem will poll fixed TSRs, so works best for polling the whole logger
c:
cd \sy\101
rem change the status file
echo polling >c:\sy\status.txt
rem
rem now, the big scancode keystroke feeder
rem it calls test2.bat, which runs Parset, to which all these keystrokes
rem are fed
rem when Parset exits (as instructed to by scancode at the end), test2 will
rem write DONE to the status file and close the DOS window
rem this part is run last, it exits Parset after polling
scancode k "x" w 0,0 "Any Key" " " 1 1 "q" d 1 "compl" 28
rem this is the backup exit strategy, if polling fails and "Any Key" never
rem shows up
scancode i 1900 " " 1 1 1 1 "q" d 1 "fail" 28
rem this is the polling sequence, run second
scancode w 0,0 "Any Key", " t" 56 56 "c", d 6, "0" 28 "12" 28 "f" 28 "temp.raw" 28 28 28 "x"
rem                                                     ^^           ^^^^^^^
rem                                                 last TSR         filename
rem this part calls test2.bat (which calls parset) and feeds the keystrokes
rem to connect to the logger
scancode "test2" 28, s 1, i 4, 28 28 GDn GDn GDn 28 "cp" 28 28 28
echo DONE
rem                       ^^^
rem                   no GDn in this space - select fist logger
rem
rem the above parts will need to be changed for different loggers
rem
rem to select a different logger, add or remove down arrows (GDns) in the
rem last scancode
rem
rem the number of TSRs depends on how many the logger can hold (full test
rem logger is 14335)
rem
rem the filename is the logger serial number plus a .raw extension
rem
rem for the time being, only polling 12 TSRs and filename is bla.raw
rem
rem
rem test2.bat will also change the status.txt file after Parset exits
rem and close the DOS window, thus unloading itself from memory as well
rem so, no need to have this batch file do any of that
```

A.2.2 Main file for Parset 1.48

```
@echo off
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem automatically poll using Scancode to feed Parset keystrokes
rem will poll fixed TSRs, so works best for polling the whole logger
c:
cd \sy\491
del temp.raw
rem change the status file
echo polling >c:\sy\status.txt
rem
rem now, the big scancode keystroke feeder
rem it calls test2.bat, which runs Parset, to which all these keystrokes
rem are fed
rem when Parset exits (as instructed to by scancode at the end), test2 will
rem write DONE to the status file and close the DOS window
rem this part is run last, it exits Parset after polling
scancode k "x" w 0,0 "Any Key" " " 1 1 "q" d 1 "compl" 28
rem this is the backup exit strategy, if polling fails and "Any Key" never
rem shows up
scancode i 1900 " " 1 1 1 1 "q" d 1 "fail" 28
rem this is the polling sequence, run second
scancode w 0,0 "Any Key", " t", d 2, "c", d 6, "f" 28 "0" 28 "2660" 28 "temp.raw" 28 28 28 "x"
rem                                       ^^     ^^^^^^^
rem                                    last TSR    filename
rem this part calls test2.bat (which calls parset) and feeds the keystrokes
rem to connect to the logger
scancode "test2" 28, s 1, i 4, 28 28 GDn GDn GDn 28 "cp" 28 28 28
echo DONE
rem                    ^^^
rem                    no GDn in this space - select fist logger
rem
rem the above parts will need to be changed for different loggers
rem
rem to select a different logger, add or remove down arrows (GDns) in the
rem last scancode
rem
rem the number of TSRs depends on how many the logger can hold (full test
rem logger is 14335)
rem
rem the filename is the logger serial number plus a .raw extension
rem
rem for the time being, only polling 12 TSRs and filename is bla.raw
rem
rem
rem test2.bat will also change the status.txt file after Parset exits
rem and close the DOS window, thus unloading itself from memory as well
rem so, no need to have this batch file do any of that
```

A.2.3 Main file for Parset 1.5

```
@echo off
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem automatically poll using Scancode to feed Parset keystrokes
rem will poll fixed TSRs, so works best for polling the whole logger
c:
cd \sy\h
rem change the status file
echo polling >c:\sy\status.txt
rem delete old raw file to make sure Parset won't complain about wanting to
rem overwrite it - in the final version, should probably not just destroy it
rem
rem now, the big scancode keystroke feeder
rem it calls test2.bat, which runs Parset, to which all these keystrokes
rem are fed
rem when Parset exits (as instructed to by scancode at the end), test2 will
rem write DONE to the status file and close the DOS window
rem this part is run last, it exits Parset after polling
scancode k "x" w 0,0 "Any Key" " " 1 1 "q" d 1 "compl" 28
rem this is the backup exit strategy, if polling fails and "Any Key" never
rem shows up
```

```
scancode i 1900 " " 1 1 1 1 "q" d 1 "fail" 28
rem this is the polling sequence, run second
scancode w 0,0 "Any Key", " t" d 6 "c", d 6, "f" 28 "0" 28 "12" 28 "temp.raw" 28 28 28 "x"
rem                                              ^^      ^^^^^^^
rem                                          last TSR    filename
rem this part calls test2.bat (which calls parset) and feeds the keystrokes
rem to connect to the logger
scancode "test2" 28, s 1, i 4, 28 28 28 "cp" 28 28 28
echo DONE
rem                     ^^^
rem             no GDn - select first logger
rem
rem the above parts will need to be changed for different loggers
rem
rem to select a different logger, add down arrows (GDn) before the third 28
rem (the fourth 28, if you count the one right after "parset", too...)
rem
rem the number of TSRs depends on how many the logger can hold (full test
rem logger is 14335)
rem
rem the filename is the logger serial number plus a .raw extension
rem
rem
rem test2.bat will also change the status.txt file after Parset exits
rem and close the DOS window, thus unloading itself from memory as well
rem so, no need to have this batch file do any of that
```

A.2.4 Script launched in case of success

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
echo Complete > c:\sy\status.txt
exit
```

A.2.5 Script launched in case of failure

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
echo Failed > c:\sy\status.txt
exit
```

A.3 Scripts for ABB loggers (AlphaPlus program)

Four batch files are used. The first is the main polling sequence, which must be altered appropriately for each logger. If polling fails (as determined by Scancode reading the information on the screen), the failure program (which must also be altered appropriately for each logger – although not strictly necessary for the system to run appropriately, it is useful if the status file contains specific information as to which logger failed) is launched. In case of success, the report-generating program is launched, and it in turn launches the finishing program, which updates the status file.

A.3.1 Main polling sequence

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem automated polling script for ABB meter
rem this version will select the first meter in the second phone list
rem which ends up being the WTAMU phone list and the Activity Center meter
rem the script calls rpt.bat which generates a report file and in turn calls
rem finish.bat which backs up the data
rem
rem first, make sure there's no files called billread.rp3 or a.rp3 to get
rem in the way of the later parts of the script
c:
echo Activity Center > c:\aplus\status.txt
cd \aplus\billing
copy old.rp3 /b + billread.rp3 /b + a.rp3 /b temp.rp3 /b
copy temp.rp3 /b old.rp3 /b
del billread.rp3
del a.rp3
cd ..
rem
rem clear out any previously loaded scancodes
scancode u
rem
rem these keystrokes are actually run at the end, after the data is polled
rem (after the scancode sequence below is executed)
rem the 56s (Alt key) are a kludge to delay the 1s (Esc key)
rem the rpt launches the next batch file
scancode d 800 1 56 1 56 1 56 1 56 1 56 1 56 1 56 1 "y" d 1 "faila" 28
scancode w 24, 0 "1 " 1 56 1 56 1 alt-35 1 1 "y" "rpt" 28
rem
rem this is the sequence for polling
rem in order to select a different meter, GDns below need to be removed
rem or added
rem the first GDn is to select the second phone list
rem in order to select another meter on the list, add GDns after the "c"
rem no GDns means the first meter gets selected
rem other than that, everything remains constant for all meters
scancode "aplus" 28 "alpha" 28 28 88 GDn 28 "c" 28 w 0,0 "CONN" 28 w 3,19 "Register R"
GryRight GDn 28 28 28
```

A.3.2 Script launched in case of polling failure

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem This is run when AUTOA.BAT fails to poll and times out
echo failed Activity Center > c:\aplus\status.txt
scancode "exit" 28
```

A.3.3 Report generation script

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem automated billing data report generation script for ABB meters
rem this is run by the auto.bat polling script, and at the end launches the
rem finish.bat script, which backs up the data
rem
rem first, make sure the newly collected data will be at the beginning of
rem the list of data that can be reported on
rem as the list is in alphabetical order by file, we copy the data to a
rem file called a.rp3
c:
cd \aplus
ren billing\billread.rp3 billing\a.rp3
rem
rem unload old leftover scancodes and key in the report-generating sequence
rem then call finish.bat
rem the file will be called temp.rpt - obviously, change the "temp" part
rem to generate a file with a different name
scancode u
scancode "aplus" 28 "alpha" 28 28 GryRight 28 GryRight GryRight 28 GDn GDn 28 28 Gdn 28
"temp" 28 28 f10 1 1 1 1 1 "y" "finish" 28
```

A.3.4 Backup and cleanup script launched after report generation

```
rem Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
rem this script is called by rpt.bat (which is called by auto_.bat) and backs
rem up the freshly polled data
c:
cd \aplus\billing
copy a.rp3 /b + old.rp3 /b temp.rp3 /b
copy temp.rp3 old.rp3 /b
```

```
del a.rp3
del temp.rp3
echo Complete > c:\aplus\status.txt
exit
```

APPENDIX B: Server-side polling engine software

```
'Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
Option Explicit
Dim timerctr As Integer
Dim scnt As Integer
Dim activ As Boolean
Dim connected As Boolean
Dim subst As Boolean
Dim toact As Boolean
Dim map As Boolean

Private Sub Command1_Click()
 If Not (tcpServer.State = sckClosed) Then
   tcpServer.Close
   Do Until tcpServer.State = sckClosed
   Loop
 End If
 statusck
 DoEvents
 tcpServer.Listen
 statusck
 DoEvents
End Sub

 Private Sub Form_Load()
 On Error GoTo errend
 map = True
 Open "c:\map.txt" For Output As #8
 Print #8, "start"
 Close #8

 statusck
 DoEvents

 Open "c:\check.txt" For Output As #4
 Print #4, "init"
 Close #4
 If map Then
   mapper ("Closing Port 1011.")
 End If
 If Not (tcpServer.State = sckClosed) Then
   tcpServer.Close
   Do Until tcpServer.State = sckClosed
   Loop
   statusck
   DoEvents
  End If
  If map Then
   mapper ("Port 1011 closed.")
  End If
  tcpServer.LocalPort = 1011
  tcpServer.Listen
  statusck
  DoEvents
  If map Then
   mapper ("listen #1")
  End If
  Exit Sub
  On Error GoTo 0
errend:
  If map Then
   mapper ("Form_Load Error")
  End If
  Close
  End
 End Sub


Private Sub tcpServer_Close()
On Error GoTo errend
  If Not (tcpServer.State = sckClosed) Then
    tcpServer.Close
    Do Until tcpServer.State = sckClosed
    Loop
    statusck
    DoEvents
  End If
  Open "c:\check.txt" For Output As #4
  Print #4, "init"
  Close #4
  tcpServer.LocalPort = 1011
  tcpServer.Listen
  statusck
  DoEvents
  If map Then
   mapper ("listen #2")
  End If
   Exit Sub
  On Error GoTo 0
errend:
 If map Then
   mapper ("tcpServer_Close Error")
 End If
 Close
 End
End Sub

Private Sub tcpServer_Connect()
On Error GoTo errend
  connected = True
  If map Then
   mapper ("connected")
 End If
   statusck
 DoEvents
   Exit Sub
 On Error GoTo 0
errend:
 If map Then
   mapper ("tcpServer_Connect Error")
 End If
 Close
 End
End Sub

Private Sub tcpServer_ConnectionRequest _
(ByVal requestID As Long)
   'Check if the control's State is closed. If not,
'close the connection before accepting the new
'connection.
On Error GoTo errend
 statusck
 DoEvents
 If Not (tcpServer.State = sckClosed) Then
   tcpServer.Close
   Do Until tcpServer.State = sckClosed
   Loop
     statusck
 DoEvents
 End If
'Accept the request with the requestID
'parameter.
 tcpServer.Accept requestID
  statusck
 DoEvents
 If map Then
```

```
   mapper ("connection accepted")
 End If
   Exit Sub
  On Error GoTo 0
errend:
  If map Then
   mapper ("tcpServer_ConnectionRequest Error")
  End If
  Close
  End
End Sub

Private Sub tcpServer_Error(ByVal Number As Integer, Description As String, ByVal Scode As
Long, ByVal Source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
 On Error GoTo errend
 mapper Str(Number) + "    " + Description

 If Not (tcpServer.State = sckClosed) Then
   tcpServer.Close
   Do Until tcpServer.State = sckClosed
   Loop
     statusck
  DoEvents
 End If
 connected = False
 If map Then
   mapper ("error occurred")
 End If
    Exit Sub
  On Error GoTo 0
errend:
  If map Then
   mapper ("tcpServer_Error Error")
  End If
  Close
  End
End Sub

Private Sub tcpServer_DataArrival _
   (ByVal bytesTotal As Long)
   'Declare a variable for the incoming data.
   'Invoke the GetData method and set the Text
   'property of a TextBox named txtOutput to
   'the data.
   Dim strData As String
Dim h As Long

On Error GoTo errend
tcpServer.GetData strData

If Trim(strData) = "close" Then
    If map Then
      mapper ("Received Close - Closing.")
    End If
    Command1_Click
Else

   If Trim(strData) = "abb" Then
    If Shell("C:\command.com /k c:\aplus\ipserver.exe", vbMaximizedFocus) = 0 Then
       If map Then
         mapper ("ABB Server didn't launch.")
       End If
    Else
       If map Then
         mapper ("ABB Server launched.")
       End If
       tcpServer.SendData "abb"
       Timer1.Enabled = True
       Timer1.Interval = 60000
    End If
    Exit Sub
   End If

    If Trim(strData) = "syn" Then
     Open "c:\check.txt" For Output As #4
      Print #4, "init"
      Close #4
     If Shell("C:\command.com /k c:\sy\syner_server.exe", vbMaximizedFocus) = 0 Then
        If map Then
            mapper ("Synergistic Server didn't launch.")
        End If
     Else
        If map Then
            mapper ("Synergistic Server launched.")
        End If
        tcpServer.SendData "syn"
        Timer2.Enabled = True
        Timer2.Interval = 60000
     End If
     Exit Sub
    End If
```

```
        mapper "unknown command: " + Trim(strData)
 End If

   Exit Sub
 On Error GoTo 0
errend:
 If map Then
  mapper ("tcpServer_DataArrival Error")
 End If
 Close
 End
End Sub
Private Sub Timer1_Timer()
  Dim str1 As String

  On Error GoTo errend
  Timer1.Enabled = False
  'End
  Open "c:\check.txt" For Input As #4
  Line Input #4, str1
  Close #4
  If Mid(LCase(Trim(str1)), 1, 4) = "abbe" Then
     tcpServer.SendData "abbe"
     If map Then
       mapper ("ABB Server ended.")
     End If
     Open "c:\check.txt" For Output As #4
     Print #4, "init"
     Close #4
  Else
     Timer1.Enabled = True
     Timer1.Interval = 60000
  End If
  Exit Sub
 On Error GoTo 0
errend:
 If map Then
  mapper ("Timer1_Timer Error")
 End If
 Close
 End
 End Sub

Private Sub Timer2_Timer()
  Dim str1 As String
  On Error GoTo errend
  Timer2.Enabled = False

  Open "c:\check.txt" For Input As #4
  Line Input #4, str1
  Close #4
  If Mid(LCase(Trim(str1)), 1, 4) = "syne" Then
      tcpServer.SendData "syne"
      If map Then
        mapper ("Synergistic Logger ended.")
      End If
      Open "c:\check.txt" For Output As #4
      Print #4, "init"
      Close #4
      tcpServer.Close
      statusck
      DoEvents
  Else
      Timer2.Enabled = True
      Timer2.Interval = 60000
  End If
  Exit Sub
 On Error GoTo 0
errend:
 If map Then
   mapper ("Timer2_Timer Error")
 End If
 Close
 End
End Sub
Private Sub mapper(tag As String)
 Open "c:\map.txt" For Append As #8
 Print #8, tag
 Close #8
End Sub
Sub statusck()
Select Case tcpServer.State    ' Evaluate state.
Case 0
  Text1.Text = "Current State: Closed"
Case 1
  Text1.Text = "Current State: Open"
Case 2
  Text1.Text = "Current State: Listening"
Case 3
  Text1.Text = "Current State: Connection pending"
```

```
Case 4
  Text1.Text = "Current State: Resolving host"
Case 5
  Text1.Text = "Current State: Host resolved"
Case 6
  Text1.Text = "Current State: Connecting"
Case 7
  Text1.Text = "Current State: Connected"
Case 8
  Text1.Text = "Current State: Peer is closing the connection"
Case 9
  Text1.Text = "Current State: Error"
Case Else
  Text1.Text = "Unknown State"
End Select
End Sub
```

APPENDIX C: Client-side polling engine software

C.1 The polling client

```
'Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
Option Explicit
Public errcnt As Integer
Public gocnt As Integer

Private Sub Command1_Click()
   Timer5.Enabled = False
   tcpClient.SendData "close"
   status ("waiting to close")
End Sub

Private Sub Command2_Click()
  If Not (tcpClient.state = sckClosed) Then
   'MsgBox "Closing Connection"
   tcpClient.Close
   Do Until tcpClient.state = sckClosed
   Loop
  End If
  End
End Sub

Private Sub Command3_Click()
  tcpClient.SendData "syn"
  Timer5.Enabled = False
End Sub

Private Sub Command4_Click()
  Timer5.Enabled = False
  engine.Hide
  display.Show
End Sub

Private Sub Command5_Click()
   tcpClient.SendData "abb"
   Timer5.Enabled = False
End Sub

Private Sub Form_Load()
  'The name of the Winsock control is tcpClient.
  'Note: to specify a remote host, you can use
  'either the IP address (ex: "121.111.1.1") or
```

```
 'the computer's "friendly" name, as shown here.
 Timer3.Enabled = True
 Timer3.Interval = 3000
 Timer5.Enabled = False
 Timer5.Interval = 60000
 display.Timer1.Interval = 20000
 Open "\\eslnt\common\dlockhar\loggers\polling_results.txt" For Output As #1
 Print #1, Trim("start")
 Close #1
 errcnt = 0
 'If LCase(InputBox("Connect?", "Connect or Check Status", "y")) = "y" Then
    tcpClient.LocalPort = 1011
    tcpClient.RemoteHost = "PRINTSVR"
    tcpClient.RemotePort = 1011
    If Not (tcpClient.state = sckClosed) Then
     tcpClient.Close
     Do Until tcpClient.state = sckClosed
     Loop
    End If
    'MsgBox "connect"
    On Error Resume Next
    tcpClient.Connect
    On Error GoTo 0
 'End If
End Sub


Private Sub tcpClient_error(ByVal Number As Integer, Description As String, ByVal Scode As
Long, ByVal source As String, ByVal HelpFile As String, ByVal HelpContext As Long,
CancelDisplay As Boolean)
 errcnt = errcnt + 1
 MsgBox "ERROR: " + CStr(Number) + vbCrLf + vbCrLf + _
 "DESCRIPTION: " + Description
 If Not (tcpClient.state = sckClosed) Then
  tcpClient.Close
  status ("waiting to close - error")
  Do Until tcpClient.state = sckClosed
  Loop
  status ("closed - error")
 End If
 If errcnt > 3 Then
  End
 Else
  try_again
 End If
End Sub
```

```
Private Sub tcpClient_Connect()
   status ("connected")
   gocnt = 0
   Timer5.Enabled = True
End Sub
Private Sub tcpClient_Close()
  status ("entered the close exit")
  If Not (tcpClient.state = sckClosed) Then
    tcpClient.Close
    Do Until tcpClient.state = sckClosed
    Loop
  End If
  status ("connection closed - reconnecting")
  tcpClient.Connect
End Sub

Private Sub tcpClient_DataArrival _
  (ByVal bytesTotal As Long)
  Dim strData As String
  Dim str1 As String
  Dim i As Integer

  tcpClient.GetData strData
  str1 = Trim(strData)
  If LCase(Trim(str1)) = "abb" Then
     status ("ABB Logging Process started on " + str(Date) + " at " + str(Time))
     Timer1.Enabled = True
     Timer1.Interval = 60000
  Else
     If LCase(Trim(str1)) = "syn" Then
          status ("Synergistic Logging Process started on " + str(Date) + " at " + str(Time))
          Timer2.Enabled = True
          Timer2.Interval = 60000
     Else
          If LCase(Trim(str1)) = "abbe" Then
               status ("ABB Logging Process ended on " + str(Date) + " at " + str(Time))
               tcpClient.SendData "abb"
          Else
               If LCase(Trim(str1)) = "syne" Then
                 status ("Synergistic Logging Process ended on " + str(Date) + " at " + str(Time))
                 If Not (tcpClient.state = sckClosed) Then
                   tcpClient.Close
                   Do Until tcpClient.state = sckClosed
                   Loop
                 End If
```

```
            Else
             status ("spurious message: "' + str1 + "' ingnoring.")
            End If
        End If
    End If
  End If

End Sub
Private Sub status(state As String)
  Open "\\eslnt\common\dlockhar\loggers\polling_results.txt" For Append As #1
  Print #1, Trim(state)
  Close #1
End Sub

Private Sub Timer1_Timer()
   Dim h As Integer
   Timer1.Enabled = False
   If Shell("C:\vbtestdev\Poll ABB\connect_to_polling_PC.exe", vbMaximizedFocus) = 0 Then
     status ("ABB Client Cratered.")
   Else
     status ("ABB Client Started.")
   End If
End Sub

Private Sub Timer2_Timer()
   Dim h As Integer
   Timer2.Enabled = False
   If Shell("C:\vbtestdev\Poll Synergistic\Syner_Cient.exe", vbMaximizedFocus) = 0 Then
     status ("Synergistic Client Cratered.")
   Else
     status ("Synergistic Client Started.")
   End If
End Sub
Private Sub try_again()
  tcpClient.LocalPort = 1011
  tcpClient.RemoteHost = "PRINTSVR"
  tcpClient.RemotePort = 1011
  If Not (tcpClient.state = sckClosed) Then
    tcpClient.Close
    Do Until tcpClient.state = sckClosed
    Loop
  End If
  ''MsgBox "connect"
  On Error Resume Next
  tcpClient.Connect
  On Error GoTo 0
```

```
End Sub
Private Sub Timer3_Timer()
  statusck
  DoEvents
End Sub
Sub statusck()
Select Case tcpClient.state    'Evaluate state.
Case 0
  Text1.Text = "Current State: Closed"
Case 1
  Text1.Text = "Current State: Open"
Case 2
  Text1.Text = "Current State: Listening"
Case 3
  Text1.Text = "Current State: Connection pending"
Case 4
  Text1.Text = "Current State: Resolving host"
Case 5
  Text1.Text = "Current State: Host resolved"
Case 6
  Text1.Text = "Current State: Connecting"
Case 7
  Text1.Text = "Current State: Connected"
Case 8
  Text1.Text = "Current State: Peer is closing the connection"
Case 9
  Text1.Text = "Current State: Error"
Case Else
  Text1.Text = "Unknown State"
End Select
End Sub

Private Sub Timer5_Timer()
  If gocnt = 2 Then
      Timer5.Enabled = False
      tcpClient.SendData "abb"
  End If
  gocnt = gocnt + 1
End Sub
```

C.2 ABB report file format conversion program

```
'Copyright 2001 Texas Engineering Experiment Station, Energy Systems Laboratory
Public Sub abb()
  Dim rstsiteinfo1 As ADODB.Recordset
  Dim cnn As ADODB.Connection
Dim strcnn  As String
Dim server As String
Dim cnn1 As ADODB.Connection
Dim strcnn1  As String
Dim infile As String
Dim outfile As String
Dim thefile As String
Dim buffer1 As String
Dim unit1 As String
Dim unit2 As String
Dim outbuf As String
Dim holdbuf As String
Dim fso, MyFile, f
Dim i As Integer
Dim j As Integer
Dim k   As Integer
Dim ctr As Integer
Dim first As Boolean
Dim therest As Boolean
Dim startdate As String
Dim starttime As String
Dim startdatetime As String
Dim resumedatetime As String
Dim realdatetime As Date
Dim FileDateTime As Date
Dim cntrdatetime As Date
Dim time1 As Date
Dim time2 As Date
Dim time3 As Date
Dim time4 As Date
Dim dorf As Long
Dim valx(4) As String
Dim SENDFILES(20) As String
Dim val3 As String
Dim State As String
Dim logger As String
Dim chan As String
Dim val4 As String
Dim msg As String
Dim db As String
Dim place As String
Dim fs, a, f2, f1, s, fc, fd


Set cnn = New ADODB.Connection
Set cnn1 = New ADODB.Connection
```

```
  strcnn = "DSN=informix online connect;" & _

"UID=lvk4173;PWD=lvk4412;DATABASE=lsd;HOST=lstaraxp.tamu.edu;SERVER=lsol;SER
VICE=sqlexec;PROTOCOL=olsoctcp"
  cnn.Open strcnn
  Set rstsiteinfo1 = New ADODB.Recordset

  Call ftp_thing_on(" ", " ", "lstaraxp.tamu.edu", " ")

  Set fs = CreateObject("Scripting.FileSystemObject")
  Open "\\eslnt\common\dlockhar\loggers\abb\polling_status.txt" For Input As #1
  Line Input #1, val3
  Close #1


  If LCase(Trim(val3)) = "polling complete" Then
   'fs.deletefile "\\eslnt\common\dlockhar\loggers\abb\polling_status.txt", True
  Else
   status ("there was an error polling the ABB loggers.")
   End
  End If
  On Error Resume Next
  fs.deletefolder "\\eslnt\common\dlockhar\loggers\abb\data", True
  Set fd = fs.createfolder("\\eslnt\common\dlockhar\loggers\abb\data")
  On Error GoTo 0

  Set f = fs.GetFolder("\\eslnt\common\dlockhar\loggers\abb\")
  Set fc = f.Files

 '--------- START OF MAIN PROCESSING LOOP --------------------

For Each f1 In fc

   If LCase(Mid(Trim(f1.Name), 1, 3)) = "sub" Then
     logger = "436"
     chan = "4470"
     infile = "\\eslnt\common\dlockhar\loggers\abb\" + Trim(f1.Name)
     GoTo ext1
   End If
   If LCase(Mid(Trim(f1.Name), 1, 3)) = "act" Then
     logger = "437"
     chan = "4472"
     infile = "\\eslnt\common\dlockhar\loggers\abb\" + Trim(f1.Name)
     GoTo ext1
   End If
   GoTo ext2
```

```
ext1:

  logit ("processed " + Trim(f1.Name) + " on " + Str(Date) + " at " + Str(Time) + ".")

  scratch1 = "\\eslnt\common\dlockhar\scratch1.txt"
  scratch2 = "\\eslnt\common\dlockhar\scratch2.txt"
  thefile = logger + Trim(julian(Str(Date) + " " + Str(Time))) + ".txt"
  outfile = "\\eslnt\common\dlockhar\loggers\abb\" + thefile
  resumefile = "\\eslnt\common\dlockhar\loggers\abb\resumetime" + logger + ".txt"
  Open infile For Input As #1
  Open scratch1 For Output As #3


  Do Until EOF(1)
    Line Input #1, buffer1
    If (Mid(Trim(buffer1), 15, 5) = "Total") Then
      startdate = Mid(Trim(buffer1), 5, 8)
      starttime = Mid(Trim(buffer1), 1, 5)
      Line Input #1, buffer1
      starttime = Mid(Trim(buffer1), 1, 5)
      startdatetime = startdate + " " + starttime
      realdatetime = DateAdd("n", -15, startdatetime)
      Exit Do
    End If
  Loop
  Close #1

  ''Debug.Print "know start date"
  Open infile For Input As #1
  first = False
  therest = True
  Do Until EOF(1)
    Line Input #1, buffer1
    If (Mid(Trim(buffer1), 1, 21) = "-------- LOAD PROFILE") And (therest) Then
      first = True
      therest = False
      buffer1 = StrReverse(Trim(buffer1))
      i = InStr(1, buffer1, " ")
      i = InStr(i + 1, buffer1, "-")
      buffer1 = Mid(buffer1, i + 1, 3)
      unit1 = StrReverse(buffer1)
      Print #3, "CHANNEL units= " + unit1 + "   STARTING DATE= " + startdate
    Else
      If first Then
        If (Mid(Trim(buffer1), 1, 12) = "------- LOAD") Then
          Exit Do
      Else
       If IsNumeric(Mid(Trim(buffer1), 1, 2)) And Mid(Trim(buffer1), 3, 1) = ":" Then
        Print #3, buffer1
       Else
       End If
      End If
     Else
     End If
    End If
   Loop
 'End
 ''Debug.Print "1 channel"
   buffer1 = StrReverse(Trim(buffer1))
   i = InStr(1, buffer1, " ")
   i = InStr(i + 1, buffer1, "-")
   buffer1 = Mid(buffer1, i + 1, 5)
   unit2 = StrReverse(buffer1)
   Print #3, "CHANNEL units= " + buffer1 + "   STARTING DATE= " + startdate


   Do Until EOF(1)
     Line Input #1, buffer1
     If first Then
      If (Mid(Trim(buffer1), 1, 12) = "------- LOAD") Then
       Exit Do
      Else
       If IsNumeric(Mid(Trim(buffer1), 1, 2)) And Mid(Trim(buffer1), 3, 1) = ":" Then
        Print #3, buffer1
       Else
       End If
      End If
     Else
     End If
   Loop
   Close


   Open scratch1 For Input As #3
   Open scratch2 For Output As #4
   ctr = 0
   cntrdatetime = DateAdd("n", 15, realdatetime)
   Line Input #3, buffer1
   Do Until EOF(3)
     Line Input #3, buffer1
     If (Mid(Trim(buffer1), 1, 7) = "CHANNEL") Then
      Exit Do
    Else
     buffer1 = Trim(buffer1)
     buffer1 = Trim(Mid(buffer1, 6))
     i = InStr(1, buffer1, " ")
     valx(1) = Trim(Mid(buffer1, 1, i))
     buffer1 = Trim(Mid(buffer1, i))
     i = InStr(1, buffer1, " ")
     valx(2) = Trim(Mid(buffer1, 1, i))
     buffer1 = Trim(Mid(buffer1, i))
     i = InStr(1, buffer1, " ")
     valx(3) = Trim(Mid(buffer1, 1, i))
     buffer1 = Trim(Mid(buffer1, i))
     valx(4) = Trim(buffer1)
     For i = 1 To 4
      ctr = ctr + 1
      valx(i) = Replace(valx(i), "*", " ")
      valx(i) = Replace(valx(i), "-", " ")
      If Len(Trim(valx(i))) = 0 Then
         valx(i) = "0"
      End If
      'Debug.Print Str(DateValue(cntrdatetime)) + " " + Str(TimeValue(cntrdatetime)) + " " +
Str(ctr) + " ""V    "" " + valx(i)
      dorf = CLng(valx(i))
      valx(i) = Trim(Str(dorf * 0.21))
      'Debug.Print Str(DateValue(cntrdatetime)) + " " + Str(TimeValue(cntrdatetime)) + " " +
Str(ctr) + " ""V    "" " + valx(i)
      'End
      cntrdatetime = DateAdd("n", 15, cntrdatetime)
      Print #4, Str(DateValue(cntrdatetime)) + " " + Str(TimeValue(cntrdatetime)) + " " +
Str(ctr) + " ""V    "" " + valx(i)
     Next i
    End If
  Loop
  Close #4
  cntrdatetime = DateAdd("n", 15, cntrdatetime)
  holdbuf = Str(DateValue(cntrdatetime)) + " " + Str(TimeValue(cntrdatetime))

  Open scratch2 For Input As #4
  Open outfile For Output As #2

  On Error GoTo quit1
  Do Until EOF(3) Or EOF(4)
    Line Input #3, buffer1
    If (Mid(Trim(buffer1), 1, 7) = "CHANNEL") Then
     Exit Do
    Else
```

```
        k = 4
        buffer1 = Trim(buffer1)
        buffer1 = Trim(Mid(buffer1, 6))
        i = InStr(1, buffer1, " ")
        valx(1) = Trim(Mid(buffer1, 1, i))
        buffer1 = Trim(Mid(buffer1, i))
        i = InStr(1, buffer1, " ")
        valx(2) = Trim(Mid(buffer1, 1, i))
        If Not IsNumeric(valx(2)) Then
          k = 1
          GoTo calc
        End If
        buffer1 = Trim(Mid(buffer1, i))
        i = InStr(1, buffer1, " ")
        valx(3) = Trim(Mid(buffer1, 1, i))
        If Not IsNumeric(valx(3)) Then
          k = 2
          GoTo calc
        End If
        buffer1 = Trim(Mid(buffer1, i))
        valx(4) = Trim(buffer1)
        If Not IsNumeric(valx(4)) Then
          k = 3
        End If
calc:   For i = 1 To k
         Line Input #4, outbuf
         Print #2, Trim(outbuf) + " " + valx(i)
        Next i
      End If
  Loop

  On Error GoTo 0
  GoTo here

quit1: Close
  Resume here

here:

  Close
  Open outfile For Input As #2
  Open scratch2 For Output As #3
  Line Input #2, buffer1
  startdatetime = Mid(Trim(buffer1), 1, 17)
  time1 = startdatetime
  Close #2
  Open outfile For Input As #2
  Do Until EOF(2)
   Line Input #2, outbuf
   time2 = Mid(Trim(outbuf), 1, 17)
   If time1 <= time2 Then
    Print #3, outbuf
   End If
  Loop
  Close

  'adjust for daylight savings time
  time3 = "4/2/00 01:00:00"
  time4 = "10/29/00 00:00:00"
  Open scratch1 For Output As #3
  Open scratch2 For Input As #2
  Do Until EOF(2)
        Line Input #2, buffer1
        startdatetime = Mid(Trim(buffer1), 1, 17)
        buffer1 = Mid(Trim(buffer1), 18)                    'IF 2000-04-02 01:00 <=
TIMESTAMP <= 2000-10-29 00:00 then
        time1 = startdatetime                                  'TimeStamp = TimeStamp + 1
        If time1 >= time3 And time1 <= time4 Then         'Else
           time1 = DateAdd("h", 1, time1)                       'TimeStamp = TimeStamp
           startdatetime = Trim(Str(time1))
           If InStr(1, Trim(startdatetime), ":") = 0 Then
              startdatetime = startdatetime + " 00:00:00"
           End If
        End If
        buffer1 = startdatetime + " " + buffer1
        Print #3, buffer1
 Loop
 Close


  sqlquery = "select  * from chids where site=" + Trim(logger)
  On Error GoTo notgot
   rstsiteinfo1.Open sqlquery, cnn, , , adCmdText
  On Error GoTo 0
  db = Trim(rstsiteinfo1!server)
  rstsiteinfo1.Close
  cnn.Close


  If Trim(db) = "l" Then
   place = "lstar"
   strcnn1 = "DSN=informix online connect;" & _
```

```
"UID=lvk4173;PWD=lvk4412;DATABASE=lsd;HOST=lstaraxp.tamu.edu;SERVER=lsol;SER
VICE=sqlexec;PROTOCOL=olsoctcp;"
    cnn1.Open strcnn1
    Set rstsiteinfo1 = New ADODB.Recordset
  Else
    place = "esl"
    strcnn1 = "DSN=ESLSQL;" & _

"UID=lvk4173;PWD=lvk4412;DATABASE=esld;HOST=esl.tamu.edu;SERVER=eslol;SERVI
CE=sqlexec;PROTOCOL=olsoctcp;"
    cnn1.Open strcnn1
    Set rstsiteinfo1 = New ADODB.Recordset
  End If

  sqlquery1 = "select max(timestamp) from ch" + Trim(chan)
  rstsiteinfo1.Open sqlquery1, strcnn1, , , cmdtext
  time1 = rstsiteinfo1(0)

  Call ftp_thing_dir(" ", " ", " ", LCase(Trim(place)))

  Open scratch2 For Output As #3
  Open scratch1 For Input As #2
  Do Until EOF(2)
    Line Input #2, outbuf
    i = InStr(1, Trim(outbuf), " ")
    If InStr(i + 1, Trim(outbuf), ":") = 0 Then
      time2 = Mid(Trim(outbuf), 1, i) + " 00:00:00"
    Else
      unit1 = Mid(Trim(outbuf), 1, i)
      j = InStr(i + 1, Trim(outbuf), " ")
      unit2 = Mid(Trim(outbuf), i + 1, j - (i + 1))
      time2 = Trim(unit1) + " " + Trim(unit2)
      'MsgBox Trim(Str(time2))
      'End
    End If

    If time2 > time1 Then
      Print #3, outbuf
    End If
  Loop
  Close

  Open scratch1 For Output As #3
  Open scratch2 For Input As #2
  Do Until EOF(2)
  Line Input #2, outbuf
  'Debug.Print "max time = " + Str(time1) + " first record time = " + Trim(outbuf)
  'End
  Print #3, outbuf
 Loop
 Close

 Open scratch1 For Input As #3
 Line Input #3, buffer1
 FileDateTime = Trim(Mid(Trim(buffer1), 1, InStr(1, Trim(buffer1), " ")))

 i = 0
 j = 0
 thefile = logger + Trim(julian(FileDateTime)) + ".raw"
 outfile = "\\eslnt\common\dlockhar\loggers\abb\data\" + thefile
 Open outfile For Output As #2
 'Debug.Print "OPEN:  " + outfile
 Do Until EOF(3)
  i = i + 1
  If i <= 999 Then
   Line Input #3, buffer1
   Print #2, buffer1
  Else
   Close #2
   Call ftp_thing_send(outfile, Trim(thefile), "lstaraxp.tamu.edu", Trim(db))
   SENDFILES(j) = thefile
   'MsgBox "j= " + Str(j) + "   " + SENDFILES(j)
   j = j + 1
   i = 0
   FileDateTime = Trim(Mid(Trim(buffer1), 1, InStr(1, Trim(buffer1), " ")))
   thefile = logger + Trim(julian(FileDateTime)) + ".raw"
   'thefile = logger + Replace(Str(Date), "/", "") + Replace(Str(Time), ":", "") + ".raw"
   outfile = "\\eslnt\common\dlockhar\loggers\abb\data\" + thefile
   Open outfile For Output As #2
   'Debug.Print "OPEN:  " + outfile
  End If
 Loop
 Close
 If i > 1 Then
   Call ftp_thing_send(outfile, Trim(thefile), "lstaraxp.tamu.edu", Trim(db))
 End If
 SENDFILES(j) = thefile


 msg = "Raw load file(s) for site " + logger + " have been FTPd to lstaraxp '" + Trim(db) + "'
directory on " + Trim(Str(Date)) + " at " + Trim(Str(Time)) + "."
```

```
  logit (msg)

  log_in
  Call send_em(SENDFILES, logger, j, msg)
  log_off
  fs.movefile infile, "\\eslnt\common\dlockhar\loggers\abb\processed\"
ext2:

'---------------------END OF MAIN PROCESSING LOOP ------------------

Next

  fs.deletefile "\\eslnt\common\dlockhar\loggers\abb\polling_status.txt", True
  status ("inactive")
  Close
  'cnn.Close
  Call ftp_thing_close(outfile, thefile, "lstaraxp.tamu.edu", Trim(db))
  End
notgot:
 log_in
 'MsgBox sqlquery1
 msg = "Site " + logger + " does not exist on either lstar or esl databases.  abbpoll polling
terminated."
 Call send_em(SENDFILES, logger, j, msg)
 log_off
 End
 Resume Next
End Sub
Private Sub status(State As String)
 Open "\\eslnt\common\dlockhar\loggers\abb\polling_status.txt" For Append As #1
 Print #1, Trim(State)
 Close
End Sub

Public Sub log_in()
  abbpoll.MAPISession1.SignOn
  abbpoll.MAPIMessages1.SessionID = abbpoll.MAPISession1.SessionID
End Sub

 Public Sub log_off()
     abbpoll.MAPISession1.SignOff
 End Sub
Public Sub send_em(SENDFILES() As String, logger As String, j As Integer, msg As String)
    Dim i As Integer
    Dim str1, str2 As String
    Dim hndl As Integer
    Open "\\eslnt\common\dlockhar\loggers\abb\mail_people.txt" For Input As #2
   Do Until EOF(2)
     DoEvents
     Line Input #2, str1
     str1 = Trim(str1)
     'abbpoll.MAPIMessages1.
     abbpoll.MAPIMessages1.MsgIndex = -1
     abbpoll.MAPIMessages1.Compose
     abbpoll.MAPIMessages1.RecipDisplayName = str1
     abbpoll.MAPIMessages1.MsgSubject = "Database Load Files"
     str2 = "File(s) Sent: "
     For i = 0 To j
      If i <> j Then
        str2 = str2 + SENDFILES(i) + ", "
      Else
        str2 = str2 + SENDFILES(i)
      End If
     Next i
     abbpoll.MAPIMessages1.MsgNoteText = Trim(msg) + vbCrLf + str2

     abbpoll.MAPIMessages1.ResolveName
     abbpoll.MAPIMessages1.Send
   Loop
   Close #2
 End Sub
Public Function julian(dater As Date) As String
  Select Case Month(dater)
  Case 1
   If Len(Trim(Str(Day(dater)))) = 1 Then
    julian = Mid(Trim(Str(Year(dater))), 3, 2) + "00" + Trim(Str(Day(dater)))
   Else
    julian = Mid(Trim(Str(Year(dater))), 3, 2) + "0" + Trim(Str(Day(dater)))
   End If
  Case 2
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + "0" + Trim(Str(31 + Day(dater)))
  Case 3
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + "0" + Trim(Str(59 + Day(dater)))
  Case 4
   If Len(Trim(Str(90 + Day(dater)))) = 2 Then
     julian = Mid(Trim(Str(Year(dater))), 3, 2) + "0" + Trim(Str(90 + Day(dater)))
   Else
     julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(90 + Day(dater)))
   End If
  Case 5
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(120 + Day(dater)))
 Case 6
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(151 + Day(dater)))
 Case 7
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(181 + Day(dater)))
 Case 8
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(212 + Day(dater)))
 Case 9
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(243 + Day(dater)))
 Case 10
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(273 + Day(dater)))
 Case 11
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(304 + Day(dater)))
 Case 12
   julian = Mid(Trim(Str(Year(dater))), 3, 2) + Trim(Str(334 + Day(dater)))
 Case Else
 End Select

End Function
Sub logit(msg As String)
 Open "\\eslnt\common\dlockhar\loggers\abb\abb_logfile.txt" For Append As #1
   Print #1, msg
 Close
End Sub

Public Sub ftp_thing_on(sendfile As String, rcvfile As String, site As String, dir As String)
  With abbpoll.Inet1
  .URL = Trim(site)
  .UserName = "dataload"
  .Password = "upl0ad"
  .Protocol = icFTP
 End With
 abbpoll.Inet1.Execute abbpoll.Inet1.URL
 wait
End Sub
Public Sub ftp_thing_dir(sendfile As String, rcvfile As String, site As String, dir As String)
 'Debug.Print "dir: " + Trim(abbpoll.Inet1.URL) + " cd " + LCase(Trim(dir))
 abbpoll.Inet1.Execute abbpoll.Inet1.URL, "cd " + LCase(Trim(dir))
 wait
End Sub

Public Sub ftp_thing_send(sendfile As String, rcvfile As String, site As String, dir As String)
 'Debug.Print "send: " + Trim(abbpoll.Inet1.URL) + " put " + LCase(Trim(sendfile)) + " " +
LCase(Trim(rcvfile))
 abbpoll.Inet1.Execute abbpoll.Inet1.URL, "put " + LCase(Trim(sendfile)) + " " +
LCase(Trim(rcvfile))
 wait
```

```
End Sub
Public Sub ftp_thing_close(sendfile As String, rcvfile As String, site As String, dir As String)
 ''Debug.Print "close: " + Trim(abbpoll.Inet1.URL) + " close"
 abbpoll.Inet1.Execute abbpoll.Inet1.URL, "close" ' Close the connection.
 wait
End Sub

Private Sub Inet1_StateChanged(ByVal State As Integer)

 Select Case State

  Case icNone '0 No state to report.
    'Debug.Print "State 0"
  Case icHostResolvingHost '1 The control is looking up the IP address of the specified host
computer.
    'Debug.Print "State 1"
  Case icHostResolved '2 The control successfully found the IP address of the specified host
computer.
    'Debug.Print "State 2"
  Case icConnecting '3 The control is connecting to the host computer.
    'Debug.Print "State 3"
  Case icConnected '4 The control successfully connected to the host computer.
    'Debug.Print "State 4"
  Case icRequesting '5 The control is sending a request to the host computer.
    'Debug.Print "State 5"
  Case icRequestSent '6 The control successfully sent the request.
    'Debug.Print "State 6"
  Case icReceivingResponse '7 The control is receiving a response from the host computer.
    'Debug.Print "State 7"
  Case icResponseReceived '8 The control successfully received a response from the host
computer.
    'Debug.Print "State 8"
  Case icDisconnecting '9 The control is disconnecting from the host computer.
    'Debug.Print "State 9"
  Case icDisconnected '10 The control successfully disconnected from the host computer.
    'Debug.Print "State 10"
  Case icError '11 An error occurred in communicating with the host computer.
    'Debug.Print "State 11"
  Case icResponseCompleted '12 The request has completed and all data has been received.
    'Debug.Print "State 12"
 End Select
End Sub
Public Sub wait()
 Do While abbpoll.Inet1.StillExecuting
  DoEvents
 Loop
```

```
End Sub
```