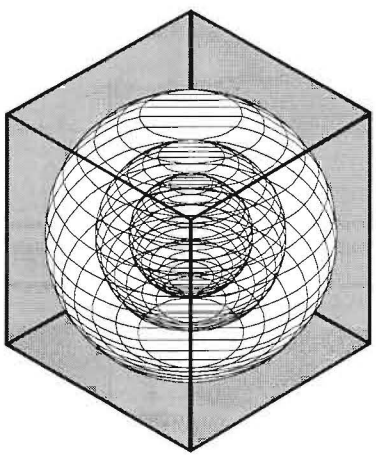
ESL Monthly Energy Consumption Report (MECR) Production Manual

Zi Liu
Jim Sweeny
Suwon Song
Jeff S. Haberl, Ph.D., P.E.

March 2005



ENERGY SYSTEMS LABORATORY

Texas Engineering Experiment Station
Texas A&M University System

ABSTRACT

This report provides the production manual for Monthly Energy Conservation Report (MECR). The MECR are six page reports of consumption and savings provided to the agencies responsible for the buildings being monitored.

Chapter 1 is an overview of the MECR production process. In Chapter 2 and 3, detail instructions on how to produce the ESL MECR and TAMU MECR. Examples and parts of programming scripts are presented to illustrate the process. The major components of the MECR program and production related files are described in Chapter 4.

An appendix is also provided that includes information on VI commands, UNIX commands, and the MECR applications. The MECR applications and an example month of MECR files are provided in the attached CDROM.

Table of Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 Monthly Energy Consumption Reports (MECR)

The Monthly Energy Consumption Reports are six page reports of consumption and savings provided to the agencies responsible for the buildings being monitored.

A MECR includes:

- A title page with usage totals, energy consumption retrofit savings and comments to the agency about observed anomalies.
- Scatter plots of daily chill and hot water usage versus daily average dry bulb temperature.
- Time series plots of hourly hot and chill water usage.
- Time series plots of total building electricity consumption, sub-metered electricity consumption, ambient dry bulb temperature, and relative humidity
- Two three-dimensional surface plots that graph hourly total building electric use. These plots are created by SAS software.
- A summary page with occupancy schedule, retrofit dates, etc.

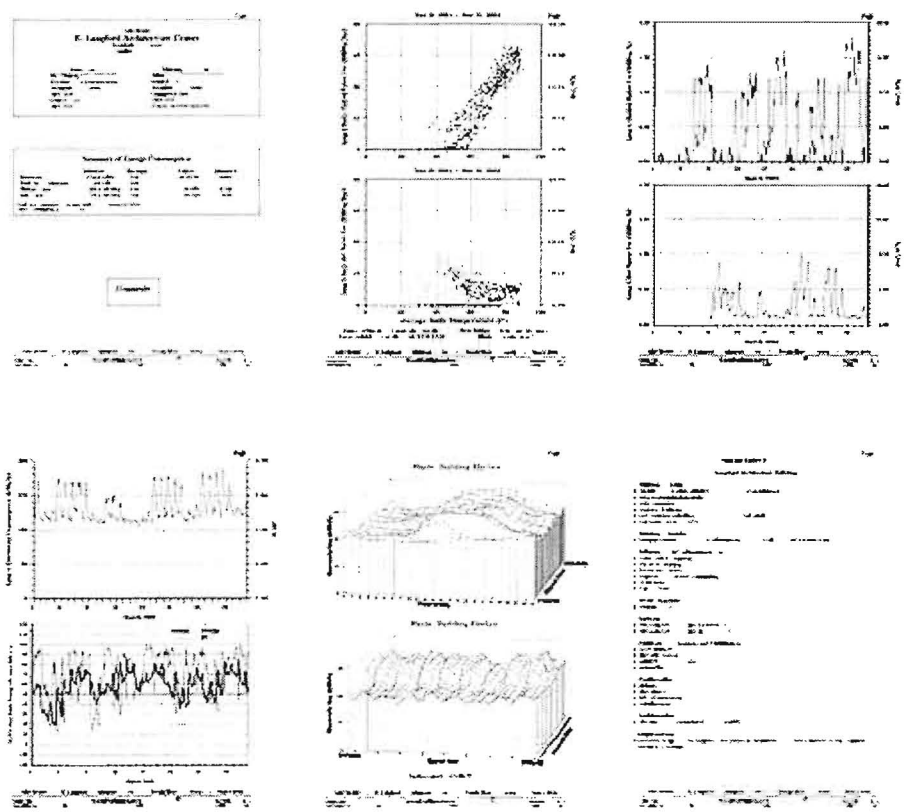


Figure 1.1 Monthly Energy Consumption Report (Pages 1 to 6)

Weekly files are concatenated into monthly files and hourly data is converted into daily data found on page two by MAKEDAY.BAT. A similar conversion script calculates the average dry bulb temperatures. Also, comments on IPNs are collected for the MECR by the data processing staff. The MECR is circulated once to collect comments and to catch processing errors, then it is circulated again to a subset of reviewers who compare it with other months' data, coordinate final corrections with data processing staff, and remove the bad data from the report. It takes 120 man hours, not including review time by the principal investigators, to produce a MECR. To circulate the MECR for comment and to print the final draft takes another three days.

After the first report is sent to a newly monitored agency, a session is scheduled so personnel from the monitoring program can meet with facilities engineers and operating personnel to discuss the report format, to answer questions, to get feedback on report format, and to offer suggestions for improvements. The MECR allows staff to visualize a greater window of information to catch anomalies that may not be apparent in the IPNs.

1.2 Overview of MECR Production

MECR requires three steps to be completed. The first step is to make the first draft using "Load_draft1" command and generate the comment sheet. The 'Load_draft1' executes three procedures sequentially including Rpt-cons, Rpt-save, and Draft1.

In second step, the savings file is inserted in the second draft after it was generated in each site. If the comment file is reviewed, the comments need to be typed or modified in each site on the second MECR draft.

In the final draft, usually the several modifications need to be done such as editing graph label, deleting the pages that have no data, and so on. For example, if a figure label is missing, the label must be added. If data must be marked 'bad' among some period, its result needs to be reflected in the MECR. In that case, the corresponding site is usually run again to adjust the scale factor or reload the raw data after it is identified.

Once the final draft is completely reviewed, make copies and take them to the secretary with the latest sending list.

1.3 Language used in MECR Production

MECR is produced using several program languages including:

- Embedded C: Most of code for database accessing in MECR production is written using embedded C.
- Tex: The final presentation of MECR is generated as the Tex form. After the Tex file is compiled, the postscript file is made.
- SAS: All the graphs in the MECR are produced using SAS.
- Awk- the temporary file processing is done using 'awk'
- C-Shell script – The script controls the whole process of working in MECR.

1.4 Log in to ESL DB

1) Log in to ESL DB using X-Win 32
   Username: lvk4173
   Password: lvk4412

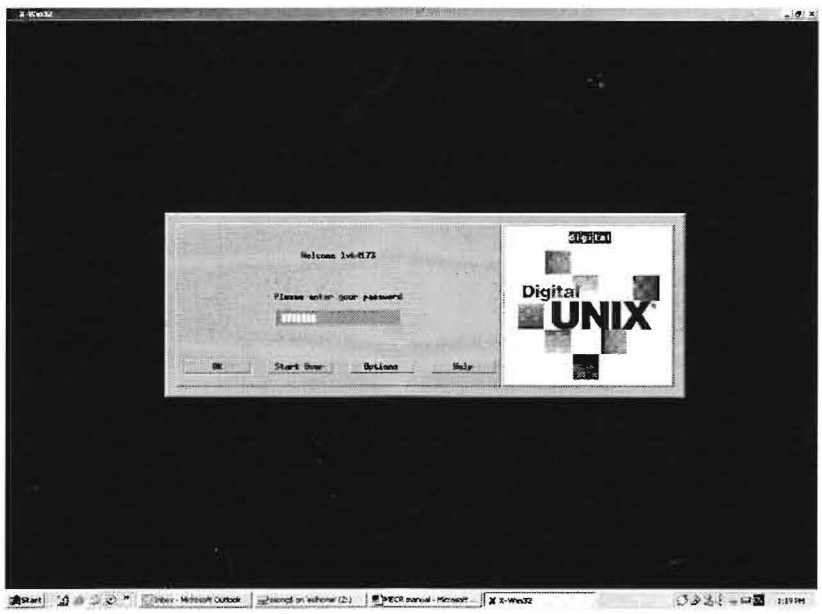


Figure 1.2 X-Win 32 screen asking for user name and password

2) Working Directory
   ESL MECR: ../prod/mecr/esl/mmyy

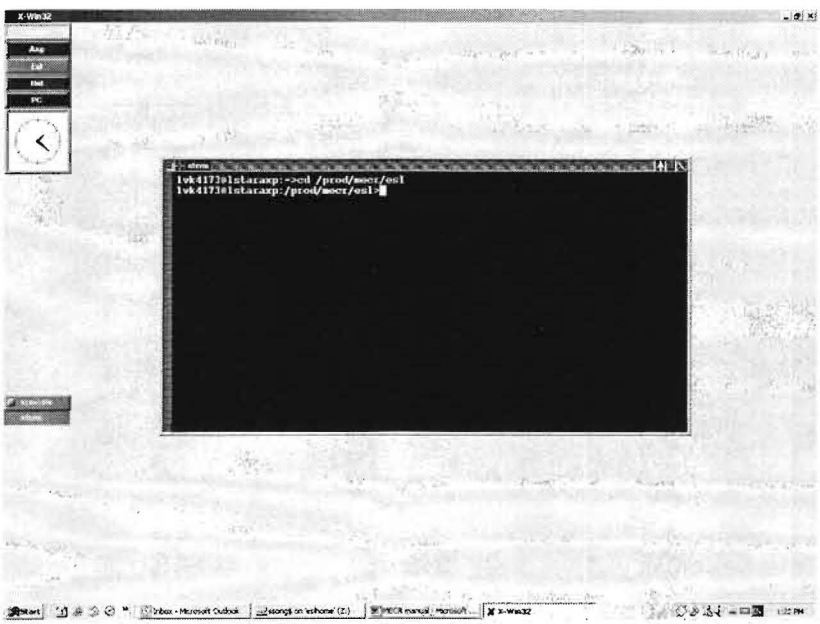


Figure 1.3 X-Win 32 screen showing the working directory for producing ESL MECR

2 PROCESS OF ESL MECR PRODUCTION

The processes to produce ESL and LoanSTAR MECRs are basically the same. In this chapter, detailed instruction with examples will be given on how to produce the ESL MECR. The entire process of MECR production is basically controlled using C-Shell script so that one needs to be familiar with the C-script command to produce the MECR. In the following, the C-scripts are also included in each step for a better understanding of the MECR production process and program.

2.1 Preparation

1) Create a new directory mmyy (e.g., 0804) for the current month using "mkdir" command.

```
lvk4173@lstaraxp:> cd /prod/mecr/esl
.. /prod/mecr/esl> mkdir 0804
../prod/mecr/esl> cd 0804
../prod/mecr/esl/0804>
```

2) Once a working directory is created for a new month, copy the related files (load_draft1, draft1, eslsite) from last month's directory to this month's directory using "cp" command as shown in the following.

```
../0704 > cp eslsite ../0804
../0704 > cp load_draft1 ../0804
../0704 > cp draft1 ../0804
```

3) If new sites need to be added or old sites need to be deleted for this month's MECR, open file "eslsite" using "vi" command as shown in the following script box and then add or delete the sites in the file. To save the updates, press "shift" and ":" at the same time and then type "wq", to quit without saving the updates, type "q" only.

```
0804> vi eslsite
146 Dallas CGC
951 Administration Building
952 Records Complex
953 Decker Correctional
954 Health & Human Service Building
955 Health & Human Service Building (Logger 2)
956 Cook/Chill Warehouse
957 Lew Sterrett Complex
938 87000 Block Thermal Plant
947 III Corp Building
940 Darnall Hospital
270 Stephen
271 Franklin
272 Thomas
273 Robert. E. Lee
274 Lincoln
275 Sam Houston
276 Travis
1001 Gutiell
1002 Perez
1003 Ruiz
1004 Service
1005 United
1006 Finley
```

2.2 The First Draft of ESL MECR

2.2.1 Overview

Once the data is loaded in the database, the first draft of MECR is generated using Load_draft1 (script). Its general role is to read the data from database and make representation of data according to the format of MECR.

- Load_Draft1 (script which is made with C shell script) executes following procedures sequentially.
    - rpt-cons
    - rpt-save
    - draft1

Also comment file for each site needs to be printed.

2.2.2 Printing the First Draft

The MECR procedure generates the first drafts of LoanSTAR and ESL MECR together. Listed below are the steps.

- Make a new directory for a new LoanSTAR and ESL MECR /prod/lstmecr/<mmyy> and /prod/eslmecr/<mmyy>.
- Copy files sitelist, draft1, load_draft1 from the previous month LoanSTAR MECR and files eslsite and draft1 from the directory of previous month ESL MECR.
- Edit three files, esldraft1, draft1, and load_draft1 by changing month and year.
- Start to run the script file, load_draft1. It normally takes a few hours.
- After load_draft1 is finished, print LoanSTAR MECR or ESL MECR. To print them, use script file, l_print (LoanSTAR) or newprint (ESL).
- Copy 4eslsite, 4load_draft1 and 4draft1 for ESL and 4sitelist, 4load_draft1 and 4sitelist for LoanSTAR MECR from the previous month directory.
- Make a list of sites that have problem and put the list of sites into 4eslsite and change the month and year in 4load_draft1 and 4draft1 (4sitelist, 4load_draft1 and 4sitelist for LoanSTAR MECR).
- Run the script, 4load_draft1.
- Print each site that had problems separately and make sure there are no problems anymore.
- Make comment sheet. In the directory where the comment file is put, (let's say, '/prod/mecr/lstar/compg'), first of all, two Tex files need modification (gencom.tex and commtable.tex). After updating month and year, compile two files and get the correct postscript file of two Tex files. Then sequential "lcom_start", "lcom_make" and "lcom_print" will make the comment sheet.

In the following, the major steps for generating first draft are illustrated in details with examples.

1) Open and edit the "load_draft1" and "draft1" to update mmyy to the current month and year as shown in the following box. Save the updated file and go back to the previous working directory.

```
0804> vi load_draft1
```

```
#! /bin/csh
echo "started:  `date`" > 0804.time
  rpt_cons 08 04 eslsite
  rpt_sav  08 04 eslsite
echo "finished: `date`" >> 0804.time
echo "started:  `date`" > 0804.mecr
  ./draft1
echo "finished: `date`" >> 0804.mecr
```

```
0804> vi draft1
```

```
#! /bin/csh
foreach i (`gawk '{print $1}' eslsite`)
  eslmecr $i 08 04
  rm imecr.$i.0804.imraw
  gzip -f *.$i.*
end
```

2) Once the data is loaded in the database and the month and date are updated, the first draft of MECR is generated using "Load_draft1" command.

```
0804> ./load_draft1
```

3) Print first draft using "newprint" command for all sites with "-t" option.

```
0804 > newprint 08 04 –t
```

2.2.3 **Making Comment Sheets**

In the comment sheet, the energy consumption data for the current month, last month, and the same month last year are shown in the table for each site. The reviewer of MECR provides comments on each site based on the increase or decrease of the energy usage of current month when compared to the same month last year.

1) Copy *.comments from last month's directory to /prod/mecr/esl/compg.

```
0704> cp *.comments /prod/mecr/esl/compg
```

2) Run "ecom_start".

```
.. compg> ./ecom_start
```

3) Open commtable.tex and gencom.tex by using "vi" command and update month and year as shown in bold fonts in the following script boxes.

```
.. compg> vi commtable.tex
```

```
\nopagenumbers
\vsize = 10.5 true in
```

```
\font\bigletter=cmr10 scaled \magstep3
\font\smallletter=cmr10 scaled 694
\def\note#1{\vbox{\font\bigbold=cmb10 scaled \magstep0
\font\bigmath=msam10 scaled \magstep0
\bigbold {\bigmath F}
#1
\vskip .2\baselineskip
}}

\def\boxit#1{\vbox{\hrule\hbox{\vrule\kern8pt\vbox{\kern8pt{#1}\kern8pt}\enskip\kern 8pt\vrule}\hrule}}

\def\cbox#1{
\vbox{
\null\hfill
\boxit{
\vskip 6pt
\advance\hsize by -.1\hsize
{#1}
\vskip 6pt}%boxit Comments
\hfill\null
}
}% end cbox

\def\ecompbox{\vbox{\boxit{\advance\hsize by -.1\hsize
\hfill
\smallletter
\baselineskip=0.694\baselineskip
\input 001.ecomp
\hfill\null
}}}

\def\oldcomments{\cbox{
\hfill Last Month's Comments \hfill\break
\input 001.comments
\hfill\null
}}

\def\newcomments{\cbox{
\hfill Comments for Page 1 \hfill\break
\vbox{\vglue 1.5 true in}
}}

\centerline{\bigletter August 2004}
{\bigletter ZEC\hfill Site\# 001}
\vskip\baselineskip

\newcount\n
\n = 1

\def\newrow#1,#2,#3 {\halign {
\strut\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .8 \hsize{##\hfill}
&##\vrule\cr
\noalign{\hrule}
&{#1}&&{#2}&&{#3}&\cr
\noalign{\hrule}
}
```

```
}

\newrow Initials,Page,Comment

{
\loop{ \newrow {\phantom{\bigletter A}},{},{} }
\ifnum\n < 15
\advance \n by 1
\repeat
}

\vskip\baselineskip
\ecompbox
\oldcomments
\newcomments
\bye
```

```
.. compg> vi gencom.tex

\nopagenumbers
\vsize = 10.5 true in

\font\bigletter=cmr10 scaled \magstep3
\def\note#1{\vbox{\font\bigbold=cmb10 scaled \magstep0
\font\bigmath=msam10 scaled \magstep0
\bigbold {\bigmath F}
#1
\vskip .2\baselineskip
}}

\def\boxit#1{\vbox{\hrule\hbox{\vrule\kern8pt\vbox{\kern8pt{#1}\kern8pt}\enskip\kern 8pt\vrule}\hrule}}

\def\cbox#1{
\vbox{
\null\hfill
\boxit{
\vskip 6pt
\advance\hsize by -.1\hsize
{#1}
\vskip 6pt}%boxit Comments
\hfill\null
}
}% end cbox

\def\oldcomments{\cbox{
\hfill Preparation Comments \hfill\break
\hfill\null
}}

\def\newcomments{\cbox{
\hfill Comments for Page 1 \hfill\break
\vbox{\vglue 1.5 true in}
}}

\centerline{\bigletter August 2004}
{\bigletter General\hfill}
\vskip\baselineskip

\newcount\n
\n = 1
```

```
\def\newrow#1,#2,#3 {\halign {
\strut\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .8 \hsize{##\hfill}
&##\vrule\cr
\noalign{\hrule}
&{#1}&&{#2}&&{#3}&\cr
\noalign{\hrule}
}
}

\newrow Initials,Page,Comment

{
\loop{ \newrow {\phantom{\bigletter A}},{},{} }
\ifnum\n < 19
\advance \n by 1
\repeat
}

\oldcomments

\newcomments

\bye
```

4) Run "ecom_make" to generate comment sheets.

```
.. compg > ./ecom_make mm yy
```

5) Print comment sheet.

```
.. compg > ./ecom_print
```

6) Print cover page separately.

```
.. compg > lpr -Plj052 gencom.ps
```

To print in other printers, change printer name (Plj052) by editing "ecom_print" script.

```
.. compg >  vi ecom_print
#!/usr/local/bin/perl

open (SITELIST, "./eslsite") || die "Couldn't open sitelist, stopped";

$range = 0;
$debug = 0;
$retex = 0;

for $i (1..$#ARGV) {
  if ($ARGV[$i] eq "-t") { $retex = 1; }
```

```
  if ($ARGV[$i] eq "-r") {
    $range = 1;
    $from = $ARGV[++$i];
    $to = $ARGV[++$i];
  }
}

print "Printing comments for ";
if ($range) {
  print "sites $from to $to.\n";
} else {
  print "all sites.\n";
}
$i = 0;
while(<SITELIST>) {
  ($siteList[$i], $siteDescript[$i]) = split(/ /);
  if ($range) {
    if ($siteList[$i] eq $from) {
      $fromElement = $i;
    }
    if ($siteList[$i] eq $to) {
      $toElement = $i;
    }
  }
  $i++;
}

if (!$range) {
  $fromElement = 0;
  $toElement = $#siteList;
}

for ($i = $fromElement; $i <= $toElement; $i++) {
  sleep 1;
  system("lpr -Plj052 ./$siteList[$i]com.ps\n");
}
```

2.3 The Second Draft of ESL MECR

2.3.1 Overview

When the savings file is ready, the saving file for each site is generated using command 'make_sav'. 'make_sav' makes savings record for each site. To accomplish this, savings file needs to be transferred into MECR production directory using ftp character transfer option first. If ftp with binary transfer option is used, the calculation result will not be correct.

Also when comment sheet is filled with comment, it needs to be included in the second draft. First user needs to type the comment for each site. The comment sheet is composed using Tex. Therefore user needs to be cautious in inserting the comment following the Tex grammar. User needs to copy the corresponding comment file into working directory. Then, 'load_draft2' is executed to generate the second draft.

Major difference between load_draft1 and load_drfat2 is in load_draft2 savings columns from MMMYY-SV.TXT (result after running 'make_sav') are loaded into rpt_sav table and the comment is inserted in second draft.

In the following, the major steps for generating first draft are illustrated in details with examples.

2.3.2 Adding Savings Data

1) Check if the savings file is available in P:/MECR/ESL_CONTRACT.

2) Copy "MC-mmmyy.txt from P:/MECR/ESL_CONTRACT/ to prod/mecr/esl/mmyy and rename to MMMyy-SV.TXT using FTP (ascii mode) as shown in Figure 2.1.

3) Run the savings file and generate *.savings file using "makesav" command.

```
..0804> makesav FEB03-SV.TXT
```

4) Run "rpt_sav".

```
..0804>  rpt_sav mm yy
```

5) Run "4load_draft" to add savings for the sites that need to report savings.

```
..0804> ./4load_draft
```

6) Run "gunzip *.*"

7) Print the second draft.

```
..0804 >newprint mm yy [-r from to] -t
```

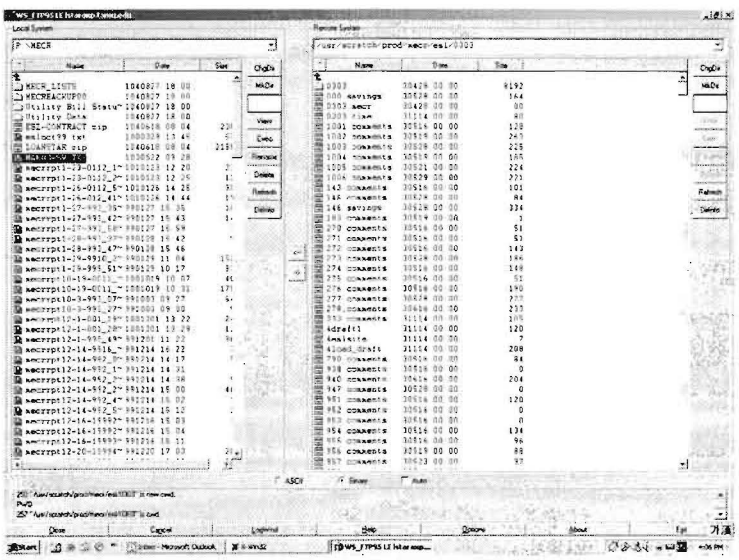


Figure 2.1 An example of FTP Program

2.3.3 Updating the Comments in Second Draft

1) Copy comments file from last month's directory to the current directory.

```
..0704> cp *.comments ../0804
```

2) Edit each site's comments file based on the comments provided in the comment sheet by the MECR reviewer. Once the comments are edited, save the files and exit to the previous working directory.

```
..0804> vi 938.comments
\note {All data are missing from 08/01/04 to 08/02/04 due to logger communication problems.}
\note {Chilled water use has increased significantly when compared to August 2003.}
```

3) Run 4load_draft for the specific sites that need to be updated.

```
..0804>  ./4load_draft
```

Or, run the related script files for a specific site to make the necessary changes and preview it.

```
..0804>  tex Report.938.0804.tex
..0804>  dvips Report.938.0804.dvi
..0804>  gv Report.938.0804.ps
```

4) Print out specific sites.

```
..0804>  newprint 08 04 –r 146 146
```

Or, open the Report file of a specific site using "gv" command as shown in the following script box and select "print " to print out all pages of the specific site from the screen of "Ghostview" program as shown in Figure 2.2

```
..0804>  gv Report.938.0804.ps
```

2.3.4 Compiling Specific One Site

1) Copy the related files (4load_draft, 4eslsite, 4draft1) from last month's directory to this month's directory.

```
../0704 > cp 4eslsite  ../0804
../0704 > cp 4load_draft  ../0804
../0704 > cp 4draft1  ../0804
```

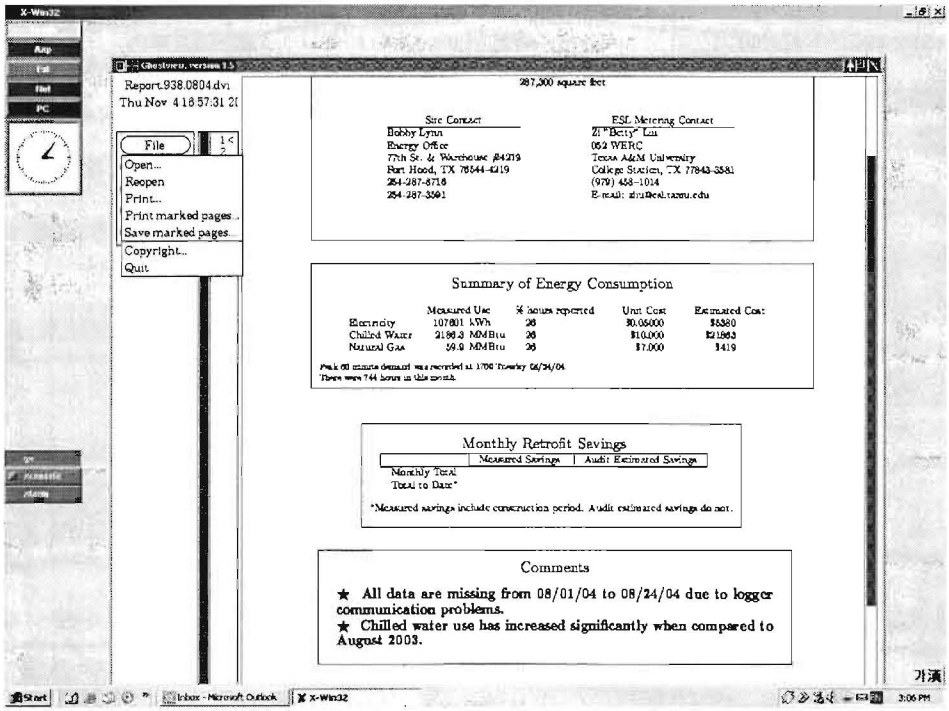


Figure 2.2 An example of the first page of MECR on the "Ghostview" Program

2) Edit 4load_draft and 4draft and update mm yy to the current month and year.

```
..0804> vi 4draft1
#! /bin/csh
foreach i (`gawk '{print $1}' 4eslsite`)
  eslmecr $i 08 04 1
  rm imecr.$i.0804.imraw
  gzip -f *.$i.*
end
```

```
..0804> vi 4load_draft
#! /bin/csh
echo "started:  `date`" > 0804.time
  rpt_cons 08 04 4eslsite
  rpt_sav  08 04 4eslsite
  rpt_sav  08 04
  rm -f rpt_sav.*.0804 rpt_cons.*.0804
  ./4draft1
echo "finished: `date`" >> 0804.time
```

3) Remove existing file before run.

```
..0804> rpt_cons mm yy 4eslsite
```

4) Run "4load_draft".

```
..0804> ./4load_draft
```

5) Print out specific sites.

```
..0804> newprint 08 04 –r 938 938
```

2.4 The Final Draft of ESL MECR

2.4.1 Overview

In the final draft, usually the several modifications need to be done such as reloading the raw data to fill data gap or replace bad data with new data, editing graph label, deleting a specific page that has no data, and so on.

In the case where there are several modifications, user does correction manually. For example, if the label of a graph is not made correctly, user adds the correct label or adjusts the position of label manually using the temporary file, which is generated as 'lstar_mecr' with compile option.

Usually the modification work is done in the result of graph. When the graph is not correctly made, user needs manual working. There are many cases when the graph is not correctly made. In that case, if the user knows how the graph is made, it helps a lot for modification.

'imecr' (script) explains how the graph is made in each page. 'imecr' reads the corresponding data into file from database and generates the graph using SAS. At that time, SAS makes the temporary file such as the position of label in graph or the SAS file for making the graph. Therefore most of modification can be made if temporary file is traced.

There are the cases where the data itself in the database needs to be modified. Usually the reason of modification of data is that data is proved to be 'wrong' by analysts or commissioning engineers. In that case, data is marked bad using script. Also the MECR report for that site needs to be run again. To generate one site MECR, not the whole list of MECR, the user needs to run '4load_draft' scripts. The '4load_draft' will do the same work like load_draft1 except that load_draft1 will do the work for all sites in MECR.

2.4.2 Reloading Raw Data to the ESL DB

If the data for a MECR site are marked 'bad' or 'missing' for some period, its result needs to be reflected in the MECR. In that case, the corresponding site is usually run again.

1) Go to the raw data directory in ESL DB.

```
Lvk4173@lstaraxp> cd /prod/raw
```

An example of raw data file name format is represented in the following box:

```
ex) 19204037.raw_loaded
    Where,        192: site #
```

```
04: year
037: julian map, which is referred to the web
     (http://eslnt.tamu.edu/dbadmin/julian_map.htm)
```

2) Remove the previous raw data file using "rm" command.

```
../prod/raw> rm 19200367.raw_loaded 19200367.raw
```

3) Reload the new raw data using "reload" command.

```
../prod/raw> reload 19200367.raw
```

2.4.3 Editing Graph Label

If the label is missing or misplaced in a graph, the label should be added or adjusted.

1) Generate temporary file by running "eslmecr" with debug mode 1(off: 0).

```
..0804>eslmecr site# mm yy 1
ex) >eslmecr 951 08 04 1
```

2) Edit temporary file. For example, to edit graph at page 4, edit pg4.tmp2. Change it to -99.0000 (means null) if there is no data. Each value listed in the pg4.tmp2 file indicates the position of labels in the graph of page 4.

```
..0804> vi pg4.tmp2
 8613.0000000   146.000000  35.898000  46.593
```

3) Recompile sas file, preview the graph and print the pages as shown in the example below.

```
..0804> sas s4.951.0804a.sas
        gv s4.951.0804a.ps
        tex Report.951.0804.tex
        dvips Report.951.0804.dvi
        newprint –r 951 951
```

2.4.4 Editing Y-axis

1) Unzip related files .sas.*. For example, to edit the lower graph at page 2 for site 938, unzip files s2.938.0804b.sas.*. In this file name, 's2' indicates page number 2 and 'b' indicates the second graph in the page 2.

```
..0804> gunzip s2.938.0804b.sas.*
```

2) Open the sas file corresponding to the page need to be edited using "vi" command and then edit the scale for Y axis. Change both axis 1 and axis 3 the same time. For example, divide both maximum value and unit by 4.

..0804> vi s2.938.0804b.sas

```
filename gsasfile 's2.938.0804b.ps';
data sheet2;
infile "imecr.938.0804.pg2";
input dd dow$ chw hw db;

array a hw db;
do over a;
  if a = -99.0 then delete;
end;
output;

run;

data anno;
  length function style color $ 8 text $ 20;
  retain xsys '2' ysys '2' hsys '1' when 'a';
  set sheet2;
  function='label'; color='black';
  size=2; text=dow; position='5';
  style='triplex'; x=db; y=hw;
  output;

title1 justify=center height=0.20in color=black 'Jun 01 2003 - Jun 30 2004';

goptions display device = pslepsf

     ftext = triplex
    ftitle = triplex
     cback = white
    colors = (black white)
  chartype = 11
     htext = 0.20 in
     hsize = 7.5 in
     vsize = 6.5 in
   horigin = 0.0 in
   vorigin = 0.0 in
   gaccess = gsasfile
   gsfmode = replace;

symbol1 value=none;
axis1 order = (0 to 280 by 70)
     label = (height = 0.22in angle = 90 justify = center 'FH Power Plant Daily
 Natural Gas Use (MMBtu/Day)')
     major = (height=.3cm width=2)
     minor = (number=9 height=.2cm width=2)
    offset = (0,0)
     width = 2;

axis2 order = (0 to 100 by 20)
     label = (height=0.22in 'Average Daily Temperature (F)')
     major = (height=.3cm width=2)
     minor = (number=9 height=.2cm width=2)
    offset = (0,0)
     width = 2;

axis3 order = (0 to 40.60 by 10.15)
     label = (height = 0.22in angle = 90 justify = center 'Btu' height = 0.1in 'f' height = 0.22in '/ft' height = 0.1in '2'
height = 0.22in 'h')
     major = (height=.3cm width=2)
     minor = (number=9 height=.2cm width=2)
```

```
    offset = (0,0)
     width = 2;

proc gplot;
plot hw * db /annotate=anno
  haxis=axis2
  vaxis=axis1
  frame
  autohref autovref frame lvref=3 lhref=1;
plot2 hw * db /vaxis=axis3;
run;
quit;
```

3) Once a specific site is completely edited, recompile the sas files of a specific site including the graph to be edited according to the procedure shown in the following script box. Then use 'gv' command to preview the plot to see if the scale is changed correctly. Finally, preview the report for this site and print it.

```
..0804> sas s2.938.0804b. sas
..0804> gv s2.938.0804b. ps
..0804> tex Report.938.0804.tex
..0804> dvips Report.938.0804.dvi
..0804> gv Report.938.*.ps
..0804> newprint 08 04 –r 938 938
```

2.4.5 Deleting a Page or a Graph

1) Remove pages or graphs in the associated tex file if there are pages or graphs that have no data for a specific site. For example, if there is no data for both graphs (2a and 2b) in page 2 (s2), and need to delete the entire page, remove the corresponding parts in scripts (bold font) from 'null' to 'eject' as shown in the following scripts box. To delete a graph, for example, the first graph in page 3 (s3), delete only the script (bold font) for plotting that graph as shown in the following scripts box.

```
..0804>  vi Report.938.0804.tex
```

```
\input /usr/local/lsd/imecr/tex/globaldefs
\def\reportmonth{August}
\def\reportyear{2004}

\def\loannum{}
\def\loancount{}
\def\firstpage{1}
\def\is_blding_id{0}
\def\building_name{87000 Block Thermal Plant}
\def\site_name{Fort Hood Army Base}
\def\issitename{1}
\def\bldg_size{287,300}

\def\sitecontact{
Bobby Lynn\cr
Energy Office\cr
77th St. \& Warehouse \#4219\cr
Fort Hood, TX 76544-4219\cr
254-287-8716\cr
254-287-3591\cr
 \cr
```

```
 \cr
}
\def\metercontact{ESL }
\def\ourcontact{
Zi "Betty" Liu\cr
052 WERC\cr
Texas A\&M University\cr
College Station, TX 77843-3581\cr
(979) 458--1014\cr
E-mail: zliu@esl.tamu.edu\cr
}

\def\use_summary{
Electricity&107601&kWh&&26&&&\$0.05000&&&\$5380\cr
Chilled Water&2186.3&MMBtu&&26&&&\$10.000&&&\$21863\cr
Natural Gas& 59.9&MMBtu&&26&&&\$7.000&&&\$419\cr
}
\def\peaktime{Peak 60 minute demand was recorded at 1700 Tuesday 08/24/04.}
\def\numhours{744}

\def\savings_box{
\noalign{\hrule}
\omit&Monthly Total&\omit&&\omit&&\omit&&\omit&&\omit\cr
\omit&Total to Date$^*$&\omit&&\omit&&\omit&&\omit&&\omit\cr
}
\def\comments{
\input 938.comments
}

\def\topleft{Energy Systems Laboratory}
\def\botleft{Texas Engineering Experiment Station}
\def\toprght{Texas A\&M University}
\def\botrght{College Station, Texas}

\input /usr/local/lsd/imecr/tex/body

\null
\special{psfile=s2.938.0804a.ps voffset=-309 hoffset=-10 hscale=87 vscale=66}
\null
\special{psfile=s2.938.0804b.ps voffset=-605 hoffset=-10 hscale=87 vscale=66}
\vfill
\centerline{\vbox{\font\smaller cmr10 scaled 833\smaller\halign{\hfil#\hfil\quad&\quad\hfil#\hfil\cr
Data points for the current month are shown as letters.&Points from this month last year are shown as +.\cr
Monday through Sunday are represented as M,T,W,H,F,S,U.&All other points are shown as *.\cr
}}}
\eject

\null
\special{psfile=s3.938.0804a.ps voffset=-350 hoffset=-70 hscale=108 vscale=100}
\null
\special{psfile=s3.938.0804b.ps voffset=-660 hoffset=-70 hscale=108 vscale=100}
\vfill\ eject

\null
\special{psfile=s4.938.0804a.ps voffset=-350 hoffset=-55 hscale=100 vscale=100}
\null
\special{psfile=s4.938.0804b.ps voffset=-660 hoffset=-46 hscale=89 vscale=100}
\vfill\eject

\null
\vglue 0.0 truein
\epsfbox{s5.938.0804a.ps}
```

```
\vglue 0.25 truein
\epsfbox{s5.938.0804b.ps}
\nobreak\vfill\nobreak
\centerline{Sundays are marked with an ``S"}
\eject

\bye
```

2) Run again the script files to verify the correction.

```
..0804>  tex Report.938.0804.tex
..0804>  dvips Report.938.0804.dvi
```

3) Print out the updated pages.

```
..0804> newprint 08 04 –r 938 938
```

2.4.6 Printing the Final MECR

Two script files, "newprint" and "l_print", are used to print ESL and LoanSTAR respectively.

1) To print reports for all sites, use the following command.

```
..0804>  > newprint 08 04 –t
```

2) To print selected sites, for example, from sites 938 to 951 according to the list of sites in file 'eslsite', use the following command.

```
..0804>  newprint 08 04 –r 938 951
```

Or, open the report file of a specific site using 'gv' command and select 'print' in "Ghostview" screen as shown in Figure 2.2.

```
..0804>  gv Report.146.0804.ps
```

2.4.7 Making the Final Copies

After finishing all the correction and completing the final draft, take the final draft to the copy center to make necessary copies for each site according to the distribution document at Y drive (Y:\MECR\MECR_LISTS\SEND\ *.doc). Then the administrative assistant will distribute the MECRs to the customers.

3 PRECESS OF TAMU MECR PRODUCTION

3.1 Making the Main Contents (PS) Files

1) Make new(current) directory of this month.

Create a new sub-directory mmyy for the current month using "mkdir" command in the TAMU directory.

```
lvk4173@lstaraxp:> cd /prod/mecr/tamu
../prod/mecr/tamu> mkdir 0604
../prod/mecr/tamu> cd 0604
../prod/mecr/tamu/0604>
```

2) Copy the required files (load_draft1, draft1, and tamsite) from last month's directory to this month's directory using "cp" command.

```
../0504 > cp tamsite  ../0604
../0504 > cp load_draft1  ../0604
../0504 > cp draft1  ../0604
```

3) Edit "tamsite" if necessary.

Edit "tamsite" by using "vi" command as shown in the following script box if new sites need to be added or old sites need to be deleted.

```
0604> vi tamsite
001 Zachry Engineering Center
491 Evans Library (Old)
494 E. Landford Architecture Center
495 Old Architecture
496 Biological Sciences Building
497 Teague
498 Reed McDonald
499 Heldenfels Hall
509 Harrington Tower
510 Blocker
511 Oceanography and Meterology
512 Kleberg Animal&Food Sciences
513 New Chemistry Building
514 Chemistry (1959)
531 Chemistry (1972)
515 Bright Building
516 CE/TTI Tower
517 Petroleum Eng (Richardson)
518 Engineering/Physics Lab
519 Halbouty Geosciences
532 Halbouty Geosciences (New)
520 Engineering Research Center
521 Clinical Sciences (Vet)
522 Vet Med Hospital
523 Vet Med Center Addition
524 Soil & Crop/Entomology
525 Medical Sciences Building
526 Horticulture-Forest Sciencess
527 Biochemistry /Biophysics
528 New Business Building
585 State Headquarters
```

```
530 TI Building
533 Harrington Lab
534 Plant Science
535 EV Adams Band Hall
536 Academic
537 Biological Sciences Building E
538 Academic Adminstration
539 Anthropology
540 Agriculture Engineering
541 Cyclotron
543 Dulie Bell
544 Vet Sciences Building
545 Vet Hospital
547 Vet Med Admin Bldg
549 Southern Crop Improvement Facility
550 Ocean Drilling Facilities
551 West Campus Library
552 Medical Sciences Library
553 Offshore Technology
560 Wells Hall
561 Rudder Hall
562 Eppright Hall
563 Appelt Hall
564 Lechner Hall
565 Underwood Hall
566 Commons
568 Krueger Hall
569 Dunn Hall
570 Mosher Hall
571 West Dorm-Aston Hall
572 Clayton Williams Alumni Center
575 G. Rollie White Annex
576 Koldus Student Services
577 Cain Hall Kitchen
578 Rudder Auditorium
579 Duncan Dining Hall
580 G. Rollie White
581 Memorial Student Center
582 MSC Annex
586 Clements Hall
587 Haas Hall
588 McFadden Hall
589 Neely Hall
590 Hobby Hall
591 McKenzie Terminal
593 Recreational Sports & Natorium
```

5) Edit "load_draft1" and "draft1".

By using "vi" command, open the files (load_draft1 and draft1) and update "mmyy" to the current month and year and save the updated file.

```
0604> vi load_draft1
#! /bin/csh
echo "started:  `date`" > 0604.time
  rpt_cons 06 04 tamsite
  rpt_sav  06 04 tamsite
echo "finished:  `date`" >> 0604.time
echo "started:  `date`" > 0604.mecr
  ./draft1
```

```
echo "finished: `date`" >> 0604.mecr
```

```
0604> vi draft1
#! /bin/csh
foreach i (`gawk '{print $1}' tamsite`)
  lstarmecr $i 06 04
  rm imecr.$i.0604.imraw
  gzip -f *.$i.*
end
```

6) Run "load_draft1".

Once the data is loaded in the database (lvk4173@lastaraxp:/prod/raw/ *.*) and the month and date are updated, the TAMU MECR is generated by executing "Load_draft1" command.

```
0604> ./load_draft1
```

3.2 Overall Page Numbering

1) Page numbering for each site.

```
0604 > make_TAMU_pageno 06 04
```

2) Add each site page to overall.

```
0604 > add_TAMU_pageno 06 04
```

3) List overall site page.

```
0604 > list_TAMU_pageno 06 04
```

4) Check the result of overall page numbering.

```
0604 > tamprint 06 04 -r(range) from(site#1) to(site#n)-t(recompile .tex file)
```

Or open the Report file of a specific site using "gv" command and check the page number using "Ghostview" program as shown in Figure 2.2.

```
..0604>  gv Report.593.0604.ps
```

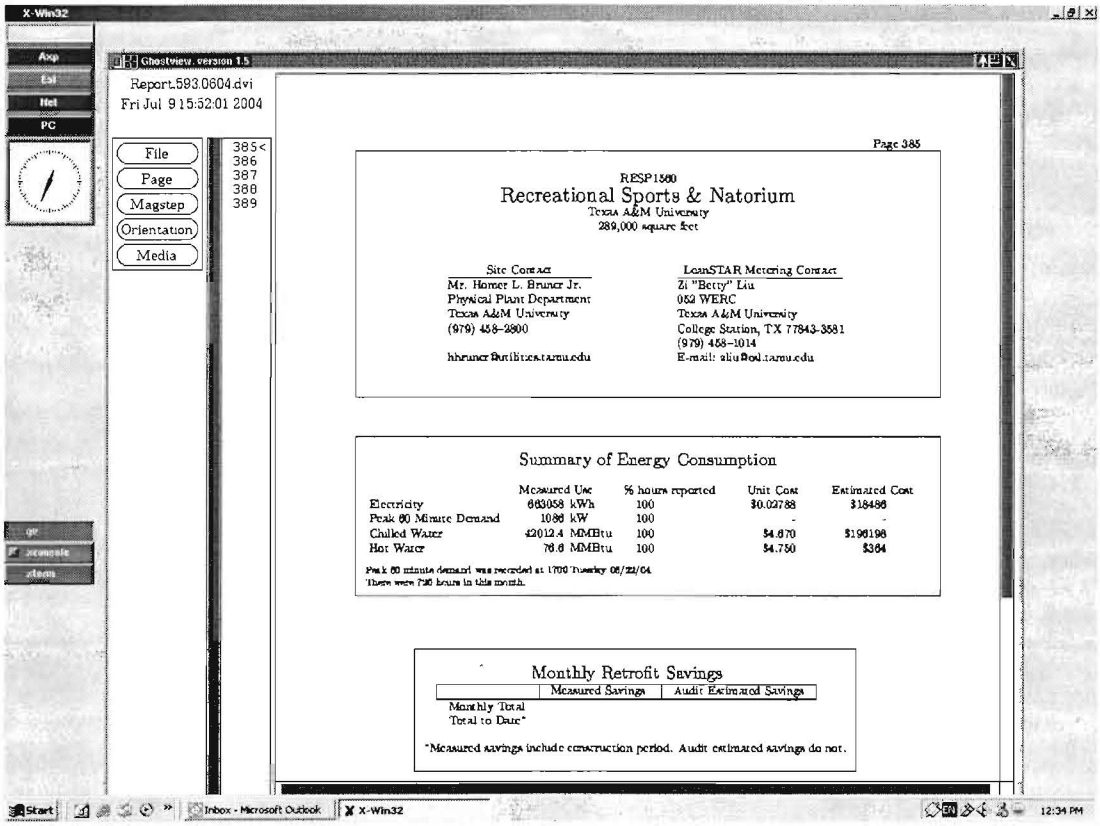


Figure 3.1 An example of the "Ghostview" Program showing the page number

5) Print, if the page numbers are correct.

```
0604 > tamprint 06 04 –t
```

3.3 Making Cover Page and Table of Contents

3.3.1 Making Cover Page

1) Copy Cover.tex from old month directory to current dir.

```
..0504 > cp Cover.tex ../0604
```

2) Edit Cover.tex by changing month to current one.

```
..0604 > vi Cover.tex
\newbox\centerdot \setbox\centerdot=\hbox{hskip.7em.\hskip.7em}
\nopagenumbers
\parindent=0pt
\parskip=1pt
\font\reallybig=cmb10 scaled 2488
\font\littlebigger=cmb10 scaled 1728
\font\big=cmr10 scaled 1440
\font\regular=cmr10
```

```
\font\notsobig=cmr10 scaled 833
\font\notsobigb=cmb10 scaled 833

\null
\vglue .6 true in
\centerline{\littlebigger Texas A\&M Monitoring and Analysis Program}
\vglue 1.0 true in
\centerline{\reallybig Monthly Energy}
\centerline{\reallybig Consumption Report{\raise .25 in\hbox{\rm\copyright}}\kern -.1 in}
\vglue .5 true in
\centerline{\reallybig June 2004}
\vglue 1.5 true in
\vfill

\big
{
\advance\lineskip 2pt
\centerline{\strut ESL INTERNAL DISTRIBUTION ONLY}
}
\vfill\vfill\vfill

\eject
\bye
```

3) Compile Cover.tex.

```
0604 > tex Cover.tex
0604 > dvips Cover.dvi
```

4) Print the Cover page showing current mmyy.

```
0604 > lpr -Plj052 Cover.ps
```

3.3.2 Making Table of Contents

1) Copy ToC.mmyy.tex from old month directory to current dir.

```
0504 > cp ToC.0803.tex ../0604
```

2) Compile ToC.mmyy.tex.

```
0604 > tex ToC.0803.tex
0604 > dvips ToC.0803.dvi
```

3) Print the Cover page showing current mmyy, using print "plj052".

```
0604 > lpr -Plj052 ToC.0803.ps
```

3.4 Making Comment Sheets

In the comment sheet, the energy consumption data for the current month, last month, and the same month last year are shown in the table for each site. The reviewer of MECR provides comments on each site based on the increase or decrease of the energy usage of current month when compared to the same month last year.

1) Go to /prod/mecr/tamu/compg.

2) Run "tamu_start" as shown in the following script box.

```
.. compg> ./tamu_start
```

3) Open the two files (commtable.tex and gencom.tex) by using "vi" command and update month and year.

```
.. compg> vi commtable.tex
\nopagenumbers
\vsize = 10.5 true in

\font\bigletter=cmr10 scaled \magstep3
\font\smallletter=cmr10 scaled 694
\def\note#1{\vbox{\font\bigbold=cmb10 scaled \magstep0
\font\bigmath=msam10 scaled \magstep0
\bigbold {\bigmath F}
#1
\vskip .2\baselineskip
}}

\def\boxit#1{\vbox{\hrule\hbox{\vrule\kern8pt\vbox{\kern8pt{#1}\kern8pt}\enskip\kern 8pt\vrule}\hrule}}

\def\cbox#1{
\vbox{
\null\hfill
\boxit{
\vskip 6pt
\advance\hsize by -.1\hsize
{#1}
\vskip 6pt}%boxit Comments
\hfill\null
}
}% end cbox

\def\ecompbox{\vbox{\boxit{\advance\hsize by -.1\hsize
\hfill
\smallletter
\baselineskip=0.694\baselineskip
\input 001.ecomp
\hfill\null
}}}

\def\oldcomments{\cbox{
\hfill Last Month's Comments \hfill\break
\input 001.comments
\hfill\null
}}
```

```
\def\newcomments{\cbox{
\hfill Comments for Page 1 \hfill\break
\vbox{\vglue 1.5 true in}
}}

\centerline{\bigletter June 2004}

{\bigletter ZEC\hfill Site\# 001}
\vskip\baselineskip

\newcount\n
\n = 1

\def\newrow#1,#2,#3 {\halign {
\strut\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .8 \hsize{##\hfill}
&##\vrule\cr
\noalign{\hrule}
&{#1}&&{#2}&&{#3}&\cr
\noalign{\hrule}
}
}
\newrow Initials,Page,Comment

{
\loop{ \newrow {\phantom{\bigletter A}},{},{} }
\ifnum\n < 15
\advance \n by 1
\repeat
}

\vskip\baselineskip
\ecompbox

\oldcomments

\newcomments

\bye
```

```
.. compg> vi gencom.tex
```

```
\nopagenumbers
\vsize = 10.5 true in

\font\bigletter=cmr10 scaled \magstep3
\def\note#1{\vbox{\font\bigbold=cmb10 scaled \magstep0
\font\bigmath=msam10 scaled \magstep0
\bigbold {\bigmath F}
#1
\vskip .2\baselineskip
}}

\def\boxit#1{\vbox{\hrule\hbox{\vrule\kern8pt\vbox{\kern8pt{#1}\kern8pt}\enskip\kern 8pt\vrule}\hrule}}

\def\cbox#1{
\vbox{
```

```
\null\hfill
\boxit{
\vskip 6pt
\advance\hsize by -.1\hsize
{#1}
\vskip 6pt}%boxit Comments
\hfill\null
}
}% end cbox

\def\oldcomments{\cbox{
\hfill Preparation Comments \hfill\break
\hfill\null
}}

\def\newcomments{\cbox{
\hfill Comments for Page 1 \hfill\break
\vbox{\vglue 1.5 true in}
}}

\centerline{\bigletter June 2004}

{\bigletter General\hfill}
\vskip\baselineskip

\newcount\n
\n = 1

\def\newrow#1,#2,#3 {\halign {
\strut\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .1 \hsize{##\hfill}
&\vrule##
&\hbox to .8 \hsize{##\hfill}
&##\vrule\cr
\noalign{\hrule}
&{#1}&&{#2}&&{#3}&\cr
\noalign{\hrule}
}
}

\newrow Initials,Page,Comment

{
\loop{ \newrow {\phantom{\bigletter A}},{},{} }
\ifnum\n < 19
\advance \n by 1
\repeat
}
\oldcomments

\newcomments

\bye
```

4) Run "tamu_make" command.

```
.. compg > ./tamu_make mm yy
```

5) Print comment sheet.

```
.. compg > ./tamu_print
```

To print in other printers, change printer name (-Plj052) by editing "tamu_print" script.

```
> vi tamu_print
#!/usr/local/bin/perl

open (SITELIST, "./tamsite") || die "Couldn't open sitelist, stopped";

$range = 0;
$debug = 0;
$retex = 0;

for $i (1..$#ARGV) {
  if ($ARGV[$i] eq "-t") { $retex = 1; }
  if ($ARGV[$i] eq "-r") {
    $range = 1;
    $from = $ARGV[++$i];
    $to = $ARGV[++$i];
  }
}

print "Printing comments for ";
if ($range) {
  print "sites $from to $to.\n";
} else {
  print "all sites.\n";
}
$i = 0;
while(<SITELIST>) {
  ($siteList[$i], $siteDescript[$i]) = split(/ /);
  if ($range) {
    if ($siteList[$i] eq $from) {
      $fromElement = $i;
    }
    if ($siteList[$i] eq $to) {
      $toElement = $i;
    }
  }
  $i++;
}

if (!$range) {
  $fromElement = 0;
  $toElement = $#siteList;
}

for ($i = $fromElement; $i <= $toElement; $i++) {
  sleep 1;
  system("lpr -Plj052 ./$siteList[$i]com.ps\n");
}
```

4 DESCRIPTION OF COMPONENTS IN THE MECR

4.1 Directories and Files Associated with MECR Production

Figure 4.1 shows the hierarchy of the directories associated with MECR production. There are two main directories ('User_local' and 'Prod') related to MECR Production under the root (lvk4173@lstarasp: /). The 'BIN' sub-directory under 'imecr' directory contains the original files (e.g., imecr.*) for the "imecr" script. The 'imecr' reads the corresponding data into file from ESL database (RAW / *.*) and generate each page with graphs in MECR. Production directory (.../PROD /) contains the directory for raw data files and the files associated with MECR and AECR production. The MECR directory is composed of five sub-directories, including COMPG_SRC, MARKBADS, LSTAR, TAMU, and ESL folders. TAMU and ESL directories also have two sub-directories. One is for comment sheets; the other is for monthly working directories related to each month MECR production. This chapter describes the directories and the files related to MECR production.

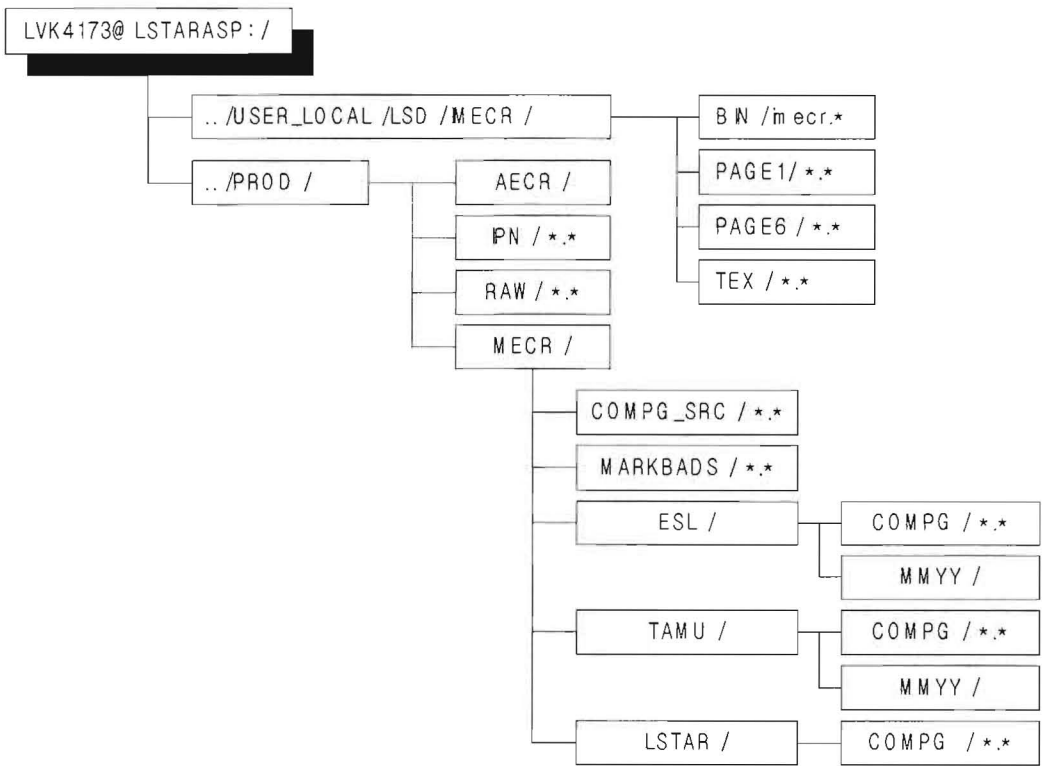


Figure 4.1 Hierarchy of the Directories Associated with MECR Production

4.1.1 Raw Data Files in the "raw" Directory

The sub-directory "raw" of the "prod" directory (lvk4173@lastaraxp:/prod/raw/ *.*) contains all the raw data loaded to the ESL database. The following script box shows an example of the raw data files for the sites starting with number "9" that were loaded to the ESL database.

```
lvk4173@lstaraxp:/prod/raw> ls 9*.*
93203133.raw_loaded  95902239.raw_loaded  99702239.raw_loaded
94102239.raw_loaded  95903133.raw_loaded  99703084.raw_loaded
94602239.raw_loaded  98602239.raw_loaded  99703133.raw_loaded
94703049.raw_loaded  98603133.raw_loaded
```

4.1.2 Files in a Monthly Working Directory

The files needed and generated in ESL MECR production are showed in the following script box. Files in bold text (eslsite, load_draft1, and draft1) are the files copied from the last month directory, which need to be updated to the current month and year for producing the MECR. "load_draft1" and "draft1" are execution files. The other files are generated during the production process for site 938. Table 1.1 shows the description of files associated with producing August 04 MECR for site #938 in the working directory 0804.

```
lvk4173@lstaraxp: / prod/mecr/esl/0804> ls *.*
eslsite                    load_draft1                draft1
Report.938.0804.dvi        Report.938.0804.log
Report.938.0804.ps         Report.938.0804.tex
rpt_cons.938.0804          rpt_cons.938.0804.ctrl     rpt_cons.938.0804.ctrl.acs
rpt_cons.938.0804.ctrl.log rpt_sav.938.0804
s2.938.0803b.sas           s2.938.0804a.ps
s3.938.0804a.ps            s3.938.0804b.ps
s4.938.0804a.ps            s4.938.0804b.ps
s5.938.0804a.ps            s5.938.0804b.ps
...
```

Table 4.1 Description of the Files Associated with Producing August 04 MECR for Site #938

Items	File names	Descriptions	Remarks
Original Files	**eslsite**	Site list	Text File
	load_draft1	Batch file for executing eslsite, rpt_cons, rpt_sav, and draft1 files	Execution Files
	draft1	Batch file for generating eslmecr	
	rpt_cons.938.0804	Energy consumption data file	Savings files
	rpt_cons.938.0804.ctrl	Control File for "getdatc"	
	rpt_cons.938.0804.ctrl.acs	Output from "getdatc"	
	rpt_sav.938.0804	Savings file	
Generated Files unzipped	Report.938.0804.tex	Text source file	Report Files
	Report.938.0804.dvi	Dvi source file from "tex" file	
	Report.938.0804.log	Log of "tex" to "dvi" translation	
	Report.938.0804.ps	Postscript file from "dvi"	
	s2.938.0803b.sas	Sas file for the second graph on page 2	SAS file
	s2.938.0804a.ps	Postscript files for each page in the site938 Where, S(page#).site#.(mmyy)graph#.extension	Page #2 file in a report

4.1.3 Files in the "compg" directory

The files in folder 'compg' are related to the comment sheet of ESL MECR. The following script box shows the list of files in "compg" folder. Files in bold text are the original files saved in this directory. Others are the related files generated in the process to make comment sheets. Table 4.2 shows the description of the typical files in the "compg" directory.

```
Lvk4173@lstaraxp: / prod/mecr/esl/compg> ls *.*
```

1004com.ps	272com.ps	938.comments	955com.log
1004com.tex	272com.tex	938.ecomp	955com.ps
1005.comments	273.comments	938com.dvi	955com.tex
1005.ecomp	273.ecomp	938com.log	956.comments
1005com.dvi	273com.dvi	938com.ps	956.ecomp
1005com.log	273com.log	938com.tex	956com.dvi
1005com.ps	273com.ps	940.comments	956com.log
1005com.tex	273com.tex	940.ecomp	956com.ps
1006.comments	274.comments	940com.dvi	956com.tex
1006.ecomp	274.ecomp	940com.log	957.comments
1006com.dvi	274com.dvi	940com.ps	957.ecomp
1006com.log	274com.log	940com.tex	957com.dvi
1006com.ps	274com.ps	947.comments	957com.log
1006com.tex	274com.tex	947.ecomp	957com.ps
143.comments	275.comments	947com.dvi	957com.tex
143com.tex	275.ecomp	947com.log	**bend.awk**
146.comments	275com.dvi	947com.ps	**commtable.tex**
146.ecomp	275com.log	947com.tex	**gencom.dvi**
146com.dvi	275com.ps	951.comments	**gencom.log**
146com.log	275com.tex	951.ecomp	**gencom.ps**
146com.ps	276.comments	951com.dvi	**gencom.tex**
146com.tex	276.ecomp	951com.log	**loan_print.old**
158com.tex	276com.dvi	951com.ps	**lstar_compg.tar**
Ecom_make	**ecom_print**	**ecom_start**	**ecomp2**

Table 4.2 Description of files in the "compg" directory for comment sheets

Files	File names	Descriptions	Remarks
Original Files	957.comments	A comment sheet for site #957	Text source Files
	commtable.tex	Text source file	
	gencom.tex	Text source file	
	Ecom_make	Create text based comments with "bend.wak"	Execution Files
	ecom_start	Clean up subdirectories	
	ecom_print	Print the text files	
	bend.awk	Process of creating comments	
	ecomp2		
	loan_print.old	Legacy program called 'loan-print'	
	lstar_compg.tar	'Tar' type of zipped comment directory	
Generated Files (Unzipped)	gencom.dvi	'Dvi' source file from 'tex'	Report Files
	gencom.log	Log of 'tex' to 'dvi' translation	
	gencom.ps	Postscript file from 'dvi'	
	957.ecomp		
	957com.tex	'Dvi' source file from 'tex'	
	957com.dvi	Log of 'tex' to 'dvi' translation	
	957com.log		
	957com.ps	Postscript file from 'dvi'	

4.2 Flow Chart of Producing the ESL MECR

Figure 4.2 shows the flow chart of the ESL MECR production. The "Load_draft1" is the first script, which runs the first draft of MECR production. When the draft1 runs, it executes the script 'imecr'. The 'imecr' reads the corresponding data into file from the database and generate the Monthly Energy Consumption Report (MECR) including graphs in each page using SAS. At that time, SAS makes the temporary file such as the position of label in graph or the SAS file for making the graph. Figure 4.3 shows an example of the postscript file for the site #947 MECR. Boxes in red line indicate the energy consumption data and savings data generated by executing 'rpt_cons' file and 'rpt_sav' file, respectively.

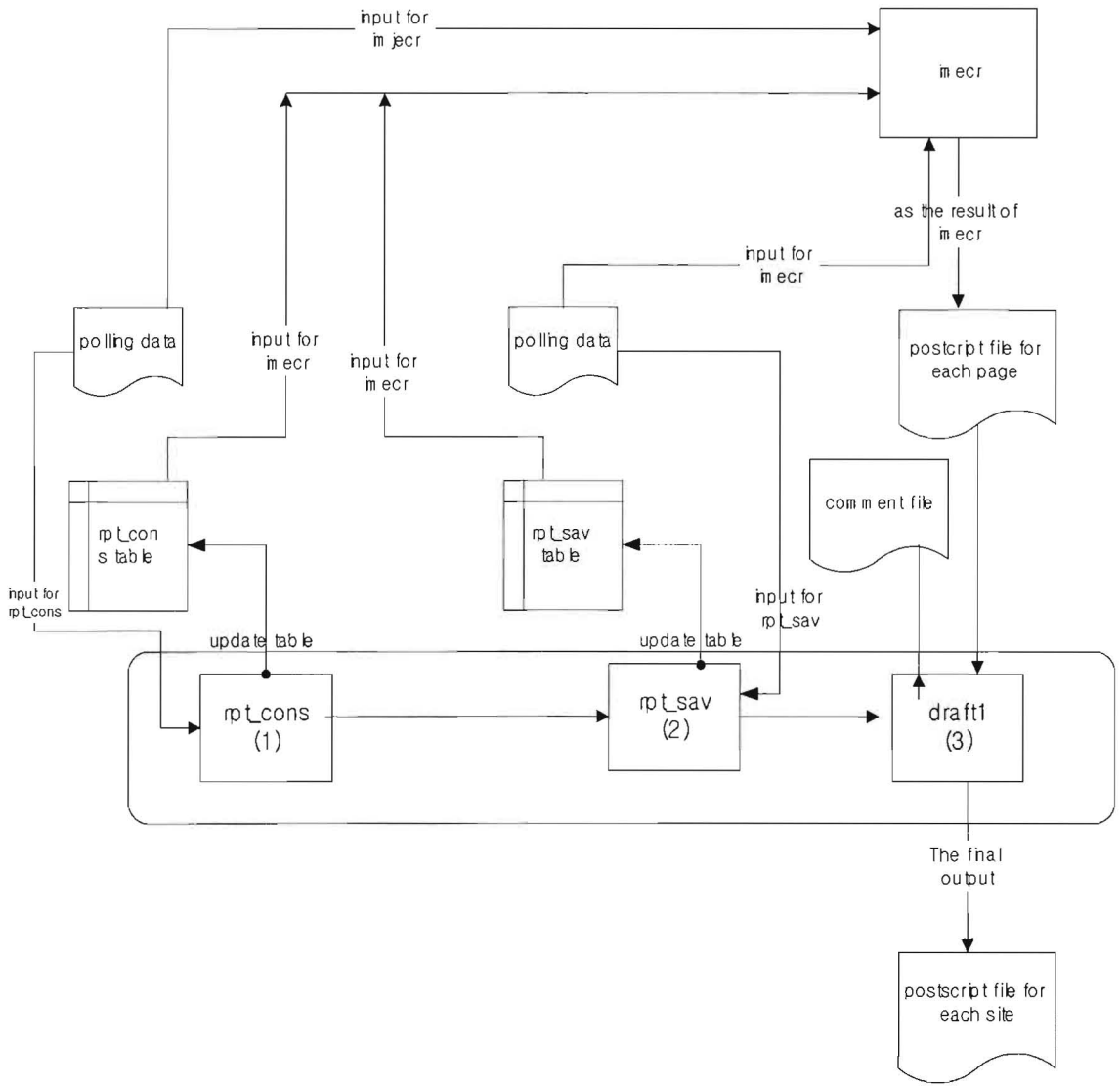


Figure 4.2 Flow Chart of Producing the ESL MECR

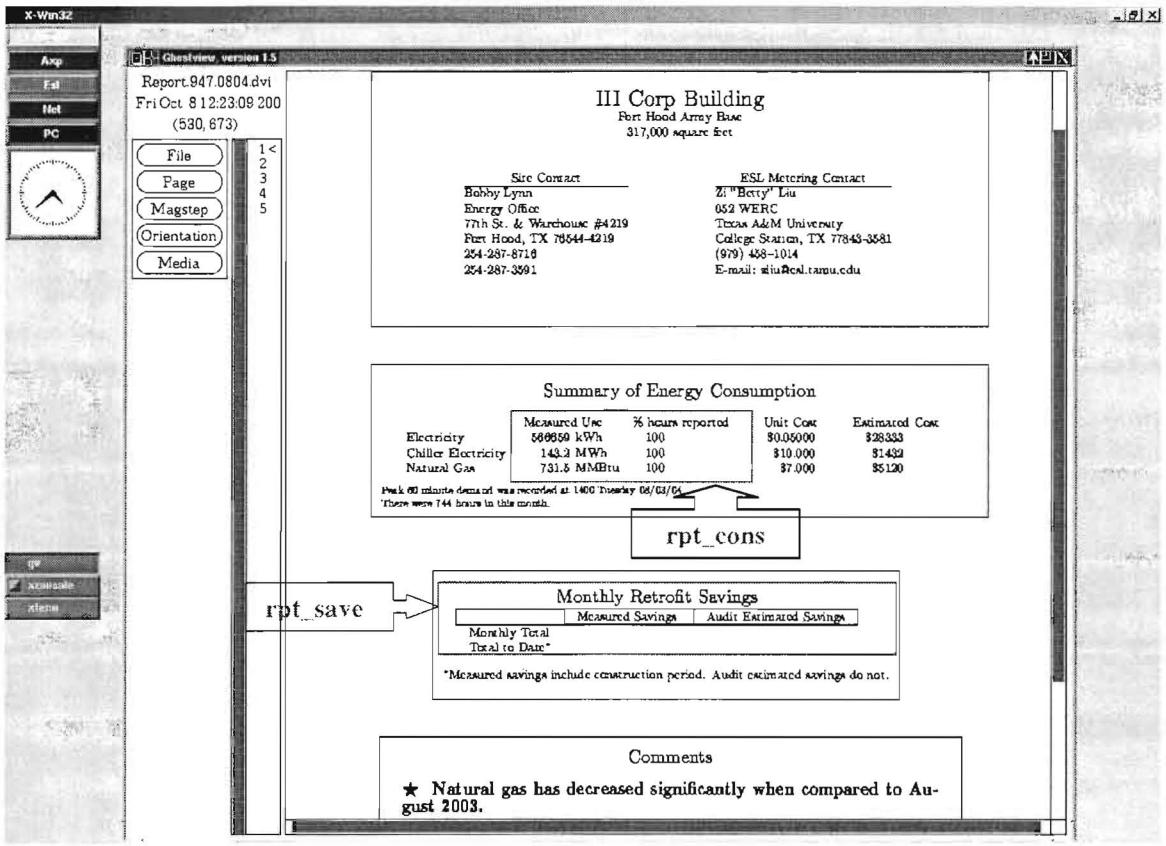


Figure 4.3 An example of the postscript file (Report .947.0804.ps) showing the cover page of the MECR (site 947) for August of 2004 (0804).

4.2.1 Load_Draft1

The Load_Draft1 is composed of three files including Rpt-cons, Rpt-save, and Draft1, which execute the procedures sequentially as shown in Figure 4.2. C shell script controls the whole process of working in MECR. The C shell script of the load_draft1 and the draft1 are shown in the following script boxes showing the sequence of executing the related files.

0804> **vi load_draft1**

```
#! /bin/csh
echo "started:  `date`" > 0804.time
  rpt_cons 08 04 eslsite
  rpt_sav  08 04 eslsite
echo "finished:  `date`" >> 0804.time
echo "started:  `date`" > 0804.mecr
  ./draft1
echo "finished:  `date`" >> 0804.mecr
```

0804> **vi draft1**

```
#! /bin/csh
foreach i (`gawk '{print $1}' eslsite`)
  eslmecr $i 08 04
  rm imecr.$i.0804.imraw
  gzip -f *.$i.*
end
```

Each component of the load_draft1 is described in the following:

1) RPT_CONS

RPT_CONS gets a full months' worth of data for the site using 'getdatc'. The result of reading the data from database is saved in the file rpt_cons.sitenumber.monthyear.gz Then it calculates the energy consumption value by reading the data from the file. It updates or inserts rpt_cons and rpt_cons_dmd columns in the database table using 'load_rpt_cons' (embedded SQL using C)

Table name which is accessed in rpt_cons
Mecr_energy
Site_expr
s1datainfo
rpt_cons
rpt_cons_dmd
Mecr_lock

2) RPT_SAVE

RPT_SAVE gets a full month's worth of data for the site using 'getdatc'. The result of reading the data from database is saved in the file rpt_save.sitenumber.monthyear.gz Then it calculates the energy saving value by reading the data from the file. It updates or inserts rpt_sav and rpt_sav_dmd columns in the database table using 'load_rpt_sav' (embedded SQL using C)

Table name which is accessed in rpt_save
Mecr_energy
site_expr
s1datainfo
rpt_sav
rpt_sav_dmd
mecr_lock

4.2.2 DRAFT1

As shown in Figure 4.2, the draft1 executes the script 'imecr'. The imecr (script) collects the data using getdatc and generates the postscript file of each site using various tools such as SAS and Tex. In 'imecr' each page is generated as the form 's1(pagenumber).site.....a.ps'. using 'new_make_tex', output 'Report.site....tex' is generated. And the Report.site.....ps file is finally produced. An example of the files (imecr.*) generated by 'imecr' is also shown in the following script box, which contains the files related to the site #938 of the ESL MECR, including the control file, the data file, and the page files.

```
lvk4173@lstaraxp: / prod/mecr/esl/0804> ls imecr.*
imecr.938.0604.ctrl      imecr.938.0604.pg2     imecr.938.0604.pg4b
imecr.938.0604.ctrl.acs  imecr.938.0604.pg3     imecr.938.0604.pg5
imecr.938.0604.ctrl.log  imecr.938.0604.pg4a
```

4.3 C Shell Script of the IMECR

```
#!/bin/csh
# Y2k compliant by Nasir Abbas

set bin_dir = /usr/local/lsd/imecr/bin
unset noclobber

##########################################################################
# GRAB COMMAND LINE ARGUMENTS AND INITIALIZE VARIABLES
##########################################################################

#
# asdsaThe option to select specific page numbers has been discontinued.  Previously
# you could specify page 1-5 or all.  While the code still accounts for those
# options, only "all" is permitted.
#
set page  = all

#
# Usage: imecr site month year type [debug_mode]
# where debug_mode is:
#    0 = debug mode off (default)
#    1 = debug mode on
#
set DEBUG = 0
if ($#argv == 5) then
  set DEBUG = $5
else if ($#argv != 4) then
  echo "Usage: imecr site month year type [debug_mode]"
  echo ""
  echo "      where type      is 0 for ESL"
  echo "                      1 for LoanStar"
  echo "      and   debug_mode is 0 for off (default)"
  echo "                      1 for on"
  exit
endif

@ site = $1 + 0
if ( $site < 100 ) then
 if ( $site < 10 ) then
  set sss = "00"$site
 else
  set sss = "0"$site
 endif
else
 set sss = $site
endif

@ month = $2 + 0
if ( $month < 10 ) then
 set mm = "0"$month
else
 set mm = $month
endif

# Convert 1996 to 96 or leave alone if just 96 used.
@ year = $3 + 0
if ( $year > 1900 && $year < 2000) then
  @ year = $year - 1900
```

```
else if ( $year >= 2000) then
   @ year = $year - 2000
endif

if ( $year > 79) then
   @ stop_century = 19
else
   @ stop_century = 20
endif
@ start_century = $stop_century

if ( $year == 0) then
  @ start_century = 19
endif

if ( $year == 99 && $mm == 12 ) then
  @ stop_century = 20
endif

if ( $year < 10 ) then
 set yy = "0"$year
else
 set yy = $year
endif

set type = $4

###############################################################################
# START IMECR
###############################################################################
#
# Get full data set from getdatc into imecr.sss.mmyy.imraw
#

echo Creating imecr for site $sss for $mm/$yy
#
# 1. Set start and stop dates (month, year)
#
@ strt_mm = $mm
@ strt_yy = $yy - 1

if ( $yy < 1 ) then
  @ strt_yy = 99
endif

if ($mm == 12) then
   @ stop_mm = 1
   @ stop_yy = $yy + 1
else
   @ stop_mm = $mm + 1
   @ stop_yy = $yy
endif

if ( $strt_yy < 10 ) then
 set strt_yy = "0"$strt_yy
endif

if ( $strt_yy == 100 ) then
 set strt_yy = "00"
endif
```

```
if ( $stop_yy < 10 ) then
 set stop_yy = "0"$stop_yy
endif

if ( $stop_yy == 100 ) then
 set stop_yy = "00"
endif


#
# 2. Create control file for getdatc.
#
cat > imecr.$sss.${mm}${yy}.ctrl << EOF1
$start_century$strt_yy-$strt_mm-01 00
$stop_century$stop_yy-$stop_mm-01 00
wb $sss
EOF1

#
# 3. Use getdatc to get a full 13 months worth of data for the site.  Then
#    strip off first two lines (start/stop dates) from acs file.  If imraw
#    file already exists, then just use it and skip the call to getdatc, etc.
#
if (-e imecr.$sss.${mm}${yy}.imraw && `$bin_dir/15or60 $sss` == 60) then
  echo "  Using existing imraw file - skipping getdatc"
else
  getdatc imecr.$sss.${mm}${yy}.ctrl
  tail +2 imecr.$sss.${mm}${yy}.ctrl.acs > imecr.$sss.${mm}${yy}.imraw
endif

#
# Convert 15 minute imraw file to 60 minute imraw file if necessary
#
if (`15or60 $sss` == 15) then
  15to60 imecr.$sss.${mm}${yy}.imraw
endif

if ($DEBUG == 0) then
  rm -f imecr.$sss.${mm}${yy}.ctrl
  rm -f imecr.$sss.${mm}${yy}.ctrl.log
  rm -f missing.log
  rm -f imecr.$sss.${mm}${yy}.ctrl.acs
endif

###############################################################################
# PAGE 1
###############################################################################

# Do nothing.  Page 1 info is now contained in rpt_cons and rpt_sav tables.

###############################################################################
# PAGE 2
###############################################################################
if ($page == 2 || $page == all) then
  #
  # 1. Set start and stop dates (month, year)
  #
  @ strt_mm = $mm
  @ strt_yy = $yy - 1

if ( $yy < 1 ) then
  @ strt_yy = 99
```

```
endif

  if ($mm == 12) then
    @ stop_mm = 1
    @ stop_yy = $yy + 1
  else
    @ stop_mm = $mm + 1
    @ stop_yy = $yy
  endif

if ( $strt_yy < 10 ) then
 set strt_yy = "0"$strt_yy
endif

if ( $strt_yy == 100 ) then
 set strt_yy = "00"
endif


if ( $stop_yy < 10 ) then
 set stop_yy = "0"$stop_yy
endif

if ( $stop_yy == 100 ) then
 set stop_yy = "00"
endif


  #
  # 2. No getdatc_extract is needed as page 2 uses the full 13 months of data.
  #
  cp imecr.$sss.${mm}${yy}.imraw imecr.$sss.${mm}${yy}.pg2

  #
  # 3. Create page2[a,b] input file. WARNING, page 2 is a tangled mess of
  #    file operations. Most sites have two graphs of page 2. On top is
  #    whole building cooling (wbcool) and on bottom is whole building heating
  #    (wbheat). However, a few sites have only one graph (chillers) which is
  #    placed at the top. This difference requires the extraction of
  #    different data columns from the raw acs file. So extract either (1)
  #    timestamp, chillers, wbheat, and drybulb, or (2) timestamp, wbcool,
  #    wbheat, and drybulb.
  #
  if ($sss == 144 || $sss == 145 || $sss == 205 || $sss == 240 || $sss == 241 || $sss == 529 || $sss == 530 || $sss == 970 ||
$sss == 971 || $sss == 972 || $sss == 585) then
    nawk '{print $1,$2,$3,$4,"0",$6,$7,$13,$10,$11}' \
            imecr.$sss.${mm}${yy}.pg2 > pg2.tmp1
  else
    nawk '{print $1,$2,$3,$4,"0",$6,$7,$9,$10,$11}' \
            imecr.$sss.${mm}${yy}.pg2 > pg2.tmp1
  endif
  min_conv pg2.tmp1 60 1440 | minshift -1440 > pg2.tmp2

  @ temp = $stop_yy - 1
  if ($temp < 0) then
    @ temp = 99;
  endif
  if ( $temp < 10 ) then
    set temp = "0"$temp
  endif
```

```
  set breakdate = `datcon $mm 01 $yy 0100`
  set prev_yy   = `datcon $stop_mm 01 $temp`

  npg2avg pg2.tmp2 $breakdate $prev_yy > imecr.$sss.${mm}${yy}.pg2

  #
  # 4. Create s2.sss.mmyya.sas and s2.sss.mmyyb.sas files.
  #    Replace call to PF macro with newpath.  This removes the white box
  #    created for the graph by sas, allowing the page number to appear.
  echo "   Making page 2a"

  page2a $sss $mm $yy
  sas s2.$sss.${mm}${yy}a.sas
  if (-e s2.$sss.${mm}${yy}a.ps) then
    sed -e "s/ PF/ newpath/" s2.$sss.${mm}${yy}a.ps > pg2.tmp3
    mv pg2.tmp3 s2.$sss.${mm}${yy}a.ps
  endif

  echo "   Making page 2b"
  page2b $sss $mm $yy
  sas s2.$sss.${mm}${yy}b.sas

  #
  # Clean up
  #
  if ($DEBUG == 0) then
    rm -f imecr.$sss.${mm}${yy}.pg2
    rm -f pg2.tmp?
    rm -f s2.$sss.${mm}${yy}?.sas
    rm -f s2.$sss.${mm}${yy}?.log
  endif

endif

###############################################################################
# PAGE 3
###############################################################################
if ($page == 3 || $page == all) then
  #
  # 1. Set start and stop dates (month, year)
  #
  @ strt_mm = $mm
  @ strt_yy = $yy

  if ($mm == 12) then
    @ stop_mm = 1
    @ stop_yy = $yy + 1
  else
    @ stop_mm = $mm + 1
    @ stop_yy = $yy
  endif

  if ( $strt_yy < 10 ) then
    set strt_yy = "0"$strt_yy
  endif


  if ( $stop_yy < 10 ) then
    set stop_yy = "0"$stop_yy
  endif

  if ( $stop_yy == 100 ) then
```

```
    set stop_yy = "00"
   endif

   #
   # 2. Extract single month of data (timestamp, wbheat, and wbcool) from
   #    imecr.xxx.mmyy.imraw into imecr.xxx.mmyy.pg3.
   #

   set dec_strt = `datcon -odec $strt_mm 01 $strt_yy 0100`
   set dec_stop = `datcon -odec $stop_mm 01 $stop_yy 0000`

   getdatc_extract $dec_strt $dec_stop \
            imecr.$sss.${mm}${yy}.imraw imecr.$sss.${mm}${yy}.pg3

   nawk '{print $6,$9,$10}' imecr.$sss.${mm}${yy}.pg3 > pg3.tmp
   mv pg3.tmp imecr.$sss.${mm}${yy}.pg3

   #
   # 3. Create s3.sss.mmyya.sas and s3.sss.mmyyb.sas files.
   #

   echo "   Making page 3a"
   page3 $sss $mm $yy a
   sas s3.$sss.${mm}${yy}a.sas

   echo "   Making page 3b"
   page3 $sss $mm $yy b
   sas s3.$sss.${mm}${yy}b.sas

   if ($DEBUG == 0) then
     rm -f imecr.$sss.${mm}${yy}.pg3
     rm -f s3.$sss.${mm}${yy}?.sas
     rm -f s3.$sss.${mm}${yy}?.log
   endif
endif

###############################################################################
# PAGE 4
###############################################################################
if ($page == 4 || $page == all) then

   echo "   Making page 4a"
   #
   # 1. Extract single month of data from imecr.xxx.mmyy.imraw into
   #    imecr.xxx.mm.yy.pg4a and imecr.xxx.mm.yy.pg4b
   #

   # Set start data (month, year)
   @ strt_mm = $mm
   @ strt_yy = $yy

   # Set stop date (month, year)
   if ($mm == 12) then
     @ stop_mm = 1
     @ stop_yy = $yy + 1
   else
     @ stop_mm = $mm + 1
     @ stop_yy = $yy
   endif

   if ( $strt_yy < 10 ) then
     set strt_yy = "0"$strt_yy
```

```
  endif


  if ( $stop_yy < 10 ) then
    set stop_yy = "0"$stop_yy
  endif

  if ( $stop_yy == 100 ) then
    set stop_yy = "00"
  endif

  #
  # Extract imecr.sss.mm.yy.pg4a data (timestamp, wbele, and channel data)
  # from imraw file.  Program page4a creates file pg4.tmp2
  #
  set dec_strt = `datcon -odec $strt_mm 01 $strt_yy 0000`
  set dec_stop = `datcon -odec $stop_mm 01 $stop_yy 0000`

  getdatc_extract $dec_strt $dec_stop \
                        imecr.$sss.${mm}${yy}.imraw imecr.$sss.${mm}${yy}.pg4a

  nawk -f $bin_dir/page4a.awk imecr.$sss.${mm}${yy}.pg4a > pg4.tmp1
  mv pg4.tmp1 imecr.$sss.${mm}${yy}.pg4a

  page4a $sss $mm $yy
  sas s4.$sss.${mm}${yy}a.sas

  echo "   Making page 4b"
  #
  # Extract imecr.sss.mm.yy.pg4b data (timestamp, drybulb, and relative
  # humidity) from imraw file.
  #
  set dec_strt = `datcon -odec $strt_mm 01 $strt_yy 0100`
  set dec_stop = `datcon -odec $stop_mm 01 $stop_yy 0000`

  getdatc_extract $dec_strt $dec_stop \
                        imecr.$sss.${mm}${yy}.imraw imecr.$sss.${mm}${yy}.pg4b

  nawk '{print $6, $11, $12}' imecr.$sss.${mm}${yy}.pg4b > pg4.tmp1
  mv pg4.tmp1 imecr.$sss.${mm}${yy}.pg4b

  page4b $sss $mm $yy
  sas s4.$sss.${mm}${yy}b.sas
  echo why
  if ($DEBUG == 0) then
    rm -f imecr.$sss.${mm}${yy}.pg4a
    rm -f imecr.$sss.${mm}${yy}.pg4b
    rm -f pg4.tmp1
    rm -f pg4.tmp2
    rm -f s4.$sss.${mm}${yy}?.sas
    rm -f s4.$sss.${mm}${yy}?.log
  endif
endif

################################################################################
# PAGE 5
################################################################################
if ($page == 5 || $page == all) then
  #
  # 1. Set start and stop dates.  This procedure is a little different from
  #    the other pages in that the stop date should be the last day of the
  #    current month (e.g., "imecr 07 95" should have a start date of
```

```
#    "1995-06-01 00" and a stop date of "1995-07-31 23"
#

if ($mm == 01) then
   @ strt_mm = 12
   @ strt_yy = $yy - 1
         if ( $yy < 1 ) then
                  @ strt_yy = 99
         endif
else
   @ strt_mm = $mm - 1
   @ strt_yy = $yy
endif

 if ( $strt_yy < 10 ) then
   set strt_yy = "0"$strt_yy
 endif

set dec_strt = `datcon -odec $strt_mm 01 $strt_yy 0000`

# Get Julian date for first day of next month
if ($mm == 12) then
   @ next_mm = 1
   @ next_yy = $yy + 1
else
   @ next_mm = $mm + 1
   @ next_yy = $yy
endif

if ( $next_yy < 10 ) then
   set next_yy = "0"$next_yy
endif

if ( $next_yy == 100 ) then
   set next_yy = "2000"
endif

set jul_next_mm = `datcon -ojul $next_mm 01 $next_yy 0000`

#
# Subtract one from first day of next month to get last day of current
# month, then convert that julian date to decimal.
#

 @ last_day_mm = $jul_next_mm[1] - 1

if ($last_day_mm < 10) then
  set last_day_mm = "200000"$last_day_mm
else if ($last_day_mm < 100) then
  set last_day_mm = "20000"$last_day_mm
else if ($last_day_mm < 1000) then
  set last_day_mm = "2000"$last_day_mm
else
  set last_day_mm = $last_day_mm
endif


 set dec_stop = `datcon -odec $last_day_mm 2300`

 #
 # 2. Extract single month of data from imecr.xxx.mmyy.imraw into
 #    imecr.xxx.mm.yy.pg5
```

```
  #
  #
   getdatc_extract $dec_strt $dec_stop \
                        imecr.$sss.${mm}${yy}.imraw imecr.$sss.${mm}${yy}.pg5

  nawk '{print $6,$7,$8}' imecr.$sss.${mm}${yy}.pg5 > pg5.tmp


  mv pg5.tmp imecr.$sss.${mm}${yy}.pg5

  echo "   Making page 5a"
  page5a $sss $mm $yy
  sas s5.$sss.${mm}${yy}a.sas

  # Remove mysterious "-5265" like numbers from PostScript file
  sed -e "s/\([--][0-9][0-9][0-9][0-9]\)//g" s5.$sss.${mm}${yy}a.ps > pg5.tmp
  mv pg5.tmp s5.$sss.${mm}${yy}a.ps

  echo "   Making page 5b"
  page5b $sss $mm $yy
  sas s5.$sss.${mm}${yy}b.sas

  # Remove mysterious "-5265" like numbers from PostScript file
  sed -e "s/\([--][0-9][0-9][0-9][0-9]\)//g" s5.$sss.${mm}${yy}b.ps > pg5.tmp
  mv pg5.tmp s5.$sss.${mm}${yy}b.ps

  if ($DEBUG == 0) then
    rm -f imecr.$sss.${mm}${yy}.pg5
    rm -f s5.$sss.${mm}${yy}?.sas
    rm -f s5.$sss.${mm}${yy}?.log
  endif
endif

###############################################################################
# CREATE FINAL REPORT
###############################################################################
echo "   Creating TeX file and final report"
new_maketex $sss $mm $yy $type

#if ($sss >= 160 && $sss <= 164) then
#   nawk '{sub(/Peak 60/,"Peak 15"); print $0}' Report.$sss.${mm}${yy}.tex > \
#      R.$$tmp.tex
#   mv R.$$tmp.tex Report.$sss.${mm}${yy}.tex
#endif

tex \\batchmode \\input Report.$sss.${mm}${yy}.tex \\end
dvips Report.$sss.${mm}${yy}.dvi

if ($DEBUG == 0) then
  rm -f Report.$sss.${mm}${yy}.dvi
  rm -f Report.$sss.${mm}${yy}.log
endif

###############################################################################
# CLEAN UP
###############################################################################
if ($DEBUG == 0) then
  rm -f date.log
endif
```

APPENDIX A PROGRAMS USED FOR MECR PRODUCTION

This section outlines the programs and scripts that produce the Monthly Energy Consumption Reports (MECR) for the programmer who may have to modify the code.

B1. An Overview of MECR Programs

This section lists the major programs that are called to create the MECR.

command line: imecr site mm yy

This starts a csh script and launches a series of programs to create the MECR:

Program		Descriptions
No.	Names	
1	Mk1dat	Prepares usage table for page 1 of the MECR.
2	Do2	Creates postscript daily energy vs temperature plots which are found on page 2 of the MECR
3	Do3	Creates postscript time series thermal plots found on page 3 of the MECR.
4	Do4a	Produces postscript time series electrical plots (with submetering).
5	Do4b	Creates postscript time series weather plots.
6	Do5	Makes postscript 3D time series electric plots.
7	maketex	Creates *report.sss.mmyy.tex* - the TeX shell that knits the plots together.
8	Tex	Converts *report.sss.mmyy.tex* to **.dvi* (a TeX document).
9	dvips	Converts **.dvi* to **.ps*.

1. mk1dat

This esql/C (embedded sql/C) program creates the Summary of Energy Consumption box on the first page of the MECR.

command line: mks1dat sss mm yy

1) Input: Mk1dat receives two inputs:

1	wb_sss	This is a table in the LoanSTAR database containing whole building data for one site. From this table, mk1dat obtains: timestamp, wbele (whole building electric), whole building cooling (wbcool), and whole building heat (wbheat) in hourly format.
2	s1datinfo	From this loanSTAR database table, mk1dat gets labels, units and costs: chill water label and units, hot water label and units, electricity costs, demand costs, chill water costs, and hot water costs.

2) Output : Mk1dat converts the daily data to monthly data and places it in the file s1.sss.mmyy.dat. This creates a table which contains: sss site number

	measured use	%hours reported	unit cost	estimated cost
electricity row				
demand row				
cooling row (if present)				
heating row (if present)				

This is a file that TeX can use to generate the Summary of Energy Consumption on the first page of the MECR.

2. do2

This second program in the MECR production is a csh script which creates the second page of plots. The second page contains 13 month-long crossplots of daily thermal (sometimes submetered electrics) versus outdoor dry-bulb temperature.

command line:do2 sss mm yy [debug_flag]

If debug flag is 1, then the do2 prints to the screen which programs it is calling, and it leaves all the log and temporary files after the run.

1	datcon	Calculates the decimal dates of the beginning of the month a year ago and the end of the current month.
2	ngetforpage2	Queries the data base table wb_sss to see if the right date-range exists. If it does, it retrieves the following data from wb_sss: timestamp, wbcool, wbheat, oadrybulb. If it cannot find data in the time interval specified, it calls the program genwb to create a database table that has data for the correct time interval. (This program will soon be replaced by getdatc). The timestamps are changed to decimal by datcon and the program is run through awk to create s2.sss.mmyy.inp. This file contains these columns: 0 mm dd yy 0 decimal date hhmm cooling heating oadb in hourly format.
3	missing	Occasionally there are gaps in the data records due to logger failure. Missing creates new records to fill in the gaps. All data points in these new records contain the no-data flag -99. This creates *.tmp4 as output.
4	minconv	This program converts hourly data to daily data, and creates *.tmp5 as output.
5	minshift	Backs the timestamps up by one day, since converting the hourly data to daily data shifts the times forward. This occurs because hourly data are stamped with the hour after their collection. MinShift creates *.tmp6 as output.
6	npg2avg	Assigns appropriate symbols for plotting data points: m,t,w,r,f,s,u for this month's data points + for this month last year * for all other points and filters out partial days. It then takes the totals for thermal channels and the averages for daily outside air dry-bulb temperature. Creates *inp.
7	nawk sheet2a.awk	Where "a" specifies the top plot on page 2. This awk script calls maketitle, an awk script that scans the *.inp file for the first and last decimal dates and the largest consumption value. It outputs the start and end date in a title string and the maximum size for the left axis in the file title. Then the script receives as input sheet2a.src, which contains the axis labels, max y axis value, and date string. It creates as output sheet2a.sas, from which SAS creates the plots.
8	nawk sheet2b.awk	Same as above, where b is the bottom plot on page 2. Input: sheet2b.src, title Output: sheet2b.sas
9	SAS	Receives as input sheet2{a, b}.sas and *.inp to create the two plots. The output is s2.sss.mmyy{a, b}.ps, a postscript file.
10	nopf	Removes the default SAS plot frame, which is opaque and would hide the axis labels.
11	Clean up	After the do2 programs are run, the following files are deleted: *.sas, *.log, *.tmp*, *.inp.

3. Do3

This program creates the postscript timeseries thermal plots (chill water and hot water use for the month in question) found on page 3 of the MERC.

command line: do3 sss mm yy [debug_flag]

The debug option here does the same as the one for do2. If the debug flag equals one then do3 writes to the screen which programs it calls and maintains all the files it creates at the end of the run.

1	datcon	Calculates the starting and end dates.
2	getforpg3	sss mm yy Input: wb_sss. Ensures that the right date range is in the table wb_sss. If not, getforpg3 calls genwb which will create the appropriate table. Getforpg3 creates a file s3.sss.mmyyy.tmp1 which contains the timestamp, wbcool, wbheat data. Datcon converts the timestamp to decimal date and creates a file named *.tmp2, and Nawk reorders the columns for loanSTAR standard timestamp which is necessary for the next step and names the file *.inp.
3	missing	Fills in the gaps of missing information and creates *.tmp4
4	nawk	Removes the LoanSTAR timestamp and picks out the decimal date, whole building cooling, and whole building heating columns and places them in a file called *.inp. decimal date wbcool wbheat
5	make3_ctrl alb	Calls area to calculate the right max and right step. Also uses for input pg3labels, a LoanSTAR data table that contains site, graph type (a or b), month-v, year-v, left label, x label, right label. If the month-v and the year-v are NULL, then mm yy is used to create the x axis in the form of Month Year. To determine the maximum and the step of the left axis, the *.inp file is searched for the largest number for the left axis. Then this number is compared in a function called goodsize which finds the next number easily divisible by four (which is the step) and returns that value. The right step is determined by the formula = leftstep/area * 1 x 106. The right max is determined by multiplying this value by 4. No the output of all this is *{a,b}.sas.
6	SAS	Receives as input the file *{a,b}.sas and *inp and creates the plot *{a,b}.ps.
7	nopf	Removes the frame around the plot and the final output for do3 is s3.sss.mmyy.ps.
8	clean up	Removes all the tmp and log files if the debug flag is not present.

4. Do4a

Since the two plots on page 4 are unrelated to each other, the task of creating them is devided into two programs. The upper plot containing submetered electrical information is created by the csh script do4a.

command line: sss mm yy [debug_flag]

1	datcon	Calculates the starting and end dates.
2	getforpg4a	sss mm yy This esql/C program verifies that the right information is in wb_sss. If not, it calls genwb to create the LoanSTAR database table. It then retrieves the submeter names from the table pg4submeters. Writes slave.ec and compiles it. Slave creates s4a.sss.mmyy.tmp1. Datcon gets the decimal dates and creates tmp.2. Nawk rearranges for LoanSTAR timestamp which the program missing needs in order to run and creates *.inp: timestamp whole_building_electric sub1...subn
3	missing	Fills in gaps in the data and creates *.tmp4.
4		Strips out all of the timestamp and just leaves the decimal dates and the electrics columns in

	nawk	the file *.inp: decimal_date wbele sub1 ... subn
5	make4a.ctrl	sss mm yy This esql/C program receives as input *.inp, lsdb table pg4labels and pg4submeters. Runs area. Creates *.ctrl which contains the columns: leftlabel xlabel rightlabel leftmax leftstep rightmax rightstep number_subs subname1...subnamen
6	mkpg4a	sss mm yy This C program receives *.ctrl as input. It writes *.sas which contains the SAS instructions to creates the plot by using the information in *.ctrl.
7	SAS	Receives *.ctrl and *inp as inputs and creates the plots and places them in *.ps.
8	nopf	Removes the plot frame which would obscure the axis labels. The final output for do4a is s4a.sss.mmyy.ps
9	clean up	Removes all the temporary and log files if the debug_flag is not set.

5. Do4b

This csh script creates a plot of the hourly dry bulb temperatures and relative humidity on the lower half of page 4.

command line: sss mm yy [debug_flag]

1	datcon	Calculates the starting and end dates.
2	getforpg4b	sss mm yy Checks to see if the database table wb_sss is present. If not, getforpg4b calls genwb to create it. It then selects outdoor air dry bulb temperatures and relative humidity. The LoanSTAR timestamps are recreated by datcon and nawk, just like for the 4a plot, and the file is named s4b.sss.mmyy.inp.
3	missing	Fills in data gaps and names the file *.tmp4.
4	nawk	Strips all the loanSTAR timestamp information and leaves the decimal dates and the weather information in a file called *.inp.
5	mk4bctrl	sss mm yy This Esql/C program gets the information to create *.ctrl from the database table pg4blabels. The output contains the following columns: leftlabel xlabel 120(leftmax) 10(leftstep) dblabel(dbTemp) humlabel(RH)
6	makepg4b	Receives information to create the SAS instructions from the *.ctrl file. The SAS instructions go into the file *.sas.
7	SAS	Creates the plot of the weather information and puts it into *.ps.
8	nopf	Creates the plot of the weather information and puts it into *.ps.
9	clean up	If the debug_flag is not present, then all the temporary and log files are deleted.

C.2 Brief Description of Program Command Lines

imecr sss mm yy (csh script) creates monthly energy consumption reports

mks1dat sss mm yy (esql c) creates the Summary of Energy Consumption box on page 1
input: wb_sss {lsd table containing wb info for site}
s1datinfo {lsd table with labels, units, costs}
output: s1.sss.mmyy.dat {all info needed to generate table}

do2 sss mm yy [debug] (csh script) creates plots for page 2

datcon gets decimal dates for start and end times

ngetforpg2 ss mm yy retrieves timestamp, wbcool, wbheat, osdb
input: wb_sss [genwb]

datcon changes to decimal date
awk creates data file
output: s2.sss.mmyy.inp

missing fills in data gaps
input: s2.sss.mmyy.inp
output: *.tmp4

minconv converts hourly data to daily data
input: *.tmp4
output: *.tmp5

minshift shifts timestamps up one day
input: *.tmp5
output: *.tmp6

npg2avg assigns appropriate symbols, filters out partial days, totals thermal channels and averages dry bulb
input: *.tmp6
output: *.inp

nawk sheet2{a,b}.awk creates labels for plots
input: sheet2{a,b}.src

maketitle (awk script) creates datestring for graph heading, max y value for left y axis
input: *.inp
output: title, goodsize(max)

area sss retrieves area for site
input:
output: sheet2{a,b}.sas

SAS sheet2{a,b}.sas plots info
input: sheet2{a,b}.sas, *.inp
output: s2.sss.mmyy{a,b}.ps

nopf removes SAS plot frame
input: *.ps
output: s2.sss.mmyy.ps

do3 sss mm yy [debug] (csh script) creates plots for page 3

datcon for starting and ending decimal dates

getforpg3 sss mm yy (esql/C) creates 0 mm dd yy 0 ddate hhmm wbcool wbheat
input: wb_sss

datcon generates decimal dates for the wb file
input: s3.sss.mmyy.tmp1
output: *.tmp2

nawk reorders fields for ls std timestamp
input: *.tmp2
output: *inp

missing fills in data gaps
input: s3.sss.mmyy.inp
output: *.tmp4

nawk picks out ddate, wbcool, wbheat columns
input: *.tmp4
output: *.inp

make3_ctrl alb sss mm yy (esql/C) creates file w/ area,left and right max, axis labels
input: *inp, pg3labels

area sss retrieves area for site
input: site table
output: s3.sss.mmyy{a,b}.ctrl

makepg3 alb sss mm yy creates file for SAS plot
input: s3.sss.mmyy{a,b}.ctrl
output: s3.sss.mmyy{a,b}.sas

sas s3.sss.mmyy{a,b}.sas plots data
input: *.sas, *inp
output: *.ps

nopf removes plot frame
input: *.ps
output: s3.sss.mmyy{a,b}.ps

do4a sss mm yy [debug_flag] (csh) creates upper plot of submetered electrics

datcon calculates beginning and ending decimal dates

getforpg4a sss mm yy (esqlc) pulls the correct data from the database
input: wb_sss, pg4submeters

slave (esqlc) pulls the appropriate submeters from the database
input: pg4submeters
output: s4a.sss.mmyy.tmp1

datcon converts the dates to decimal
input: *.tmp1
output: *.tmp2

nawk rearranges for the LoanSTAR timestamp
input: *.tmp2
output: *.inp

missing fills in data gaps
input: *.inp
output: *.tmp4

nawk removes timestamp
input: *.tmp4
output: *.inp

make4a_ctrl sss mm yy (esqlc) creates file to make SAS instruction set
input: *.inp, pg4labels, pg4submeters

area pulls out square footage of the building
output: *.ctrl

mkpg4a sss mm yy (C) creates the SAS instruction set
input: *.ctrl
output: *.sas

SAS creates plot
input: *.sas, *.inp
output: *.ps

nopf removes SAS frame
input: *.ps

output: *.tmp7

mv changes file name
input: *.tmp7
output: s4a.sss.mmyy.ps

do4b sss mm yy [debug_flag] (csh script) plots hourly dry bulb and relative humidity

datcon calculates beginning and ending decimal dates

getforpg4b sss mm yy (esqlc) gets data from the database
input: wb_sss

datcon converts dates to decimal

nawk creates LoanSTAR timestamp
output: s4b.sss.mmyy.inp

missing fills in data gaps
input: *.inp
output: *.tmp4

nawk strips timestamps
input: *.tmp4
output: *.inp

mk4bctrl sss mm yy (esqlc) makes file to create SAS instructions
input: pg4blabels
output: *.ctrl

makepg4b sss mm yy (esqlc) creates SAS instruction set
input: *.ctrl
output: *.sas

SAS makes plot
input: *.sas, *.inp
output: *.ps

nopf removes plot frame
input: *.ps
output: s4b.sss.mmyy.ps

APPENDIX B UNIX COMMAND SUMMARY

Command/Syntax	What it will do
awk/nawk [options] *file*	scan for patterns in a file and process the results
cat [options] *file*	concatenate (list) a file
cd [directory]	change directory
chgrp [options] *group file*	change the group of the file
chmod [options] *file*	change file or directory access permissions
chown [options] *owner file*	change the ownership of a file; can only be done by the superuser
chsh (**passwd -e/-s**) *username login_shell*	change the user's login shell (often only by the superuser)
cmp [options] *file1 file2*	compare two files and list where differences occur (text or binary files)
compress [options] *file*	compress file and save it as *file.Z*
cp [options] *file1 file2*	copy *file1* into *file2*; *file2* shouldn't already exist. This command creates or overwrites *file2*.
cut (options) [*file*(s)]	cut specified field(s)/character(s) from lines in file(s)
date [options]	report the current date and time
dd [if=infile] [of=outfile] [operand=value]	copy a file, converting between ASCII and EBCDIC or swapping byte order, as specified
diff [options] *file1 file2*	compare the two files and display the differences (text files only)
df [options] [resource]	report the summary of disk blocks and inodes free and in use
du [options] [*directory* or *file*]	report amount of disk space in use
echo [text string]	echo the text string to stdout
ed or **ex** [options] *file*	Unix line editors
emacs [options] *file*	full-screen editor
expr *arguments*	evaluate the arguments. Used to do arithmetic, etc. in the shell.
file [options] *file*	classify the file type
find directory [options] [actions]	find files matching a type or pattern
finger [options] *user[@hostname]*	report information about users on local and remote machines
ftp [options] *host*	transfer file(s) using file transfer protocol
grep [options] 'search string' *argument* **egrep** [options] 'search string' *argument* **fgrep** [options] 'search string' *argument*	search the argument (in this case probably a file) for all occurrences of the search string, and list them.
gzip [options] *file* **gunzip** [options] *file* **zcat** [options] *file*	compress or uncompress a file. Compressed files are stored with a .**gz** ending
head [-number] *file*	display the first 10 (or number of) lines of a file

hostname	display or set (super-user only) the name of the current machine
kill [options] [-SIGNAL] [pid#] [%job]	send a signal to the process with the process id number (pid#) or job control number (%n). The default signal is to kill the process.
ln [options] *source_file target*	link the *source_file* to the *target*
lpq [options] **lpstat** [options]	show the status of print jobs
lpr [options] *file* **lp** [options] *file*	print to defined printer
lprm [options] **cancel** [options]	remove a print job from the print queue
ls [options] [*directory* or *file*]	list *directory* contents or *file* permissions
mail [options] [user] **mailx** [options] [user] **Mail** [options] [user]	simple email utility available on Unix systems. Type a period as the first character on a new line to send message out, question mark for help.
man [options] *command*	show the manual (**man**) page for a command
mkdir [options] *directory*	make a *directory*
more [options] *file* **less** [options] *file* **pg** [options] *file*	page through a text file
mv [options] *file1 file2*	move *file1* into *file2*
od [options] *file*	octal dump a binary file, in octal, ASCII, hex, decimal, or character mode.
passwd [options]	set or change your password
paste [options] *file*	paste field(s) onto the lines in *file*
pr [options] *file*	filter the file and print it on the terminal
ps [options]	show status of active processes
pwd	print working (current) directory
rcp [options] *hostname*	remotely copy files from this machine to another machine
rlogin [options] *hostname*	login remotely to another machine
rm [options] *file*	remove (delete) a file or directory (**-r** recursively deletes the directory and its contents) (**-i** prompts before removing files)
rmdir [options] *directory*	remove a *directory*
rsh [options] *hostname*	remote shell to run on another machine
script *file*	saves everything that appears on the screen to file until **exit** is executed
sed [options] *file*	stream editor for editing files from a script or from the command line
sort [options] *file*	sort the lines of the *file* according to the options chosen
source *file* **.** *file*	read commands from the *file* and execute them in the current shell. **source**: C shell, **.**: Bourne shell.
strings [options] *file*	report any sequence of 4 or more printable characters ending in <NL> or <NULL>. Usually used to search binary files for ASCII strings.

stty [options]	set or display terminal control options
tail [options] *file*	display the last few lines (or parts) of a file
tar key[options] [*file*(s)]	tape archiver--refer to man pages for details on creating, listing, and retrieving from archive files. Tar files can be stored on tape or disk.
tee [options] *file*	copy stdout to one or more files
telnet [host [port]]	communicate with another host using telnet protocol
touch [options] [date] *file*	create an empty file, or update the access time of an existing file
tr [options] *string1 string2*	translate the characters in string1 from stdin into those in string2 in stdout
uncompress *file.Z*	uncompress *file.Z* and save it as a file
uniq [options] *file*	remove repeated lines in a file
uudecode [*file*]	decode a uuencoded file, recreating the original file
uuencode [*file*] *new_name*	encode binary file to 7-bit ASCII, useful when sending via email, to be decoded as new_name at destination
vi [options] *file*	visual, full-screen editor
wc [options] [*file*(s)]	display word (or character or line) count for *file*(s)
whereis [options] *command*	report the binary, source, and man page locations for the command named
which *command*	reports the path to the command or the shell alias in use
who or **w**	report who is logged in and what processes are running
zcat *file.Z*	concatenate (list) uncompressed file to screen, leaving file compressed on disk

APPENDIX C VI REFERENCE CARD

PREPEND ALL APPROPRIATE vi COMMANDS BY *n* TO REPEAT *n* TIMES
TYPE (ESC) TO RETURN FROM INPUT TO COMMAND MODE

a [A]	append after cursor [line]	w [W]	word ["Word"]
b [B]	back one word ["Word"]	x [X]	cross out char at [before] cursor
c [C]	change next [to end of line]	y [Y]	yank next [whole line]
d [D]	delete next [to end of line]	z*pos*	redraw zone at *pos* [., -, or (CR)]
e [E]	end of word ["Word"]	ZZ	exit vi, write changes to file
f [F]	find next [previous] in line	^b [^f]	backward [forward] paging
G [*n*G]	Go to last [*n*th] line	^d [^u]	downward [upward] scrolling
h [l]	cursor left [right]	^d [^t]	delete [tab] one *sw* during insert
H [L]	to Home [Last] line on screen	^e [^y]	expose 1 more line at bottom [top]
i [I]	insert before cursor [line]	^h [^w]	back 1 char [word] during insert
j [k]	cursor down [up]	~	change case (upper/lower) of char
J	Join line with next	+ [-]	to first char in next [prev] line
m*x* [‘*x*]	mark [return to] position *x*	0 [$]	to first [last] character in line
M [*n*|]	to Middle line [*n*th column]	; [,]	repeat [reverse] last f, F, t, or T
n [N]	to next [previous] occurrence	.	repeat last change of the text
o [O]	open a line below [above]	<< [>>]	shift line one *sw* left [right]
p [P]	put in after [before]	([)]	to beginning of [next] sentence
Q	Quit vi, go to ex	{ [}]	to beginning of [next] paragraph
r [R]	replace 1 [all] character[s]	/ [?]	search forward [backward]
s [S]	substitute character [line]	‘‘ [’’]	return to previous position [line]
t [T]	to next [previous] in line	! [:!]	execute ex [shell] command
u [U]	undo last change[s in line]	!! [!!]	shell command on next [this line]