NRAO 225 GHz Tipping Radiometer Computer Operation

Scott M. Foster
National Radio Astronomy Observatory
949 N. Cherry Ave.
Tucson, AZ 85721-0655
email: sfoster@nrao.edu

September 24, 1996

Contents

1 WARNING

This document is for internal use only. It contains specific information about how to access the MMA Site Testing computers. Do not distribute outside of NRAO.

2 Introduction

Much has changed since the original radiometer computer operation document was written by Foster (1989). This document attempts to describe the current computer hardware and software configuration in enough detail to allow continued operation and any necessary modifications. The radiometer hardware itself is beyond the scope of this document. For a detailed description of the radiometer hardware, see Liu (1987) or contact Gerry Petencin at the Array Operations Center.

Section 3 describes the hardware setups used to test the Chajnantor, Chile site and, until recently, the Mauna Kea, Hawaii site. Section 4 describes, in detail, the software used to run the radiometer computer. Finally, section 6 provides a detailed description of the current data retrieval software.

3 Hardware Configuration

Two different hardware configurations have seen recent use. Until recently, NRAO operated a radiometer at the VLBA station on Mauna Kea. Since

this site is regularly staffed, a relatively simple setup was used. If a problem developed, a telephone call or e-mail message could have things fixed within hours. In addition, the presence of an Internet connection at that site made data retrieval and communication easy. Although site testing on Mauna Kea was discontinued recently, this type of configuration may eventually find use elsewhere. Therefore, we will also outline the Mauna Kea setup later in this section.

Unlike the Mauna Kea site, the Chajnantor site is very remote and unstaffed. The site has no power, no phone line, and, certainly, no Internet connection. If a problem develops on the Chajnantor site, it takes days or weeks to get someone to the site to check on the equipment. For this reason, we must use a more complicated system to improve reliability.

3.1 Chajnantor, Chile, Hardware Configuration

The Chajnantor site testing equipment is housed in a shipping container on the site. On the north side of the shipping container is an array of solar panels. The solar panels charge a bank of batteries stored inside the container. The batteries, in turn, provide power for the site testing and communications equipment. A windmill provides an additional source of power for battery charging. Communications with the site is accomplished by satellite telephone (but might be replaced by a cellular phone in the near future). The satellite telephone antenna is attached to a mast on one corner of the container. The rest of the satellite telephone hardware is stored inside.

The radiometer sits in a window on the south side of the container with the feed horn sticking out. In addition, data cables run from an external weather station, through a bulkhead connector, to the radiometer to provide information on wind speed and direction. Outside temperature measurements are obtained from a temperature probe which dangles below the feed horn. Finally, inside the container, data from a number of monitor points also connect to the radiometer. Table 1 lists the monitor points currently in use.

Figure 1 shows a block diagram of the Chajnantor site radiometer hardware configuration. We will describe the function of all of this hardware by tracing an incoming call to the container.

At the time of this writing, an outline of the procedures for calling the Chajnantor site is maintained by Simon Radford. The most recent copy of these procedures appears in Appendix A. For more information, see the Saturn CompacT satellite phone documentation. The radiometer hardware

Channel	Source	Monitor Point
0	Internal	Signal - Reference
1	Internal	Hot - Reference
2	Internal	Reference
3	Internal	Reference Temp
4	Internal	Hot Temp
5	Internal	Out Temp
6	Internal	In Temp
7	Internal	Mixer Current
8	Internal	Trippler Current
9	Internal	Gunn Current
10	External	Battery Voltage
11	External	Solar Charge Current
12	Internal	Supply Current
13	External	Wind Speed
14	External	Wind Direction
15	External	Aux Temp

Table 1: Chajnantor Radiometer Monitor Points

connects to the secondary ID of the satellite phone. An incoming call to the secondary ID of the satellite telephone will be answered by the Black Box Corporation ASD-4 (Automatic Sharing Device).

3.1.1 The ASD-4 (Automatic Sharing Device)

The ASD-4 takes one input phone line and routes that signal to one of four output ports. The default port (labeled "Phone" on the ASD-4) routes an incoming call to the modem. The modem is configured to answer an incoming call on the first ring it receives. This means that it actually takes two rings to connect to the PC. The first ring is answered by the ASD-4. The second ring (the first one the modem hears) is answered by the modem. During a data transfer, the modem should be allowed to answer. Once a successful connection is established, the radiometer communications software (see section 4) attempts to transfer the most recent data. After the ASD-4 answers, but before the modem does, a caller may use a touch-tone code to transfer the call to one of the other lines. At the moment, the

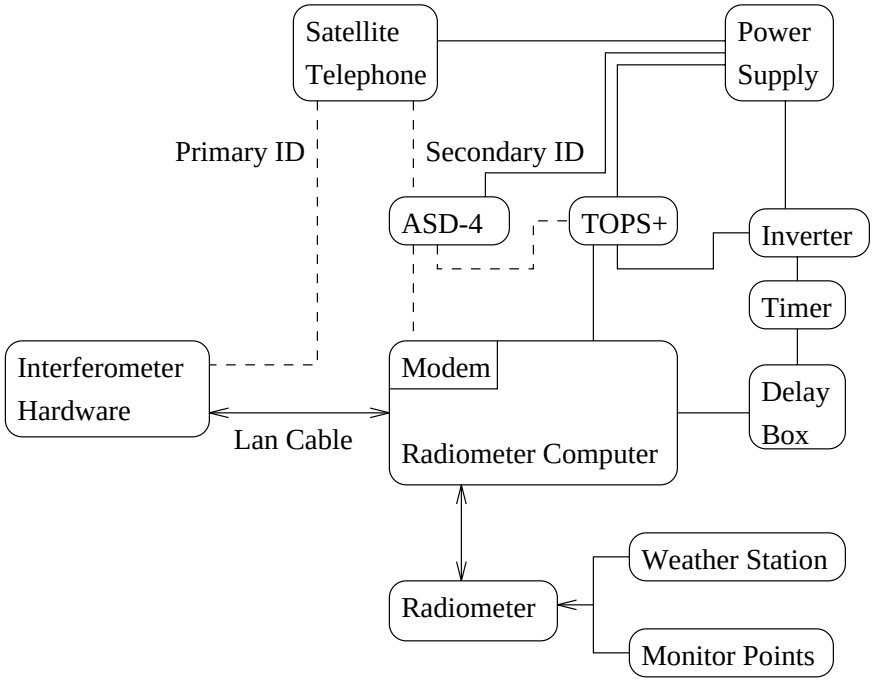

Figure 1: Block diagram of the Chajnantor, Chile, site radiometer hardware configuration. Dashed lines indication telephone lines. Solid lines without arrows indicate power lines. Solid lines with arrows indicate data lines.

only other line from the ASD-4 leads to the Black Box Corporation TOPS+ (Tone Operated Power Switch). This line is accessed by pressing “ ”. It may be necessary to press “ ” several times in rapid succession.

3.1.2 The TOPS+ (Tone Operated Power Switch)

The TOPS+ allows someone with a touch tone phone to call the site and turn power on or off for up to four separate devices. The TOPS+ takes a phone line and a power source as its inputs. The outputs are four, three pronged, electrical outlets. When a caller instructs the ASD-4 to transfer a call from the modem to the TOPS+, the TOPS+ will let the line ring three times before answering. When it answers, a voice will say, “Enter you’re access code.” The access code is a four digit touch tone code set by the four

dials on the front of the TOPS+. At the time of this writing, the access code is ( ). Once the correct access code is entered, the voice will list the current state of the four receptacles. At the time of this writing, it should say, "One: on, Two: off, Three: on, Four: on." The caller can then use single digit touch tone codes to manipulate the switch. Table 2 lists the effect of each touch tone code. After making any necessary adjustments, the caller can simply hang up, but it's much more fun to press '*'. At the time of this writing, only receptacles #1 and #2 are in use. Receptacle #1 controls the power to the radiometer computer, and can be used to cause a hard reboot. Receptacle #2 is connected to an inverter which controls the power to the radiometer itself. When power to receptacle #2 is off, the radiometer is on. When power to receptacle #2 is on, the radiometer is off. The receptacle assignments are summarized in table 3.

Code	Effect
1	Receptacle 1: Power On
2	Receptacle 1: Power Off
3	Receptacle 2: Power On
4	Receptacle 2: Power Off
5	Receptacle 3: Power On
6	Receptacle 3: Power Off
7	Receptacle 4: Power On
8	Receptacle 4: Power Off
9	Not Used
0	Not Used
#	List the state of each receptacle
*	Say, "Goodbye" and hang up

Table 2: Touch Tone Commands for the TOPS+

3.1.3 The Timer and Delay Box

Now that we have traced the path of an incoming call and described the connections between the radiometer, weather station, and monitor points, there are only a few more items to note. The radiometer computer's power flows through two more devices. First, we use a digital timer (of the type used by vacationers to turn house lights on and off) to turn the radiometer

Receptacle	Controlled Hardware	Notes
1	Radiometer Computer	See delay box discussion
2	Radiometer	See inverter discussion
3	unused	
4	unused	

Table 3: Receptacle Assignments for the TOPS+

computer off once per day (at about midnight, UT) for approximately two minutes. This forces the radiometer computer to reboot at least once per day, automatically clearing any software glitches which cause the computer to hang. Note that the same thing can be done with the TOPS+, however, there will be times when no one who knows how to use the TOPS+ is available due to weekends, vacations, or other distractions. The second device is a "delay box" on the power line. This delay box ensures that if the power goes off, the computer will remain off for approximately two minutes (this time is adjustable at the site, contact Gerry Petencin for instructions). Turning power to a computer off and then immediately back on can cause the computer to hang. The delay box prevents this situation.

3.1.4 The LAN (Local Area Network)

Finally, the LAN cable deserves a brief mention in this discussion. Both the radiometer computer and the interferometer computer run a simple LAN (Local Area Network) program called Little Big LAN. Little Big LAN will be described in more detail in the next section. The presence of the LAN has proved most useful in the past when one or the other computer has developed a fault.

3.1.5 Chajnantor Computer Hardware

Four computers are devoted to the Chajnantor site testing effort. The present radiometer computer is a Dell 486 desktop. It is working well at this time. A duplicate of this computer is kept in Tucson for use as a backup and to test software modifications. The other two computers are laptops. The Austin 486 laptop has a damaged hard drive. It will work for short times, but is unreliable for long term use. The GRID 386 works well,

but is a bit slow. It makes a good emergency backup. Both laptops have outdated radiometer software.

3.2 Mauna Kea, Hawaii, Hardware Configuration

The hardware setup on the Mauna Kea VLBA site (which was dismantled in July, 1996) was considerably simpler than the one on the Chajnantor site for two reasons. First, VLBA site technicians are available on the Mauna Kea site on a regular basis. This means that the complicated power and telephone switches are not needed. If the radiometer computer hangs, a phone call or electronic mail message is enough to get the system reset in a timely fashion. Second, the Mauna Kea VLBA site has an Internet connection. This greatly simplifies communications.

Figure 2 shows a block diagram of the Mauna Kea hardware configuration. Unlike the Chajnantor Chile setup, there are no telephone lines involved. Instead, the computer's Ethernet card allows it to be accessed from the Internet. The radiometer computer runs an ftp server which allows access to data and other files on the radiometer computer's disk. Power is supplied by the VLBA station. Unlike the Chajnantor setup, there is no weather station (we use the VLBA weather station data) or external monitor points.

3.2.1 Mauna Kea Computer Hardware

The Mauna Kea computer is a Gateway 486 desktop computer with an Ethernet card. At the time of this writing, the computer and monitor are in boxes under the table in my office.

3.3 Receiving Computer Hardware Configuration

In sections 3.1 and 3.2, we described the communications hardware at each of two sites. Our hardware discussion, therefore, can not be complete without some mention of the computer which communicates with the remote sites, referred to as the "receiving computer." Historically, the receiving computer was a PC. At the time of this writing, however, the receiving computer is a Sun workstation. The only special requirement for using a Sun workstation as the receiving computer is access to a modem. While a modem is not a standard component on most workstations, it is easy enough to add. Consult the local computer staff for more information. The modem should be set up

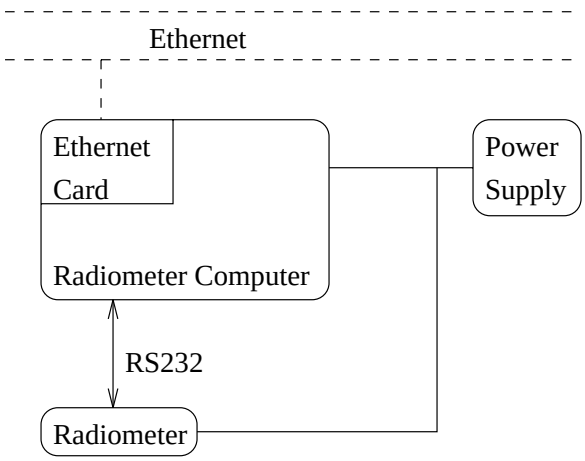

Figure 2: Block diagram of the Mauna Kea, Hawaii, site radiometer hardware configuration. Dashed lines indication Ethernet connections. Solid lines without arrows indicate power lines. Solid lines with arrows indicate data lines.

to dial out and a dedicated modem is preferable over a shared modem as it makes troubleshooting easier.

4 Radiometer Computer Software

The radiometer computer software package runs on any PC compatible computer. At the time of this writing, the software runs as a DOS application under Microsoft Windows 3.1 or 3.11. DOS is the preferred operating environment if there is no communications program running. The main component of the software package is the radiometer control, data acquisition, and processing program (CTIP) described in section 4.2. All other software is either secondary in importance or site specific. For this reason, we discuss the CTIP program in some detail in section 4.2. Next, we discuss the Chajnantor and Mauna Kea site communications software in sections 4.4 and 4.5. Between these two sites, we cover the two most likely communications software configurations. Finally, we look at the rest of the system in section 4.6.

4.1 A few notes on MS-DOS (Micro Soft Disk Operating System) File Extensions

Throughout this section we will be describing a number of MS-DOS programs. An MS-DOS file name consists of up to eight characters, followed by a ".", followed by up to three more characters. The last three characters are called the file name extension. Different file name extensions have different meanings. The ones we will encounter are summarized in table 4.

Extension	File Type
BAT	Batch
CMD	Procomm Script
COM	Compiled
DAT	Generic Data or Radiometer Data
EXE	Executable
MON	Radiometer Monitor Data
OUT	Generic Output or Allan Variances
PAS	Pascal Source Code
PIF	Program Information File
TXT	ASCII Text or Phase Stability Data
ZIP	Zip archive

Table 4: MS-DOS File Name Extensions

When we refer to a program, we will typically give the file name without the extension. The executable will have a .EXE detention and the source code will have a .PAS extension. For any other type of file, the full file name and extension will be give.

4.2 CTIP (Charlottesville TIPper)

The heart of the radiometer software package is the data acquisition software, CTIP (Charlottesville TIPper). CTIP was not written all at once. It started out in the early 1980's as a Turbo Pascal v3.0 program called ARTEST. Development continued in a rather haphazard manner through CTIP89, completed in the summer of 1989. Many programmers, some of whom preferred poor programming styles, contributed to the source code over the years. Despite this, the resulting program works. Maintenance of the program, however, is not a task for the light hearted. The use of global

variables to pass data from one procedure or function to another, the use of GOTO statements, and other bizarre programming practices mean that what appears to be a trivial modification to one part of the program may adversely affect another part of the program. Modifications should always be thoroughly tested before installing new code on a remote site.

CTIP89, with occasional, minor modifications, was in use through March of 1995, when it was updated. A bug which prevented it from running on anything faster than a 80286 was corrected. In addition, a new method for calculating the opacity was added and the data file format was changed. In May of 1996, CTIP was again modified to improve timing. All time delays in the code are now based on the computer's real time, hardware clock. There are presently two versions of CTIP in use. CTIP96 is the general radiometer control program. CTIP96C, on the other hand, contains several modifications specific to the Chajnantor MMA site testing program. Since CTIP96 is the more general program, we will describe it first.

4.2.1 CTIP96

A complete listing of CTIP96.PAS appears in appendix C.

The CTIP96 program controls the radiometer operation, collects data from the radiometer, performs the first stages of data analysis, and stores the data on disk. CTIP96 operates in one of two modes: opacity mode or phase stability mode. In opacity mode, the radiometer performs a tipping scan every ten minutes. The sky brightness is recorded at several elevations, the zenith opacity is determined by the three different methods described in appendix B, and the opacity data and monitor data are recorded in the appropriate files. In phase stability mode, CTIP96 instructs the radiometer to "stare" at the zenith for one hour. The system makes a sky brightness measurement about once every half second. CTIP96 calculates a 3.52 second average from every seven measurements and records this value on disk. At the end of the hour, CTIP96 calculates the Allan variance of the phase data and store it on disk. The radiometer can also be configured to perform a phase stability measurement while pointing straight down at a constant temperature load. This allows direct measurement of the instruments stability and is called a calibration measurement.

CTIP96 requires a number of auxiliary files. The first, and perhaps most important, is the serial port device driver, COMDRV.COM. This device driver must be installed at boot time by including the line, "DEVICE = COMDRV.COM" in the CONFIG.SYS file. The serial port driver installs

several special purpose radiometer communications functions in memory. When CTIP96 (or ARTEST) runs later, these functions are available to aid communications with the radiometer.

In addition to the serial port driver, CTIP96 reads two configuration files upon invocation. The first of these files is FUDGE.PAS, which must be located in the present working directory on the C: drive. A FUDGE.PAS file might look something like this:

```
0.955     Gain Fudge factor
19        Temperature Fudge Factor
-1.0      LOfudge
3.5       Frequency of phase stability runs in hours (0 deactivates)
5         Freq of calibration runs vs sky tests (0 deactivates)
256       Number of periods of gain averaging (default 4) (multiple of 2)
C         site id character
```

The first item on each line is a value. CTIP96 ignores anything after that first value, meaning that the rest of the line may be used for comments. The first value is the gain calibration factor. This number should be adjusted until the zenith opacity reported by CTIP96 approximately equals the tipping scan opacity on a clear day. If this number is not of order unity, there is probably a hardware problem with the radiometer.

The second and third lines should not be adjusted in most cases. The second line gives the assumed adiabatic lapse rate ($9.8°C/km$) multiplied by the assumed scale height ($1.93km$) of water vapor. It should only be adjusted if there is reason to expect that either of these assumptions is significantly in error. The third line specifies a correction to the measurements for local oscillator reflection within the radiometer enclosure. It is no longer needed and can be safely set to 0.0. The value of -1.0 presently used does not significantly affect the measurements. See appendix B for more details.

The next three lines affect the phase stability mode of the program. The first gives the frequency of phase stability measurements in hours. In the above example, the radiometer will perform one phase stability measurement every 3.5 hours. The next line specifies the frequency of calibration measurements. In this example, 1 out of every 5 phase stability measurements will actually be a calibration measurement. Note that the radiometer may ignore these values under certain circumstances (see below). The last of the phase stability measurement parameters gives the number of 3.51 second measurements used for gain averaging. In the above example, 256 periods

of 4 measurements each are used for the gain averaging. This number must be a multiple of 2 in the range of 2 to 1024. If an incorrect value is given, it will default to 4.

The last parameter in the FUDGE.PAS file is the site ID character. The data files generated by earlier versions of CTIP had the form yymmddnn.EXT, where EXT was one of the following: DAT, MON, TXT, or OUT. When the MMA project started testing multiple sites, it was decided to replace the last character of the file name extension with a character which was unique to each site. At the time of this writing, two site ID characters are in use: 'C' for the Chajnantor, Chile site and 'M' for the Mauna Kea VLBA site.

After reading in the FUDGE.PAS file, CTIP96 reads PHITIME.TXT, which, like FUDGE.PAS, must be located in the present working directory on the C: drive. The parameters in PHITIME.TXT, combined with some of the parameters from FUDGE.PAS, determine the behavior of the phase stability measurements. An example PHITIME.TXT appears below:

```
0                 0 default timing 1 file values 2 both
1                 0 will cause the converted file to be erased
960904 0845
960405 0650
```

As with the FUDGE.PAS file, CTIP96 looks for its parameters at the beginning of a line and ignores anything after them, allowing the user to leave comments. The first value determines the timing of the phase stability measurement. A 0 indicates that the phase stability run frequencies given in FUDGE.PAS should be used. A 1 indicates that CTIP96 should read the date-time pairs found on lines three through the end of the file and perform phase stability measurements only at those times. A value of 2 indicates that CTIP96 should use both methods. Any other value deactivates the phase stability mode entirely. In practice, 0 is used almost exclusively.

The second line determines if the text file containing all 1024 sky brightness measurements should be deleted once the Allan variances are calculated. This parameter was added in the days when a 20 megabyte hard disk was considered large. Today, there is no need to delete these files. In fact, since they may contain more useful information than the Allan variance file, it is probably better to keep them.

Finally, as mentioned above, the third and successive lines contain dates and times when a phase stability measurement should be done. In practice,

this feature is rarely used, so PHITIME.TXT will usually contain only two lines.

On execution, once CTIP96 figures out how to decide if it should perform a phase stability measurement, it makes the decision. If phase stability measurements are to take place at regular intervals specified in the FUDGE.PAS file, CTIP96 calculates the Julian hour, divides by the phase stability measurement frequency, and examines the remainder. If the remainder is 30 minutes or less, a phase stability measurement is started. If the phase stability measurements are to take place at specific times given in the PHITIME.TXT file, CTIP96 compares these dates and times with the date and time read from the system clock. Again, if the time is 30 minutes or less past a scheduled phase stability measurement, one is started.

Regardless of whether a phase stability measurement was performed, CTIP96 next reads the binary file STORBOOL.DAT, which must be in the present working directory on the C: drive. STORBOOL is written by CTIP96 at the end of each opacity measurement and the program WHRSTRT, which should be called once every time the computer reboots. Source code for WHRSTRT appears in appendix D. STORBOOL.DAT contains a single boolean variable which tells whether or not the system was just booted. If the system was just booted, CTIP96 will initialize its variables and determine the next data file to write. If the computer was not booted since the last measurement, CTIP96 will read some of its global variables from data files, where they were stored at the end of the previous measurement. In practice, this really isn't necessary, but with the tangled mess which is the CTIP96 source code, it is easier to leave it in than to take it out.

CTIP96 now waits until at least 10 minutes have passed since the previous opacity measurement and then performs a new opacity measurement. At this time, it also records a record in the monitor data file.

CTIP96 generates four data files. All four files have names of the form yymmddnn.EXT where yymmdd is the date in scientific format, nn is a number which starts at 00 and increments every time a new file is written (reseting to 00 at the beginning of the day) and EXT is the file name extension (see table 5).

Once CTIP96 has performed its measurement, it exits, returning control to the system or, more likely, a batch file which performs other tasks. We will describe such a batch file later in this section.

Extension	File Type
DAT	Opacity Data (binary)
MON	Monitor Data (binary)
OUT	Allan Variance Data (ASCII)
TXT	Phase Stability Data (ASCII)

Table 5: CTIP96 Data File Extensions

4.2.2 CTIP96C

CTIP96C is the version of CTIP96 which is presently running on the Chajnantor MMA site in Chile. It behaves exactly like CTIP96 with the following exceptions:

- Procedures screen and process_data were modified to reflect the addition of weather station data and other external monitor points which are read by the radiometer's data acquisition board. Sadly, a modification of the source code is the only way to add external monitor points. Fortunately, this is a simple change.
- During phase stability measurements, we also record the wind speed and direction at 3.51 second intervals. CTIP96C stores this data in three column text file with an extension of .WND. The first column gives the time. The second and third are wind speed and wind direction respectively.

4.3 CTIP.BAT and Pkzip

As indicated earlier, CTIP96 performs only one opacity measurement each time we invoke it. In fact, CTIP96 makes no more than one opacity measurement every 10 minutes. However, it only takes about 4 minutes to make the actual measurements. Phase stability measurements notwithstanding, this means that the radiometer computer spends about 60% of its time doing nothing. Historically, this time was used to run a communications program to listen for attempts to down load data. However, with the advent of multitasking operating systems for PC's, this is no longer necessary. The communications package now runs continuously. Although we no longer need to time share between CTIP96 and the communications program, we still make some use of the dead time.

CTIP96 is called from within the CTIP.BAT batch file. (A batch file is analogous to a Unix shell script.) A typical CTIP.BAT might look like:

```
@echo off
break on
WHRSTRT
:BEGINLOOP
CTIP96C
if not exist c:\procomm\lock.txt pkzip -ex -i c:\pickup\pickup.zip c:\data\*.*
GOTO BEGINLOOP
```

The first two lines tell the PC not to echo each command as it executes and to allow the user to stop the batch job by pressing Control-C. The third line executes WHRSTRT, which we described in section 4.2.1. The batch file then starts into a loop. The ":BEGINLOOP" is merely a label. It serves as a target for the "GOTO BEGINLOOP" statement at the end of the file. The first step in the loop calls CTIP96C (this is the Chile version of CTIP.BAT). Once CTIP96C completes its measurements and exits, the batch file instructs the PC to check for the existence of a lock file in the Procomm (see section 4.4) subdirectory. If the lock file exists, it probably means that Procomm is in the middle of a file transfer. If the lock file does not exists, we call the pkzip program to add the new data to an archive file in the PICKUP directory on the C: drive. For Unix users, pkzip acts like a combination of the tar utility and the compress utility and produces a single, compressed file from one or more files. The compressed file makes modem data transfers easier, faster (and therefore cheaper), and more reliable (zip archives include error checking information).

For more details on pkzip, refer to the Pkzip User's Guide.

WARNING: Because pkzip is so widely used, it is a favorite target for hackers. Virus contaminated or Trojan Horse versions of pkzip are relatively common. When upgrading, it is best to check the Pkware web page (http://www.pkware.com) for the latest version. The Usenet comp.compression newsgroup FAQ also contains information on the availability of safe versions.

4.4 Chajnantor Communications Software

The Chajnantor MMA site testing equipment uses three different communications programs. Procomm is the most important of these, as it handles regular data transfers from the site. In addition to Procomm, it is also

possible to use a "remote control" program called Close-up. Close-up is both temperamental and unreliable, yet has proved useful in some sticky situations. At the time of this writing, Close-up does not work, presumably due to a software misconfiguration of the modem on the computer in Chile. Finally, the Little Big LAN program provides a simple LAN (Local Area Network) connecting the radiometer computer with the site testing interferometer computer. Little Big LAN is mostly used to synchronize the clocks on the two PC's, but has also proven useful by allowing access to one computer through the other. This is particularly useful for diagnosing and repairing apparent communications problems.

4.4.1 Procomm

In the summer of 1989, we chose Procomm 2.4.2 to handle communications between the remote sites and the receiving computer. Procomm was chosen because its powerful script language made automation of the communications process easy. Since then, Procomm Plus has replaced Procomm in common usage. There is no reason why we could not use Procomm Plus for the radiometer project. It would, however, require modification and testing of the scripts. For detailed information on Procomm, see the Procomm manual.

On the radiometer computer, Procomm is found in the C:\PROCOMM directory. In addition to procomm's executable and configuration files, there should also be a file called REMOTE.CMD. REMOTE.CMD is the radiometer computer's Procomm script. The present version of this script appears below. Note that ';' is the comment character and the Procomm scripting language is not case sensitive.

```
;remote.cmd
SET PORT COM2
SET BAUDRATE 9600
GOSUB INIT                ;initialize the modem
LOOP:
WAITFOR ''CONNECT'' 600
IF WAITFOR
  DOS ''c:\procomm\Watchdog on''
  PAUSE 10
  TRANS ''Use Close-up (y/n)?''
  RGET S1 1 30
```

```
  FIND S1 ‘‘y’’
  IF FOUND
    DOS ‘‘c:\pro2cup.bat’’
    hangup
    GOTO END
  ENDIF
  PAUSE 1
  TRANS ‘‘Begin Transmission?’’
  RGET S1 2 30
  FIND S1 ‘‘go’’
  IF NOT FOUND
    hangup
    GOTO END
  ENDIF
  DOS ‘‘copy c:\config\lock.txt c:\procomm’’
  DOS ‘‘c:\dsz\dsz port 2 sz C:\PICKUP\PICKUP.ZIP’’
  PAUSE 1
  TRANS ‘‘Transmission Ok?’’
  RGET S1 1 30
  FIND S1 ‘‘y’’
  IF NOT FOUND
    hangup
    GOTO END
  ENDIF
  DOS ‘‘copy c:\config\done.txt c:\procomm’’
  PAUSE 1
  TRANS ‘‘Hangup’’
ENDIF
GOTO LOOP
END:
QUIT

INIT:                    ;initialize the modem
HANGUP
TRANS ‘‘ATM0!’’            ;tell modem speaker to shut up
PAUSE 3
TRANS ‘‘ATS0=1!’’          ;answer on first ring
PAUSE 3
RETURN
```

The first few lines of the script and the INIT subroutine initialize the modem and the software. The script then waits for up to 10 minutes (600 seconds) for an incoming call. If no call arrives, the script loops and starts waiting again. The reason for this loop is that procomm's WAITFOR command must be given a finite time limit.

If an incoming call does connect, the script immediately starts a program called watchdog. Watchdog monitors the serial port and reboots the radiometer computer if the DCD (Data Carrier Detect) signal is lost. This prevents the communications software from accidently hanging the computer in the event of a bad connection. For more information on watchdog, see the documentation in appendix E.

Once the script has protected itself from bad phone connections, it begins a handshaking dialog with the caller. Most of the time, the caller will be the data retrieval computer. However, the script issues enough text prompts for a human to respond as well. First, the script determines if the caller wishes to connect with close-up. If that is the case, it runs C:\PRO2CUP.BAT to change the system configuration and reboots. Pro2cup.bat will be explained in more detail later.

If the caller is not interested in using Close-up (smart caller...), the script prepares for a data transfer. First, it copies a lock file into the Procomm directory. The existence of a lock file will prevent pkzip from trying to update pickup.zip during the data transfer. Then, Procomm calls DSZ. DSZ is a program which transfers data with the z-modem file transfer protocol. At the time of this writing, Z-modem is the fastest data transfer protocol available. In its full featured version (DSZ), it is also among the most flexible and reliable. DSZ has literally hundreds of options to deal with any number of communications difficulties. For our purposes, however, we only take advantage of the capability to restart a failed file transfer, which is invoked at the receiving end. For more information on DSZ, see the DSZ documentation.

Once the data transfer is complete, the script asks the caller if it received the data file. If the caller indicates that the transfer was a success, the script copies the "done.txt" file into the Procomm directory and reboots. On reboot, the presence of the "done.txt" file in the Procomm directory signals the computer to delete the pickup.zip file and to remove the lock file. This allows pkzip to begin updating the pickup file again. The reboot also forces CTIP to start a new data file.

4.4.2 Close-up

Close-up is a remote control communications program. It allows someone calling from a remote computer to (theoretically) control the computer as if they were sitting at the console. This means that the communications software has to send graphical information as opposed to simple text. Close-up has a poorly written user interface as well as a few bugs. Close-up's data transfer protocol is also slow and unreliable, especially on noisy lines. Despite all of this, Close-up has been of use in the past.

To dial into the radiometer computer with close-up, first call with Procomm. As described above, when a call comes in to the radiometer computer, the script prompts the caller to determine if this is a data transfer, or if the caller wants close-up. If the caller selects close-up, the script modifies the system configuration files so that close-up has control of the serial port. This causes a reboot. The caller can then call back normally with close-up. See the close-up manual for more information.

WARNING: When a closeup call is completed, the caller must manually run C:\CUP2PRO.BAT to reconfigure the system for data transfers. Data transfers will not be possible until this is done.

NOTE: Strictly speaking, Close-up is not necessary. With the exception of Microsoft windows, there is nothing on the radiometer computer which requires a graphical interface. In the future, it might be worth considering the use of Procomm Plus in host mode for both data transfers and maintenance access to the radiometer computer. The Procomm Plus Host Mode script could be easily modified for this purpose and the elimination of both Procomm and Close-up would greatly simplify the radiometer computer software.

4.4.3 Little Big LAN

Little Big LAN allows two or more computers to communicate over serial or parallel connections as if they were part of a Local Area Network. The Little Big LAN manual provides unusually clear and complete documentation, so we will not describe it here. Refer to the manual for more information.

4.5 Mauna Kea Communications Software

The Mauna Kea communications set up is considerably simpler than the Chajnantor communications set up. Since the Mauna Kea VLBA site has

an Internet connection, we simply use the PC-TCP software package to set up an ftp (File Transfer Program) server.

If this communications method is chosen again in the future, contact the local systems administrator to get an Internet address for the radiometer computer and assistance on configuring the software.

4.6 The Radiometer Computer Setup

Conceptually, the radiometer software and the communications software do all of the work required to run the site testing equipment and send data back for analysis. Practically, however, one must also consider the underlying operating system. All of the software described to this point is designed to run under MS-DOS. It can also run under Microsoft Windows 3.1.

4.6.1 CONFIG.SYS

When an MS-DOS system boots, it reads two files from the root directory on the boot drive (typically C: unless there is a floppy in the A: drive). The first file, CONFIG.SYS, loads device drivers and sets some parameters for the system. The CONFIG.SYS file from the Chajnantor radiometer computer appears below. It is considerably more complicated than the corresponding file for the Mauna Kea setup because the Chajnantor setup is considerably more complicated. Note that it is often not necessary to modify the CONFIG.SYS file manually when installing new software. Most installation programs will offer to modify CONFIG.SYS for you. This is usually safe as most installation programs will first make a backup copy of CONFIG.SYS.

```
DEVICE=C:\DOS\SETVER.EXE
DEVICE=C:\HIMEM.SYS
DOS=HIGH
FILES=30
BUFFERS=40
SHELL=C:\DOS\COMMAND.COM /P /E:1024
DEVICE=C:\DOS\POWER.EXE
DEVICE=C:\IFSHLP.SYS
STACKS=9,256
DEVICE=c:\tipper\COMDRV.COM
DEVICE = C:\IOMEGA\ASPIPPA3.SYS Scan Info Quiet
DEVICE = C:\IOMEGA\SCSICFG.EXE /V
DEVICE = C:\IOMEGA\SCSIDRVR.SYS
```

```
DEVICE=C:\LBL\netunits.sys
DEVICE=C:\LBL\net00000.sys #1 ‘‘Radiometer’’
```

The first two lines are MS-DOS device drivers. SETVER.EXE merely reports the MS-DOS version number when the system boots. The second line gives MS-DOS access to the higher memory addresses in the computer. When MS-DOS was first written, many years ago, no one ever envisioned the need for more than 640 Kbytes of memory on a personal computer. In order to address more than 640 Kbytes of memory, while maintaining backward software compatibility, a special device driver (HIMEM.SYS) is required. The third line simply tells the computer to put the operating system in the high memory area made available by the HIMEM.SYS device driver.

Four of the next six lines set some of the more important MS-DOS environment variables. The FILES variable tells the operating system how many file handles should be available. In this case, no more than 30 files may be opened at once. Likewise, BUFFERS and STACKS specify the maximum number of each of these items. For STACKS, we also must specify a stack size, 256 bytes in this case. Finally, the SHELL variable tells MS-DOS where to look for its command interpreter. It should rarely be necessary to modify these variables. Installation programs will modify them for you if necessary.

The seventh and eighth lines install two more device drivers. POWER.EXE is Dell computer's power management software, and will not appear in all CONFIG.SYS files. IFSHLP.SYS is a Microsoft Windows device driver.

The remainder of the file installs device drivers for specific programs. COMDRV.COM is the serial port driver used by CTIP. This driver must be present on any computer running CTIP.

The next three drivers allow the use of the Iomega ZIP drive installed on the Chajnantor computer. Originally, the purpose of this extra drive was to provide some protection against hard disk crashes. However, such use created more problems than it solved. It could be removed if necessary.

Finally, the last two device drivers are used by Little Big LAN. See the Little Big LAN manual for more information.

4.6.2 AUTOEXEC.BAT

The second file that MS-DOS reads in the boot-up process is AUTOEXEC.BAT. The Chajnantor AUTOEXEC.BAT file appears below.

```
@ECHO OFF
```

```
rem -- THE NEXT LINE IS REQUIRED FOR DMI SUPPORT; DO NOT REMOVE OR MODIFY.
rem mi\dos\bin\sl.exe
C:\SMARTDRV.EXE /X
C:\DOS\DOSKEY /INSERT
PROMPT $P$G
PATH
C:\;C:\DOS;C:\WINDOWS;C:\MOUSE;C:\LBL;c:\tp;c:\tipper;c:\pkzip;c:\procomm;c:\dsz;c:\p
C:\UTILITY\DPMSSAVE.COM 20
SET TEMP=C:\DOS
SET COMSPEC=C:\DOS\COMMAND.COM
SET MOUSE=C:\MOUSE
@SET SCSI_DRIVER = C:\IOMEGA
@SET SCSI_UTILITY = C:\IOMEGA
C:\MOUSE\MOUSE.EXE /Q
REM [Dellmenu]
rem ELL\DELLMENU.EXE
REM [End_Dellmenu]
C:\LBL\net8
C:\LBL\par_link lpt2 int13
C:\LBL\net21
c:\lbl\utils\netclock node:2
if exist c:\procomm\lock.txt del c:\procomm\lock.txt
if exist c:\procomm\done.txt del c:\pickup\pickup.zip
if exist c:\procomm\done.txt del c:\procomm\done.txt
win

REM Reminder: Network drivers MUST be loaded before Close-Up.
```

AUTOEXEC.BAT is nothing more than a special batch file which is run when the system boots. In this example, the system uses this file to set several variables and load some more device drivers into memory. See an MS-DOS manual for details on the individual commands. See the Little Big LAN manual for details on the LBL programs.

The last four lines are of particular interest to us. First, the batch file checks to see if a lock file exists. If it does, it is removed. If the "done" file also exists, we know that a successful data transfer just took place and we can remove the pickup.zip file. Finally, the batch file starts Microsoft Windows.

4.6.3 Microsoft Windows

When Microsoft Windows starts up, it checks a special program group called the startup group. Program icons shown in the startup group window are executed as soon as Microsoft Windows starts. In our case, we have two such icons in the startup window. The first runs CTIP.BAT (see section 4.3). The second runs procomm.bat (shown below).

```
:fred
sleep
procomm /fremote
goto fred
```

Procomm.bat merely loops to ensure that if Procomm does exit unexpectedly, it gets restarted. The sleep program provides a ten second delay between the time that the batch file starts and the time that Procomm executes. Microsoft Windows will often start the programs in the startup window before checking the serial ports. It then gets confused if it finds that one of these programs has claimed the serial port. The delay allows time for Microsoft Windows to check the serial ports.

5 Setting Up a New System

Eventually, it may be necessary to set up a new site testing computer from scratch. The following procedure should work. These instructions are somewhat vague, but should cover a wide range of situations. It's probably best to copy an existing setup and use this as a checklist as you modify it.

- First, set up the computer so that it can run MS-DOS and Windows (if desired). Most new PCs come configured this way.
- Create the following subdirectories for CTIP: C:\tipper, C:\DATA.
- Install CTIP in the C:\tipper directory and edit the fudge.pas and phitime.txt files as needed. Modify the CTIP source code and recompile if any of the radiometer's external data lines are to be used.
- Choose a communications strategy (if desired). Inevitably, the communications setup is unique for every site. Try to choose communications software which will work well with the hardware available on the

site. For modem transfers, the Procomm, pkzip, and dsz combination described in section 4.4 is strongly recommended.

- Create or modify the CTIP.BAT, AUTOEXEC.BAT, and CONFIG.SYS files as appropriate for the chosen communications setup. Make sure COMDRV.COM appears in the CONFIG.SYS file.
- In Windows, use the PIF Editor in the Main Group to create icons for CTIP.BAT and the communications program (if necessary). Move these icons to the startup group.
- Add any extra software which might be useful on site. A copy of the Turbo Pascal compiler is strongly recommended. A variety of common text editors is also wise. Finally, any diagnostic software which came with the computer should be installed as well as any software required to operate any special hardware.
- Test the system thoroughly. Simulate power failures by pulling the plug at different times (during a tipping scan, during a data transfer, etc). Simulate communications failures by pulling the phone line out of the modem. Simulate failure of the radiometer by disconnecting it from the serial port. Make sure the system doesn't "hang" when any of these failures occur.
- Test the system thoroughly again.
- If the system is to be combined with other hardware (such as a LAN connection to another computer) hook up the additional hardware and test the system again.
- If the system is going to a remote site, where it will be difficult to access in the event of a problem, test the system again.
- Sit back and wait for the system to break down anyway. If you've done your job well, this won't happen very often. If you've done your job very well, a broken system will eventually fix itself.

6 Tucson Software

Up to this point, we have described the data acquisition and communications software from the radiometer computer standpoint without explaining in any

detail what becomes of this data when it leaves the radiometer computer. In this section, we will describe the software setup in use at the time of this writing.

The "receiving" site testing computer is a Sun workstation. There are a number of reasons for this. Most of the observatory's computing resources are based around workstations. The Unix operating system has a built in scheduling program, cron, which makes automation of various tasks considerably simpler. The Unix text based interface is much better suited to automation than the graphical interfaces now in favor among PC users. And, finally, Unix handles errors much more gracefully than a PC, which will usually stop what it is doing and ask the user for help.

Common data handling tasks are accomplished by C-shell scripts. Some of these scripts are called manually, but many of them are routine, and best handled by cron. Section 6.1 describes cron in general and explains a typical MMA site testing crontab. Sections 6.3, 6.4, and 6.5, describe the shell scripts which are run on a daily, weekly, and monthly basis. Section **??** describes some of the scripts which might be run manually. Finally, section 6.7 describes the shell scripts which update the site testing web pages.

6.1 Cron

The Unix cron daemon allows a user to schedule commands to execute at certain times. Tasks are scheduled by the crontab command. To submit scheduling information to cron, type "crontab filename", where "filename" is the name of a file containing the scheduling information. A typical schedule file appears below. A "#" character indicated a comment. Blank lines are ignored. Non-comment, non-blank lines contain six fields. The last field is the command to be scheduled. The first two fields give the time of day when the command should be executed. The first field is minutes. The second is hours. The third field gives the day of the month. The fourth field gives the month. The fifth field gives the day of the week (0 = Sunday). Multiple entries may be specified as a comma separated list or as a range of values separated by a dash. A star in any field indicates that all values of that field match.

```
#
# ~sfoster/tipper/events
#
# Mauna Kea Data Retrieval, Plotting, and Raw Data Archving scripts
```

```
#
#45 18 * * * /home/dietcoke/sfoster/tipper/MaunaKea/MKdata.csh
45 19 * * * /home/dietcoke/sfoster/tipper/MaunaKea/MKwdata.csh
#30 19 * * * /home/dietcoke/sfoster/tipper/MaunaKea/MKplots.csh
#00 08 * * 1 /home/dietcoke/sfoster/tipper/MaunaKea/rawdata.csh
#
# Chile Data Retrieval, Plotting, and Raw Data Archving scripts
#
30 18 * * * /home/dietcoke/sfoster/tipper/Chile/Cdata.csh
30 19 * * * /home/dietcoke/sfoster/tipper/Chile/Cplots.csh
00 08 * * 1 /home/dietcoke/sfoster/tipper/Chile/rawdata.csh
#
# Chile Meteorological Data Retrieval
#
00 2,5,8,11,14,17,20,23 * * * /home/dietcoke/sfoster/tipper/Chile/met.csh
#
# Creation of Tipper Raw Data Web Page
#
00 20 * * * /home/dietcoke/sfoster/public_html/tipper/monitor.csh
#
# Monthly cleanup of ~mma/tipper
#
00 01 1 * * /home/dietcoke/sfoster/tipper/cleanup.csh
#
# Daily Creation of localbackup.tar.Z
#
00 01 * * * /home/dietcoke/sfoster/localbackup.csh
```

In this example, we run the localbackup.csh script every night, just after midnight. The cleanup.csh script, on the other hand, runs on the first of each month. Finally, we run the rawdata.csh script every Monday morning. The function of each of these scripts will be explained in later sections.

6.2 The Tipper Directory

The shell scripts described in the coming pages must manipulate a fairly large amount of data. A sane directory structure makes this task much easier. All of the radiometer data retrieval and processing scripts reside in the tipper directory. The tipper directory contains a subdirectory for each

site. Each site directory, in turn, contains several subdirectories.

- The archive directory is further divided by month. Each month subdirectory contains all of the daily radiometer data files in an ASCII format.
- The rawarchive directory contains the same data, but in the binary format produced by the radiometer computer.
- The rawdata directory is a holding pen for any binary format opacity files and ASCII format Allan variance files which are waiting to be sent to Charlottesville.
- The database directory contains monthly and longer term databases. It also contains merged opacity, weather, and phase stability database.
- The figures directory contains postscript and gif versions of figures generated for the WWW pages and other purposes.
- In the Chile site directory, the satpics subdirectory (which is further divided by month) contains jpeg images of the GOES-8 data for South America.
- Finally, any number of other special purpose directories may be present. These directories will not be described in this document.

6.3 Daily Shell Scripts

The data retrieval and plotting shell scripts run once per day. The data retrieval scripts recover data from various sites, convert it to ASCII format (as necessary), and store it in the appropriate directories. The data plotting scripts produce the various plots used to monitor the site and equipment. Historically, the plotting scripts produced hard copy plots. The plotting scripts were kept separate from the data retrieval scripts so that the plots could print in the middle of the night. Now, the plotting scripts produce gif files and another script (monitor.csh) incorporates these plots into a web page.

Table 6 summarizes the daily shell scripts. Each script is explained in more detail below.

Script	Function
Cdata.csh	Retrieve and Store Chajnantor Data
Cplots.csh	Create Plots of Chajnantor Data
MKdata.csh *	Retrieve and Store Mauna Kea Data
MKwdata.csh	Retrieve and Store Mauna Kea Weather Data
MKplots.csh *	Create Plots of Mauna Kea Data
met.csh	Retrieve and Store South American GOES-8 Data
monitor.csh	Create Tipper Monitor Data Web Page
localbackup.csh	Create Local Backup File
MKweather.csh	Retrieve Mauna Kea Weather Data (Socorro)

Table 6: Summary of Daily Shell Scripts. Scripts marked with a '*' are not presently in use.

6.3.1 Cdata.csh

Cdata.csh retrieves data from the Chajnantor site radiometer computer. It is one of the more complicated scripts we will discuss in this section. A listing appears below.

```
#!/bin/csh

echo -n ``Chile Automatic Data Retrieval: Started ``
date

# Useful bits from the .cshrc which normally aren't executed for
# non-interactive shells.

alias ls    'ls -Fa'
set notify
umask 002
limit coredumpsize 500

setenv DISPLAY dietcoke.tuc.nrao.edu:0
setenv XFILESEARCHPATH /usr/openwin/lib/%T/%N%S

set tmp = tmp$$
```

```
cd $work/tipper/Chile

echo running > 'cdata.flg'
SoftWindows >& /dev/null
xset r on
rm cdata.flg
if (-e error.txt) then
  echo Error: DSZ reports failed file transfer.  Attempting to
continue.
  rm error.txt
endif
dos2unix session.log session.log
cat session.log >> data_retrieval.log
awk -f phonebill.awk session.log >> phonebill.log

if (-e pickup.zip) then
  ~/bin/unzip -a -L -o pickup.zip >& /dev/null
  rm pickup.zip
  rm glob*
  set phi = `ls | grep -i ``.phi'' | wc -l`
  if ( $phi != 0 ) rm *.phi
else
  echo Error: Failed to recover pickup.zip.
  goto endscript
endif

if ( -e newmon.log ) mv newmon.log $tmp.newmon
if ( -e newdat.log ) mv newdat.log $tmp.newdat
if ( -e newtxt.log ) mv newtxt.log $tmp.newtxt
if ( -e newout.log ) mv newout.log $tmp.newout
if ( -e newwnd.log ) mv newwnd.log $tmp.newwnd

set mon = `ls | grep ``.moc'' | wc -l`
if ($mon != 0) then
  foreach f (*.moc)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newmon
    if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
```

```
    mv $f rawarchive/$yymm
  end
endif
set dat = `ls | grep ``.dac'' | wc -l`
if ($dat != 0) then
  foreach f (*.dac)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    echo $date >> $tmp.newdat
    mv $f rawdata
  end
endif
set text = `ls | grep ``.txc'' | wc -l`
if ($text != 0) then
  foreach f (*.txc)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set base = `echo $f | awk '{printf(``%s\n'',substr($1,1,8))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newtxt
    if (! -e archive/$yymm) mkdir archive/$yymm
    cp $f archive/$yymm/$base.txt
    if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
    mv $f rawarchive/$yymm
  end
endif
set out = `ls | grep ``.ouc'' | wc -l`
if ($out != 0) then
  foreach f (*.ouc)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    set base = `echo $f | awk '{printf(``%s\n'',substr($1,1,8))}'`
    echo $date >> $tmp.newout
    if (! -e archive/$yymm) mkdir archive/$yymm
    cp $f archive/$yymm/$base.out
  end
  mv *.ouc rawdata
endif
set wind = `ls | grep ``.wnc'' | wc -l`
if ($wind != 0) then
  foreach f (*.wnc)
```

```
    if( -z $f ) then
      rm $f
    endif
  end
endif
set wind = `ls | grep ``.wnc'' | wc -l`
if ($wind != 0) then
  foreach f (*.wnc)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set base = `echo $f | awk '{printf(``%s\n'',substr($1,1,8))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newwnd
    if (! -e archive/$yymm) mkdir archive/$yymm
    cp $f archive/$yymm/$base.wnd
    if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
    mv $f rawarchive/$yymm
  end
endif

if(-e $tmp.newdat) awk -f plotlog.awk $tmp.newdat > newdat.log
if(-e $tmp.newmon) awk -f plotlog.awk $tmp.newmon > newmon.log
if(-e $tmp.newtxt) awk -f plotlog.awk $tmp.newtxt > newtxt.log
if(-e $tmp.newout) awk -f plotlog.awk $tmp.newout > newout.log
if(-e $tmp.newwnd) awk -f plotlog.awk $tmp.newwnd > newwnd.log

if (-e newmon.log) then
  set dates = `awk '{printf(``%s ``,$1)}' newmon.log`
  foreach d ($dates)
    set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
    set mon = `ls rawarchive/$yymm | grep $d | grep ``.moc'' | wc -l`
    if ($mon != 0) then
      foreach f (rawarchive/$yymm/$d*.moc)
        mon2text $f >> $d.mon
      end
    endif
    awk '{printf(``%7.3f  %7.3f  %7.3f  %7.3f\n'',$1,$7,$15,$16)}'
   %$d.mon>$d.wea
    if (! -e archive/$yymm) mkdir archive/$yymm
    mv $d.mon archive/$yymm
```

```
    mv $d.wea archive/$yymm
  end
endif

if (-e newdat.log) then
  set dates = `awk '{printf(``%s ``,$1)}' newdat.log`
  foreach d ($dates)
    set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
    set dat = `ls rawarchive/$yymm | grep $d | grep ``.dac'' | wc -l`
    if ($dat != 0) then
      foreach f (rawarchive/$yymm/$d*.dac)
        dat2text $f >> $d.dat
      end
    endif
    set dat = `ls rawdata | grep $d | grep ``.dac'' | wc -l`
    if ($dat != 0) then
      foreach f (rawdata/$d*.dac)
        dat2text $f >> $d.dat
      end
    endif
    if (! -e archive/$yymm) mkdir archive/$yymm
    mv $d.dat archive/$yymm
  end
endif

endscript:
rm $tmp.*

echo -n ``Chile Automatic Data Retrieval: Finished ``
date
```

The first line simply identifies the script as a C-shell script. Normally, we would call /bin/csh with the -f switch. The -f switch causes the C-shell to inherit its environment from the parent shell. However, this switch does not seem to work under Solaris and cron provides only a minimal environment. As a result, when the script executes, it will look for a .cshrc file in the user's home directory. A minimal .cshrc for this script must set the path and any other environment variables which might be needed.

The next lines let the user know that the script has started and when it started. This is not of much use when the script is run manually. However, one of cron's features is that it captures any output to STDOUT and mails it to the user. As a result, when cron runs this script, the user will receive a mail message of the form:

```
Date: Sat, 14 Sep 1996 18:34:29 -0700
From: ``S. Foster'' <sfoster@heineken.tuc.nrao.edu>
To: sfoster@heineken.tuc.nrao.edu
Subject: Output from ``cron'' command

Your ``cron'' job

/home/dietcoke/sfoster/tipper/Chile/Cdata.csh

produced the following output:

Chile Automatic Data Retrieval: Started Sat Sep 14 18:30:03 MST 1996
Chile Automatic Data Retrieval: Finished Sat Sep 14 18:34:27 MST 1996
```

Next, we set some extra environment variables, shell variables, and make sure we are in the correct directory. Note that the DISPLAY variable must be set. This is important for running SoftWindows later on.

The next block of code runs SoftWindows, a PC emulation program. SoftWindows will run procomm to handle the actual modem communications. We could do this from Unix. However, the Unix version of Z-modem contains only a minimal set of features. In addition, the Procomm script language provides more capability than any of the Unix communications programs. We will describe the SoftWindows setup in more detail below. Note that in order for SoftWindows to run, the user must be logged in and running Open Windows. It is advisable to lock the screen when not present.

Due to a bug in SoftWindows, we must call xset to turn the keyboard repeat function back on. After that, we check for the existence of an error file. The SoftWindows session creates this file if, for some reason, the data transfer failed. If detected, the script prints an error message and attempts to continue, in case a partial, but usable data file was retrieved. Cdata then converts the communications session log file from MS-DOS to Unix format and appends it to the communications log file (phonebill.log). An examination of the communications log file is the first step in troubleshooting a communications problem. An example appears below.

```
...
8-8-1996 18:30:40 - 18:32:27     1.78    0.62   Connect
8-8-1996 18:32:27 - 18:34:32     2.08    2.70
8-8-1996 18:34:32 - 18:39:46     5.23    4.17   Connect DSZ Z 150125 655 cps

8-9-1996 18:30:39 - 18:32:26     1.78    0.63   Connect
8-9-1996 18:32:26 - 18:34:30     2.07    2.70
8-9-1996 18:34:30 - 18:36:12     1.70    0.60   Connect

8-10-1996 18:30:38 - 18:35:00    4.37    3.22   Connect DSZ Z 112632 654 cps

8-11-1996 18:30:39 - 18:33:29    2.83    1.70   Connect DSZ Z 49586 616 cps

8-12-1996 18:30:37 - 18:33:37    3.00    1.82   Connect DSZ Z 54549 622 cps
...
```

Each day is separated by a blank line. The first entry gives the date. The second give the start time for the communications attempt. The third gives the end time. Note, that this time is not the total amount of satellite phone time. The next number gives the total elapsed time in minutes for the attempt. The number after that gives the elapsed time between the establishment of a successful modem to modem connection and a hangup in minutes. Note that this time will not be accurate for failed file transfers. The next several entries can be used to determine how far a file transfer attempt got before failing. If the word "Connect" appears, the two modems made a successful connection. If "DSZ" appears, it means that the Z-modem program was invoked. In other words, a file transfer was started. Everything after that is returned by Z-modem itself. A "Z" indicated a successful Z-modem transfer. An "E" indicates that an error occurred during the transfer. In either case, the number of bytes transferred are listed next, followed by the throughput in characters per second. At 9600 baud, we would expect 1200 cps. However, we never achieve this in practice. Anything over 600 cps is good throughput for the satellite phone.

More detailed information on the most recent communications attempt can be found in the "session.log" file.

```
9-12-1996 18:30:38 Try1: Dialing
9-12-1996 18:31:46 Connection Established
9-12-1996 18:31:53 Starting DSZ
```

```
9-12-1996 18:33:24 DSZ Finished
Z  52888 19200 bps  614 cps   2 errors     0 1024 pickup.zip 0
9-12-1996 18:33:26 Successful Transmission Acknowledged
9-12-1996 18:33:33 Hangup
```

The session.log file is fairly self explanatory. However, the line after "DSZ Finished" may be somewhat confusing. This line comes directly from the DSZ program and give the status code ("Z" or "E") described above, the number of bytes transferred, the computer to modem baud rate (do not confuse this with the modem to modem baud rate), the throughput, the number of errors (Note: Z-modem will resend packets which have errors), the number of flow control stoppages, the block size, the file name, and, finally, the DSZ serial number.

Next, we call unzip to extract the data files from the zip archive. See appendix G for documentation of unzip. The next section of the script moves the raw data files to either the rawdata or rawarchive directories, as apporpriate. It also creates or updates a log file for each type of data file. The log file contains all of the dates for which new data has arrived.

Finally, the scripts calls the programs dat2text and mon2text (see appendices H and I for source code and a brief description) to convert the binary data files to text format. All of the opacity data for a given day is concatenated and written to yymmdd.dat, where yymmdd is the date in scientific format in the appropriate archive directory. Likewise, all of the monitor data for a given day is concatenated and written to a file called yymmdd.mon. Weather data is then extracted from the monitor file and written to yymmdd.wea. The phase stability data (yymmddnn.txt), Allan variance data (yymmddnn.out) and high frequency wind data (yymmddnn.wnd) are simply copied to the archive directory.

As described above, we use a PC emulation program called SoftWindows to perform the actual communications tasks. SoftWindows requires some configuration. However, the configuration is a fairly straight forward, menu driven, procedure. Click on "Options" to make any necessary adjustments.

Like an actual PC, when SoftWindows starts up, it looks for a CONFIG.SYS and an AUTOEXEC.BAT file in its root directory (in this case, a binary file on the Unix file system, not a physical disk). There is nothing special in the SoftWindow's CONFIG.SYS file. The AUTOEXEC.BAT file is another matter.

```
echo off
```

```
set dszlog=dsz.log
path=c:\;c:\dos;c:\insignia;c:\tp;c:\closeup6;c:\pkzip;
set temp=c:\dos
c:\insignia\mouse.com
prompt $p$g
net use h: $HOME
net use r: /
net use t: /tmp
fsadrive e:
echo on
ver
doskey
if not exist e:\cdata.flg goto end
rem
rem Download data from Chile
rem
cd procomm
if not exist error.txt if exist pickup.zip del pickup.zip
procomm /fpickup
cd ..
if exist c:\procomm\pickup.zip copy c:\procomm\pickup.zip e:
if exist c:\procomm\error.txt copy c:\procomm\error.txt e:
if exist c:\procomm\session.log copy c:\procomm\session.log e:
if exist c:\procomm\session.log del c:\procomm\session.log
c:\insignia\exitswin
:end
```

The first thing the autoexec batch file does is set some environment variables and load a mouse driver. The next lines assign some MS-DOS logical drive names to some Unix directories. The "fsadrive" is a special assignment. The fsadrive is actually assigned in the SoftWindows configuration described above. Note that using the MS-DOS DIR command on a Unix directory may produce odd results. This is a result of the fact that MS-DOS has much more stringent file naming conventions than Unix. Any Unix filename which is too long, or otherwise improper under MS-DOS will appear as garbage. Also note that filenames are not case sensitive in MS-DOS, but are in Unix. This can also create confusion. Be very careful when copying files between SoftWindows and Unix.

Next, SoftWindows looks for the file "cdata.flg" in the tipper/Chile directory. If this file exists, SoftWindows knows that it should attempt a data transfer. If not, it skips to the end of the file and enters interactive mode. If a data transfer is to be performed, the batch file calls procomm with the pickup.cmd script (shown below). The pickup.cmd script makes three attempts to recover a pickup.zip file from the Chajnantor radiometer computer. Next, the batch file copies the pickup.zip, error.txt, and session.log files to the Unix tipper/Chile directory. As described above, pickup.zip is an archive of all of the recent radiometer data. Session.log is the log file generated by procomm. Error.txt is a flag file, if it exists, Cdata.csh knows that there was a problem with the data transfer. Finally, exitswin ends the SoftWindows session.

```
TRACE ON
TRANS ``ATS8=4!''
;TRANS ``ATF4!''
PAUSE 3
TRANS ``ATF8!''
PAUSE 3
TRANS ``ATS7=120!''
PAUSE 3
TRANS ``ATM0!''
PAUSE 3
;ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
;ASSIGN S9 ``ATDT!''
TRANS S9
DOS ``log Try1: Dialing >> session.log''
WAITFOR ``CONNECT'' 120
IF NOT WAITFOR
  GOTO TRY2
ENDIF
DOS ``log Connection Established >> session.log''
PAUSE 2
```

```
WAITFOR ‘‘Use Close-up (y/n)?’’
IF NOT WAITFOR
  GOTO TRY2
ENDIF
TRANS ‘‘n’’
MESSAGE ‘‘n’’
WAITFOR ‘‘Begin Transmission?’’
IF NOT WAITFOR
  GOTO TRY2
ENDIF
TRANS ‘‘go’’
MESSAGE ‘‘go’’
DOS ‘‘dl.bat’’
WAITFOR ‘‘Transmission Ok?’’
ISFILE ‘‘error.txt’’
IF SUCCESS
  TRANS ‘‘n’’
  MESSAGE ‘‘n’’
  GOTO TRY2
ELSE
 TRANS ‘‘y’’
  MESSAGE ‘‘y’’
  DOS ‘‘log Successful Transmission Acknowledged >> session.log’’
ENDIF
WAITFOR ‘‘Hangup’’
HANGUP
DOS ‘‘log Hangup >> session.log’’
QUIT

TRY2:
HANGUP
DOS ‘‘log Hangup >> session.log’’
TRANS S9
DOS ‘‘log Try2: Dialing >> session.log’’
WAITFOR ‘‘CONNECT’’ 120
IF NOT WAITFOR
  GOTO TRY3
ENDIF
DOS ‘‘log Connection Established >> session.log’’
```

```
WAITFOR ‘‘Use Close-up (y/n)?’’
IF NOT WAITFOR
  GOTO TRY3
ENDIF
TRANS ‘‘n’’
MESSAGE ‘‘n’’
WAITFOR ‘‘Begin Transmission?’’
IF NOT WAITFOR
  GOTO TRY3
ENDIF
TRANS ‘‘go’’
MESSAGE ‘‘go’’
DOS ‘‘dl.bat’’
WAITFOR ‘‘Transmission Ok?’’
ISFILE ‘‘error.txt’’
IF SUCCESS
  TRANS ‘‘n’’
  MESSAGE ‘‘n’’
  GOTO TRY3
ELSE
  TRANS ‘‘y’’
  MESSAGE ‘‘y’’
  DOS ‘‘log Successful Transmission Acknowledged >> session.log’’
ENDIF
WAITFOR ‘‘Hangup’’
HANGUP
DOS ‘‘log Hangup >> session.log’’
QUIT

TRY3:
HANGUP
DOS ‘‘log Hangup >> session.log’’
TRANS S9
DOS ‘‘log Try3: Dialing >> session.log’’
WAITFOR ‘‘CONNECT’’ 120
IF NOT WAITFOR
  GOTO BAD
ENDIF
DOS ‘‘log Connection Established >> session.log’’
```

```
WAITFOR ‘‘Use Close-up (y/n)?’’
IF NOT WAITFOR
  GOTO BAD
ENDIF
TRANS ‘‘n’’
MESSAGE ‘‘n’’
WAITFOR ‘‘Begin Transmission?’’
IF NOT WAITFOR
  GOTO BAD
ENDIF
TRANS ‘‘go’’
MESSAGE ‘‘go’’
DOS ‘‘dl.bat’’
WAITFOR ‘‘Transmission Ok?’’
ISFILE ‘‘error.txt’’
IF SUCCESS
  TRANS ‘‘n’’
  MESSAGE ‘‘n’’
  GOTO BAD
ELSE
  TRANS ‘‘y’’
  MESSAGE ‘‘y’’
  DOS ‘‘log Successful Transmission Acknowledged >> session.log’’
ENDIF
WAITFOR ‘‘Hangup’’
HANGUP
DOS ‘‘log Hangup >> session.log’’
QUIT

BAD:
HANGUP
DOS ‘‘log Hangup >> session.log’’
ISFILE ‘‘pickup.zip’’
IF SUCCESS
  DOS ‘‘log Transfer Failed: Some data recovered’’
ELSE
  DOS ‘‘log Transfer Failed: No data recovered’’
ENDIF
QUIT
```

6.3.2 Cplots.csh

Cplots.csh reads the newdat.log, newmon.log, newout.log, newtxt.log, and newwnd.log files created by Cdata.csh and creates a plot of the data for each date listed in each log file. Each plot is created by a call to one of the following scripts: tauplot.csh for opacity data, monplot.csh for monitor data, txtplot.csh for phase stability data, wndplot.csh for wind speed data, and wdrplot.csh for wind direction data. Each of these scripts are described in section 6.6. The giftrans program is used to make the background of each plot transparent and then each plot is copied to public_html/tipper, where monitor.csh will incorporate it into the Tipper Monitor Data Web Page.

A complete listing of Cplots.csh is given in appendix J.

6.3.3 MKdata.csh

MKdata.csh is no longer in use, but it is worth mentioning as an example. Mkdata.csh used to perform the same tasks as Cdata.csh except that it retrieved data from the Mauna Kea MMA site. There are two main differences between the two scripts. First, MKdata.csh uses ftp to retrieve its data. When ftp is given an address, it first checks for a .netrc file in the user's home directory. If the .netrc file exists, ftp will look for an entry for the given address. If such an entry exists, it will use the listed login name and password to log onto the system. Furthermore, if a macro called "init" is defined for that host, ftp will execute that macro. This allows ftp sessions to be completely automated. An example of a .netrc file appears below:

```
machine tipper.vlba.nrao.edu login mma password iswtoantsfo
macdef init
  type binary
  cd pickup
  get pickup.zip
  del pickup.zip
  bye

machine ftp.aoc.nrao.edu login anonymous password sfoster@heineken.tuc.nrao.edu
macdef init
  cd pub/sfoster
  mget *.wea
  bye
```

Second, MKdata.csh retrieves Mauna Kea weather data from the AOC in Socorro. Unlike Chajnantor, the MMA does not have its own weather station on the Mauna Kea site. Instead, we use the data available from the VLBA weather station. The MKweather.csh shell script running in Socorro reads the Mauna Kea weather data from the VLBA monitor data and stores it in an ftp directory for pickup. This extra data retrieval step accounts for the second .netrc entry in the above example.

A complete listing of MKdata.csh is available in appendix K.

6.3.4 MKwdata.csh

When the Mauna Kea site testing project was discontinued, we, nonetheless, continued to record weather data for the site. The MKwdata.csh script is a stripped down version of MKdata.csh which only retrieves the weather data from the AOC in Socorro. A complete listing is available in appendix L.

6.3.5 MKplots.csh

MKplots.csh performs the same function as Cplots.csh. This script is no longer in use. Although we still record weather station data for Mauna Kea, no plots are generated. A full listing is available in appendix M

6.3.6 met.csh

The met.csh script was originally written by Tony Beasley and later modified by Scott Foster. It uses a text-only world wide web browser called lynx to retrieve GOES-8 images of South America from the ESO web pages. The data is written to the tipper/Chile/satpics/yymm directory where "yymm" is the year and month. This script is run every three hours.

For the past several months, the ESO web pages have been very unreliable. As a result, the script often fails to recover data. Unfortunately, the script does not handle this error condition. The only way to find out if a data retrieval failed is to check the satpics directories for short ($< 200 byte$) image files. However, the script still recovers enough to be of use. A full listing appears below.

```
#!/bin/csh
# Script to automatically pull ESO Chilean meteorology data
# to specified directory, naming files using UT access times.
# Lynx exists on Sun systems at AOC.
```

```
#
# V1.0 BEASLEY 950502

# Archive location; modify as appropriate
set ARCHIVE=$work/tipper/Chile/satpics

# Desired images
set
URL1=''http://http.hq.eso.org/garching-info/computing/weatherdir/chile_meteo.
jpeg''
set
URL2=''http://http.hq.eso.org/garching-info/computing/weatherdir/chile_over.j
peg''

# File names
set datadir = `date -u +%y%m`
if (! -e $ARCHIVE/$datadir) mkdir $ARCHIVE/$datadir
set FILE1=''$ARCHIVE/$datadir/`date -u +%y%m%d_%H%M`.jpeg''
set FILE2=''$ARCHIVE/$datadir/`date -u +%y%m%d_%H%M`_mosaic.jpeg''

# Get the images
lynx -source -force_html $URL1 > $FILE1
lynx -source -force_html $URL2 > $FILE2
```

6.3.7 monitor.csh

The monitor.csh (see appendix N for a complete listing) resides in the public_html/tipper directory. This script takes the data plots produced by Cplots.csh and MKplots.csh, deletes all but the seven most recent days worth of data, and incorporates the rest into an HTML document accessible by the World Wide Web. This web page can then be accessed by anyone who needs to monitor the raw data, the radiometer performance, or the power system on the Chajnantor site.

6.3.8 localbackup.csh

A few months ago, NRAO-Tucson set aside roughly 50MB of disk space per user for the purpose of backing up high priority data. Every night, a shell script reads the .localbackup file in each user's home directory and

copies the listed files to the backup disk. We use this feature for any MMA site testing data which has yet to be backed up on tape. Unfortunately, the MMA site testing program is capable of producing a large number of small files, scattered among a number of different directories. This can cause problems for the localbackup program. As a result, the MMA site testing .localbackup file (presently ~ sfoster/.localbackup) contains only one file name: ~ sfoster/localbackup.tar.Z. The localbackup.csh script produces this data file once per day, a few hours before the system localbackup program executes.

```
#!/bin/csh -f

set tmp = tmp$$

if (! -e .timestamp) then
  echo Error: Unable to find timestamp file
  exit(1)
endif
if (-e localbackup.tar.Z) rm localbackup.tar.Z

find ~sfoster/tipper/Chile/archive -newer .timestamp -type f -print >> $tmp.1
find ~sfoster/tipper/Chile/rawdata -newer .timestamp -type f -print >> $tmp.1
find ~sfoster/tipper/Chile/rawarchive -newer .timestamp -type f -print >> $tmp.1
find ~sfoster/tipper/MaunaKea/archive -newer .timestamp -type f -print >> $tmp.1
find ~sfoster/tipper/MaunaKea/rawdata -newer .timestamp -type f -print >> $tmp.1
find ~sfoster/tipper/MaunaKea/rawarchive -newer .timestamp -type f -print >> $tmp.1

tar -cf localbackup.tar -I $tmp.1
compress localbackup.tar

rm $tmp.*
```

The localbackup.csh script uses the find command to search the MMA site testing data directories for files which are newer than the .timestamp file contained in the home directory. A temporary file holds the list of new data files. The tar command then archives all of these data files into one tar file, which is compressed to save disk space.

Note: When the disk containing the MMA site testing data is backed up, the .timestamp file should be touched. If this isn't done, the localbackup.tar.Z file will eventually exceed the 50MB size limit.

6.3.9 MKweather.csh

Unlike the other scripts described in this section, MKweather.csh runs in Socorro, where it has access to the VLBA monitor data. The MKweather.csh script calls sara, a VLBA program which accesses the monitor data. The Mauna Kea weather data for the current month and previous month is extracted and stored in a data file for each day's worth of data. These data files are then made available by anonymous ftp for MKdata.csh or MKwdata.csh in the pub/sfoster directory.

See appendix O for a complete listing.

6.4 Weekly Shell Scripts

Two scripts run on a weekly basis. They are the rawdata.csh scripts in the Chile directory and the MaunaKea directory. Both scripts are quite similar, and the Mauna Kea script is no longer in use, so we will restirct our discussion to the Chile rawdata.csh script.

```
#!/bin/csh

echo -n ``Chile Automatic Data Archiving: Started ``
date

set tmp = tmp$$

cd $work/tipper/Chile
set f = `date +%y%m%d`C
zip -D $f.zip rawdata/* >& /dev/null
foreach rf (rawdata/*)
  set yymm = `echo $rf:t | awk '{printf(``%s'',substr($1,1,4))}'`
  if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
  mv $rf rawarchive/$yymm
end
mv $f.zip ~mma/tipper
if (! -e ~mma/tipper/$f.zip) then
  echo Error: Archive file not created.
  exit 1;
endif
echo Hello Joanne, > $tmp.1
echo ``'' >> $tmp.1
```

```
echo The file, $f.zip, has been placed in the tipper directory on >>
$tmp.1
echo heineken and is ready to be picked up. >> $tmp.1
echo ‘‘’’ >> $tmp.1
echo Scott >> $tmp.1
/usr/ucb/mail -s ‘‘New Tipper Data Avilable’’ jnance < $tmp.1
rm $tmp.*

echo -n ‘‘Chile Automatic Data Archiving: Finished ‘‘
date
```

Every Monday monring, the rawdata.csh script takes any .dat and .out files stored in the rawdata directory by Cdata.csh and calls zip (see appendix F for documentation) to bundle these files into a single .zip file. This .zip file is copied to the tipper subdirectory of the mma home directory. An e-mail message is then sent to Joanne Nance in Charlottesville, who will retrieve the data, do her own processing, and archive it.

6.5 Monthly Shell Scripts

There is only one monthly shell script, cleanup.csh. A listing appears below.

```
#!/bin/csh

cd /home/dietcoke/sfoster/tipper

echo -n ‘‘Monthly zip archive directory cleanup: Started ‘‘
date

set zippath = ~mma/tipper

set yy = ‘date +%y‘
set mm = ‘date +%m‘

set old = ‘echo $yy $mm | awk -f cleanup.awk‘
rm $zippath/$old*.zip
echo ‘‘Monthly zip archive directory cleanup: Removing all zip files from $old’’
echo -n ‘‘Monthly zip archive directory cleanup: Finished ‘‘
date
```

On the first of every month, the cleanup.csh script removes all .zip files from "the month before last." This is done solely to conserve disk space.

6.6 Incidental Shell Scripts

There are a number of shell scripts which are run on an "as needed" basis. They are summarized in table 7.

Script	Fucntion
tauplot.csh	Creates Opacity Plots
monplot.csh	Creates Monitor Plots
txtplot.csh	Creates Phase Stability Plots
wndplot.csh	Created Wind Speed Plots
wdrplot.csh	Creates Wind Direction Plots
hist.csh	.ps Histogram Creation
time.csh	Creates Time Series Plots
Cdb.csh	Database update

Table 7: Summary of Incidental Shell Scripts.

A listing of the tipper/Chile directory will reveal many other shell scripts. Some of these are special purpose scripts, used once, then forgotten. Others are "subroutine" scripts. They are called by other shell scripts. Use caution when deleting a shell script that you think is unused.

6.6.1 tauplot.csh, monplot.csh, txtplot.csh, wndplot.csh, and wdrplot.csh

The tauplot.csh, monplot.csh, txtplot.csh, wndplot.csh, and wdrplot.csh scripts produce plots of opacity and weather vs. time, radiometer monitor data vs time, phase stability data vs. time, wind speed data vs. time, and wind direction data vs. time respectively for a given date. Each script must be run from either tipper/Chile or tipper/MaunaKea as appropriate. Each script takes a date (in the yymmdd format) as an argument as well as an optional pgplot device name. If no device name is specified, /ps (or /vps for monplot.csh) is assumed and the output is directed to the printer. These scripts are also called by Cplots.csh and MKplots.csh with either /gif or /vgif

as device names). Listings of these scripts are available in appendicies P - T.

6.6.2 hist.csh

The hist.csh script generates a postscript plot with two panels. The top panel contains a histogram distribution of a data set. The optional lower panel contains a cumlative distribution function with quartiles. The script has a number of options, which are displayed if the user types the name of the script.

```
Usage: hist.csh dbase col xmin xmax hbins cdfflag upflag xlabel title outfile
```

The "dbase" parameter is the path and file name of the database file which supplies the data. The "col" parameter is the column number. The "xmin" and "xmax" parameters specify the limits of the x-axis. The number of bins used in the histogram are given by "hbins". The two flags, "cdfflag" and "upflag" determine whether to include a cumulative distribution function and the percentage of time that the instrument was running. The next two parameters, "xlabel" and "title" provide title strings for the histogram x-axis and the plot itself. Remember to quote these strings if they contain more than one word. Finally, "outfile" is the name of the output file.

Both hist.csh and gifhist.csh (which produces a gif version of the same plot) are called by summary.csh, which is called in turn by Cdb.csh, however, they are often useful on their own. A complete listing of hist.csh is given in appendix U. A listing of gifhist.csh is not included because it is quite similar to hist.csh.

6.6.3 time.csh

The time.csh script is conceptually similar to hist.csh. However, this script produces a plot of a given quantity versus time. Most of the parameters are the same. However, it is now possible to specify the minimum and maximum values for the y-axis. If these values are not present, the plot will be automatically scaled to show all of the data.

```
Usage: time.csh dbase xcol ycol [ymin ymax] xlabel ylabel title outfile
```

There is no giftime.csh, as the time series data is not published on the world wide web. The time.csh script, like hist.csh, is called from summary.csh, but is sometimes useful by itself. A complete listing is given in appendix V.

6.6.4 Cdb.csh

The Cdb.csh shell script updates the Chajnantor database files. Cdb.csh first creates a Makefile, then runs the make utility. For each month of data, make checks to see if any of the files in the archive directory have been modified. When make finds a month with new data, it calls two scripts, month.csh and summary.csh. The month.csh script creates opacity, weather, and a merged databases for a given month and writes them to the tipper/Chile/database directory. The summary.csh script creates the various data summary plots and writes them to the tipper/Chile/figures directory. Once the databases for each month are updated, make calls the dbcat.csh script to concatonate the data from each month to create a full database.

A similar script called MKdb.csh, no longer in use, was used for the Mauna Kea data.

A complete listing of Cdb.csh is available in appendix W.

6.7 World Wide Web Shell Scripts

The last two scripts we will describe in this section actually belong to the mma account. Both of these scripts run out of the /home/heineken/ftp/observerinfo/mma/sites directory and are used to update the MMA Site Testing world wide web page.

The first of these scripts, tipper.csh, simply copies all of the gif files in the tipper/Chile/figures and tipper/MaunaKea/figures directories to /home/heineken/ftp/observerinfo/mm and all of the postscript files in the figures directories to the MMA Site Testing ftp directory, /home/heineken/ftp/mma/sites. However, it does not copy any figures from the current month, on the assumption that these figures will be incomplete. A listing of this script is available in appendix X.

The second script, mksites.csh, creates the actual HTML files which comprise the MMA site testing world wide web pages. The mksites.csh script taylors the HTML files to the plots present in the sites directory, so it is important that no stray files be present. A complete listing of the mksites.csh script may be found in appendix Y.

The user should "su mma" in order to run these scripts.

6.8 Procedures

Despite the apparent complexity of the scripts presented in the previous sections, monitoring and maintaining the radiometer data is quite simple if

these procedures are followed.

On a daily (weekday) basis:

- Read any e-mail reports from shell scripts. Track down errors if neccessary. If a communications attempt fails, try to determine what went wrong, but don't panic unless it fails for three or more consecutive days. Any other shell script error should be investigated. Note: if a data transfer fails, the subsequent plotting script will also fail.
- Use a World Wide Web browser to examine the radiometer monitor data. Watch for unusual values in each of the readbacks (this will require some experience). Watch for flat lines on all readbacks (bad A/D card or other problem). Watch the opacity data for large discrepencies between the tipping scan opacity and the zenith opacity (an indication of a calibration problem).

At the end of each month:

- Manually examine all of the opacity data for obvious errors. Large fit errors, unusual gains, sudden jumps in opacity readings, and negative opacities are all examples of bad data. Delete any bad data found.
- Use the check program to make sure the time entries in the dat and mon files are ok. Fix if neccessary. The two most common problems are with the clock rolling over after midnight (fix: set to 24.000) and repeated data (fix: delete redundant data points).
- Run Cdb.csh to update the databases and figures.
- Log in as mma. Run tipper.csh followed by mksites.csh.
- Use a World Wide Web browser to check the new figures for accuracy. Sometimes you will miss bad data points.
- Back up the hard disk.
- Touch the .timestamp file.

A Satellite Telephone Dialing Instructions

B Opacity Measurement

The following section is an excerpt from the Radiometer Error Analysis memo. In this section, we review the radiometer operation and measurements. In the next section, we review the algorithms used to derive the opacity from these measurements. Much of the information in these two sections was originally presented bye McKinnon (1987) and Liu (1987).

B.1 Measurements

The sky signal is reflected off of an external parabolic mirror through a small window in the radiometer enclosure. A chopper wheel positioned between the primary and secondary mirrors alternately selects signals from the primary mirror, a reference load (often referred to as the "cold load") ($45°C$), and a hot load ($65°C$). The selected signal then passes through a lens into the mixer feed horn. The signal is mixed with a local oscillator signal generated by a frequency tripled 75 GHz Gunn oscillator. The 1.5 GHz IF is then passed through two amplifier stages and a bandpass filter before arriving at the square law detector.

The radiometer measurement of the sky brightness temperature (T_s) is given by

$$T_s = T_r + (1-\epsilon)T_1 + \epsilon T_a(1-e^{-A\tau}) + T_{3K}\epsilon e^{-A\tau} \qquad (1)$$

where

- T_r = Receiver Temperature
- ϵ = coupling efficiency
- T_1 = temperature at which losses are terminated
- T_a = mean temperature of the atmosphere

- τ = Zenith Opacity
- A = airmass
- T_{3K} = temperature of the microwave background

Likewise, the measured temperatures of the reference (cold) load (T_c) and the hot load (T_h) are given by

$$T_c = T_r + (1 - \epsilon)T_1 + \epsilon T_c \tag{2}$$

$$T_h = T_r + (1 - \epsilon)T_1 + \epsilon T_h \tag{3}$$

The square law detector and its associated electronics returns a voltage proportional to the difference between the reference temperature and the sky temperature, where the gain constant, g, is the constant of proportionality. Multiplying equations 1 and 2 by g and subtracting equation 1 from equation 2 gives:

$$V_{cs} = G(T_c - T_a) + G(T_a - T_{3K})e^{-A\tau} + V_{do,cs} \tag{4}$$

where $G = \epsilon g$ is the system gain and $V_{do,cs}$ is the detector offset voltage. The radiometer is designed such that $V_{do,cs} = 0$. However, in practice, there is a small detector offset of the order of 5 - 20 millivolts.

The square law detector also returns the difference between the brightness temperatures of the reference load and the hot load. Multiplying equations 2 and 3 by g and subtracting gives:

$$V_{ch} = G(T_c - T_h) + V_{do,ch} \tag{5}$$

where $V_{do,ch}$ is another detector voltage offset, which is independent of $V_{do,cs}$.

B.2 Solving for the Opacity

B.2.1 Gain Calibration

The radiometer returns the two voltages given in equations 4 and 5 to the radiometer control computer. Under the assumption that $V_{do,ch} = 0$, and using temperature measurements from sensors on the hot and cold loads, equation 5 is solved for the radiometer system gain (G). However, this calculated gain is not quite correct. The hot load and the cold load temperature sensors are embedded in their respective loads. The radiometer, however,

sees the temperature at the surface of each load. A temperature gradient in either load will cause an error in the gain measurement. To compensate for this error, we apply a gain correction factor (colorfully referred to as the "gain fudge factor"). This gain correction factor is typically less than a 5% and is set in such a way that the zenith opacity measurement and the tipping scan opacity measurement agree.

B.2.2 Some Initial Approximations

Once the gain is calculated, we can turn to the solution of equation 4 for the opacity. We currently use three techniques to solve for the opacity (τ). All three of these techniques require a number of approximations. First, we assume that the physical temperature of the atmosphere follows an adiabatic lapse rate weighted by the exponential distribution of water vapor (McKinnon, 1987). For a lapse rate of $9.8°K/km$ and a scale height of $1.8km$, we find that $T_a \approx T_{amb} - 17°K$, where T_{amb} is the ambient temperature. Second, we assume that the $3°K$ background, T_{3K}, is negligably small compared to T_a. Finally, we assume that $V_{do,cs}$ is also negligably small. Application of these approximations results in the following, simplified version of equation 4.

$$V_{cs} = G(T_c - T_a) + GT_a e^{-A\tau} \tag{6}$$

We will examine the validity of these assumptions in the next section.

B.2.3 Direct Zenith Opacity Measurement

The first technique for calculating the opacity is the most straightforward. It requires only a single measurement of the sky at the zenith (although in practice, we average ten measurements to reduce random errors) and a direct inversion of equation 6.

$$\tau = -ln(\frac{V_{cs}}{GT_a} - \frac{(T_c - T_a)}{T_a}) \tag{7}$$

(Note, this equation is a correction of an error in MMA memo #40). As we shall see in the next section, this technique, while simple, is the least robust of the three. However, the other two techniques depend on some additional assumptions. In conditions where these additional assumptions break down (specifically, at high opacities), this technique is still accurate enough to give us a good indication of the opacity.

B.2.4 Tipping Scan Opacity Measurement

If we rewrite equation 6, taking the natural logarithm, we find that

$$ln(V_{cs} - G(T_c - T_a)) = ln(GT_a) - A\tau \tag{8}$$

If we take airmass as the independent variable, equation 8 is the equation of a straight line with a slope of $-\tau$.

The second technique for calculating the opacity takes advantage of equation 8 by having the radiometer perform a tipping scan. During a tipping scan, the primary mirror of the radiometer tips in order to make measurements at more than one airmass. In practice, we average ten measurements (again, to reduce random errors) at each of eleven air masses.

If we assume that the opacity does not vary significantly with time during a tipping scan and that it varies with elevation only in proportion to the airmass, we can perform a least squares fit to $ln(V_{cs} - G(T_c - T_a))$. We could simply take the slope returned by the least squares fit and stop. However, the least squares fit returns the y-intercept as well as the slope. We use the value of the y-intercept to improve our estimate of T_a. We then re-fit the line, and take the second calculation of slope to determine the opacity.

It should be noted that the assumption that the opacity not vary significantly temporally or spatially tends to break down under high opacity conditions ($\tau > 0.5$).

B.2.5 Gain Corrected Tipping Scan Opacity Measurement

The third technique was developed to compensate for problems with the gain calibration. If the gain is not correctly measured, but is still stable over the tipping scan, it will result in an incorrect opacity such that equation 6 is still satisfied. Therefore, we can derive a gain correction factor as follows:

$$G(T_c - T_a) + GT_a e^{-A\tau} = G^*(T_c - T_a) + G^* T_a e^{-A\tau^*} \tag{9}$$

where

- G is the correct gain
- τ is the correct opacity
- G^* is the inaccurate gain
- τ^* is the inaccurate opacity

Solving for $\frac{G}{G^*}$ yields

$$\frac{G}{G^*} = \frac{(T_c - T_a) + T_a e^{-A\tau^*}}{(T_c - T_a) + T_a e^{-A\tau}} \tag{10}$$

By taking the zenith opacity to be the incorrect opacity, and the tipping scan opacity to be a first guess at the correct opacity, we can then apply the gain correction to calculate a new gain. We use the corrected gain with the second technique to obtain a new opacity. Since the tipping scan opacity was only a first guess, we can now repeat the process, using the tipping scan opacity as the incorrect value and the newly calculated opacity as the correct value. If we iterate, the gain correction factor quickly approaches 1. The 20 iterations used in the radiometer control code are more than adequate.

B.2.6 The Local Oscillator Fudge Factor

In the original radiometer design, the material used in the window between the primary mirror and the chopper wheel was perpendicular to the direction of propagation of the radiation. The radiometer is designed such that half of the local oscillator power is radiated out of the mixed cavity, reflected off of the secondary mirror, and out of the radiometer enclosure. However, some of the local oscillator signal would reflect back off of the window and into the radiometer. This caused the sky to appear to have a higher brightness temperature than it really did. To compensate, a constant, known as the local oscillator fudge factor, was subtracted from the sky signal. This adjustment was not implemented correctly in the direct zenith opacity measurement.

The radiometer window has since been redesigned. The local oscillator fudge factor is probably no longer needed, The software configuration file still carries a $1^\circ K$ correction, however. Simulations (explained in the next section) show that such a small correction does not have a significant impact on the opacity measurements.

C CTIP96.PAS

```
{10Oct89              C T I P 8 8 . P A S
         A hybrid program derived from ARTEST1 ,BIGFILE, and PHITEST
         which automates the tipper built in Charlottesville. }

PROGRAM CTIP88(tau_file, mon_file, header_file, phase_file);
```

```
{$N+}  {Turns on the 8087 math coprocessor}

{$M 65520,0,655360}  (* Memory allocation for the stack and the heap *)

uses Dos, Crt ;

LABEL  alpha, beta, gamma, delta, psi;

CONST

{constants for program operation}

    prog        : STRING[14] = 'CTIP88a';
    version     : STRING[10] = '4.2';       {converted to Turbo 4.0 and
                                              uses phitest as a procedure}

  {added REGULAR & default}
   {version 3.1 includes communications revisions made by J. Holliman (7/87)}
   {version 4.1 is Turbo 4.0 compatible and does a phase stability scan}
   {revised by J. Ogle (2/89)}
   {version 4.2 cleaned up logic problems with the phitest procedures and
   added calibration runs to the phitests.  revised by S. Foster (6/89)}
    revdate     : STRING[10] = '890627';
    hardversion : STRING[80] = ' ';
    softversion : STRING[80] = ' ';
    fil         : string[15] = 'C:fudge.pas';
    phicount    : string[15] = 'c:phicount.txt';

   { Gfudge      : SINGLE = 0.975;
    Tfudge      : SINGLE = 21;}
    phitime     : string[15] = 'C:phitime.txt' ;   {file of times for phitest}
    phi_num_reads : INTEGER = 7; {No. points averaged in PROCEDURE PHITEST}
    num_reads   : INTEGER = 10; {No. points averaged at each airmass}
    num_airs    : INTEGER = 11;  {No. airmasses observed per scan}
    tauint      : INTEGER = 10; {Minutes between start of successive scans}
    monint      : INTEGER = 10;{Minutes between monitor records}
    default     : SINGLE = 0.01;
```

```
    check       : BOOLEAN = false;

    {When check is true, CTIP88 generates its own opacity and monitor data.}
    {The opacity data gives a perfectly straight line with a slope of -1,}
    {i.e. tau = 1,fit = -1, and rms = 0.  check is false in normal operations}

{constants for I/O, etc.}


    prefix          : BYTE = $16;
        odd = $0b;    even = $1b;
    expected_length : BYTE = 5;
    byte_count      : INTEGER = 2;
    ioctl_input     : INTEGER = 2;
    file_handle     : INTEGER = 3;           {standard aux device}
    buf_count       : INTEGER = 0;
    ret_flag        : INTEGER = 1;
    set_ascii       : INTEGER = 2;
    set_binary      : INTEGER = 3;
    flush           : INTEGER = 4;
    com1            : INTEGER = 0;
    com2            : INTEGER = 1;
    block_length    : BYTE = 255;
    state_adr       : BYTE = $80;
    gozen           : BYTE = $01;
    cw              : BYTE = $04;
    step            : BYTE = $02;
    gozero          : BYTE = $FE;
    cwzero          : BYTE = $FB;
    stzero          : BYTE = $FD;



TYPE

  WHERETOSTART = BOOLEAN;

  string16  = STRING[16];
  string9   = STRING[9];
```

```
  string20  = STRING[20];

  phi_array = ARRAY [1..1024] OF single;
  araw_data = ARRAY [1..16] of single;
  aproc_data = ARRAY [1..16] of single;
  taproc_data = ARRAY [1..16] of single;

  PHIREC = RECORD
             W2,X2,Y2,T2 : PHI_ARRAY ;
             END ;

  PHIDATA   = FILE OF PHIREC ;
  tau_array = ARRAY [0..10] OF SINGLE;
  mon_array = ARRAY [1..16] OF SINGLE;

  rec = RECORD
             time1        : STRING[4];   {UT in HHMM format}
             tau1         : SINGLE;      {tipping scan opacity}
             sigmatau1    : SINGLE;      {tipping scan rms}
             tauz1        : SINGLE;      {Zenith Opacity}
             Vz1          : SINGLE;      {Zenith mean voltage}
             GainZ1       : SINGLE;      {Zenith gain correction}
             sigmaVz1     : SINGLE;      {Zenith rms voltage measurement}
             gain1        : SINGLE;      {assumed gain}
             tauI1        : SINGLE;      {iterated opacity}
             sigtauI1     : SINGLE;      {iterated opacity rms}
             gainI1       : SINGLE;      {iterated gain}
             Tamb1        : SINGLE;      {Ambient Temperature}
{Double check calculation of tamb!!!  Does weird gain calc alter this?}
             Tc           : SINGLE;      {Cold load temperature}
             Th           : SINGLE;      {Hot load temperature}
             x1,y1,z1,G1  : tau_array; {airmass, ln(vsd), rms(vsd)}

           END;

  datafile = FILE OF rec;

  rec2 = RECORD
    time1   : STRING[4];     {UT in HHMM format}
```

```
    m1      : mon_array;     {Analog monitor data}
    dmon2   : INTEGER;       {Digital monitor word}
  END;

  monfile = FILE OF rec2;
  PIECE = STRING[20] ;


VAR

{variables for program operation}
    ISPHITEST             : BOOLEAN ; {is phitest on}
    AUX,data1file         : TEXT ; {treat AUX like a textfile}
    data2file             : TEXT ;
    STORBOOL              : FILE OF WHERETOSTART;    {IS CVTIPPER BEING EXECUTED
                                                     AFTER COMING FROM SCOM?}
    WHATISIT              : WHERETOSTART;
    NEWI                  : INTEGER;
    GLOB1                 : FILE OF INTEGER;
    GLOB2                 : FILE OF SINGLE;
    GLOB3                 : FILE OF BOOLEAN;
    GLOB4                 : FILE OF BYTE;
    GLOB5                 : FILE OF PIECE;
    fudge_file            : text;
    phitest_file          : text;
    TIPPING               : BOOLEAN;

    PHASE_FILE              : PHIDATA ;
    PHASEFILE               : piece;
    fil1,fil2               : piece;
    sig1,hot1,ref1,cold1    : PHI_ARRAY ;
    interval              : INTEGER ;
    space,num_avg,TD1,TD2 : INTEGER ;
    num_points,run    : INTEGER ;
    time1,time2,time3 : INTEGER ;
    negtime,checktime,number  : INTEGER ;
    lapse             : array[1..7] of integer ;
    sigref,hotref,hotmref,coldref     : SINGLE ;
    air,gain          : SINGLE;             {airmass}
```

```
tau_file          : datafile;         {opacity data file}
taufile           : PIECE;       {name of tau_file}
mon_file          : monfile;          {monitor data file}
monitorfile       : PIECE;       {name of mon_file}
header_file       : TEXT;             {ASCII file named STATUS.TXT}
flag              : BOOLEAN;          {worst-level flag}
zac,coldtemp      : SINGLE;           {zenith angle in DEGREES}
i,n,p,j,thismin     : INTEGER;
tau               : SINGLE;           {opacity}
slope,int         : SINGLE;           {intercept, corr.coeff.}
xx,yy,zz          : tau_array;        {airmass, ln(vsd), rms}
G                 : tau_array;
mm                : mon_array;        {analog monitors}
dmon, adr         : INTEGER;          {dig. monitor, adr. of flag}
tautime,montime   : INTEGER;           {set interval timings}
startdate         : PIECE;
achec,posit,stop  : BOOLEAN;
raw_data          : araw_data;
proc_data         : aproc_data;
tproc_data        : taproc_data;
b                 : byte;
airstep           : integer;
ambtemp,ix,iy,Tamb  : SINGLE;
sumx,sumy,sumw    : SINGLE;
w,sumxy,sumxsq    : SINGLE;
Tat,Tat1,gain2    : SINGLE;
adjust,adjust1    : SINGLE;
sigmatau,tauz     : SINGLE;
jy,jx,sigmasq     : SINGLE;
Gfudge,Tfudge     : single;
LOfudge           : single;
isdat,isphi       : piece;
phifreq           : single;
calfreq           : integer;
phicount_file     : text;
calibrate         : boolean;
decision          : integer;
sitenum           : char;   {site number}
gfreq             :integer;
```

```
    count             :integer;
    gaintip,tautip,sigmatautip,tauz2,ambtemptip :single;
    Vz,sigmaVz,GainZ,hottemp,Gcorr:SINGLE;
    cf : single;
    logfile:text;

{variables for I/O etc.}

    ch                : CHAR;
    ioctl_value       : INTEGER;
    com_base          : BYTE ABSOLUTE $40:4;
    mon_flag          : INTEGER;
    address_hi        : BYTE;
    nadr,comvar       : BYTE;


                { <- STDLIB1 / DAVID -> }

{includes support for both serial ports, using the installable device driver
 AUXDRV.COM which must be specified in the CONFIG.SYS file}

{The serial port for the VLBA interface board is COM1 and is given the highest
 priority for interrupts and is to be run at 9.6KB in binary mode.  COM2 is
 assigned the lowest priority and is to be used in ascii mode for talking to
 whatever it is supposed to talk to.

 COM1 must be at adr 3F8 and use irq4.
 COM2 must be at adr 2F8 and use irq3.

 These are the normal settings on all standard boards.}

{  IOCTL(flush,com1) and IOCTL(flush,com2) will clear the buffers.
   IOCTL(buf_count,com1) is used to check how many chars have been rcvd.
   IOCTL(ret_flag,com2)  is used to see if there is one or more carriage
                         return in the buffer.}

{  Use SET_COM1 or SET_COM2 before reading from or writing to a serial port.}

procedure caldelay;
```

```
var
  del,hour,minute,second,sec100:word;
  sec1,sec2,et:real;

begin
  gettime(hour,minute,second,sec100);
  sec1:=3600*hour+60*minute+second+sec100/100;
  delay(30000);
  gettime(hour,minute,second,sec100);
  sec2:=3600*hour+60*minute+second+sec100/100;
  cf:=30/(sec2-sec1);
  et:=sec2-sec1;
  writeln('cf = ',cf:8:3,' Elapsed time = ',et:8:2,' sec');
end;

PROCEDURE SLEEP(ms: longint);
  VAR
    newdel:longint;
    del:word;

  BEGIN
    newdel:=round(cf*ms);
    while (newdel > 20000) do
      begin
        delay(20000);
        newdel:=newdel-20000;
      end;
    del:=newdel;
    delay(del);
  END;

PROCEDURE IOCTL(function_nr,com_nr: INTEGER);
  {
  Ioctl communicates with the device driver COMDRV.COM which must
  be specified in the Config.Sys file when the PC is booted.
  function_nr specifies the sub_function, com_nr the com port.
  com_nr = 0 for COM1; com_nr = 1 for COM2.
  Ioctl(0,com_nr) is used to check how many chars have been rcvd.
```

```
  Ioctl(1,com_nr) is used to see if there is one or more carriage
                  return in the buffer.
  Ioctl(2,com_nr) sets the com port to ascii mode.
  Ioctl(3,com_nr) sets the com port to binary mode.
  Ioctl(4,com_nr) will clear the buffer.
  }

CONST
  DosFunc   = $44 ;            {DOS ioctl interrupt}
  SubFunc   = 2 ;              {Subfunction READ}
  NumBytes  = 2 ;              {No. bytes to be read}
  FileHandle= 3 ;              {3 is device AUX:}
VAR  r: registers ; (*regpack; not supported by TURBO 4.0 *)
     al,ah: BYTE;

BEGIN
  ioctl_value:=(function_nr * 2) + com_nr;
  {Load register fields for call to msdos function}
   r.ds:=SEG(ioctl_value);
    r.dx:=OFS(ioctl_value);
     r.cx:=NumBytes;
      al:=SubFunc ;  ah:=DosFunc ;
     r.ax:=(256*ah) + al;
    r.bx:=file_handle;
  msdos(r);
END;    {of procedure IOCTL}

{$I STDLIB2.PAS}

PROCEDURE SET_COM1 ;
BEGIN  com_base:=com1 ;              {com1=0}
END {SetCom1} ;

PROCEDURE SENDBYTE (DATA : BYTE ) ;
{Sends data to aux device}
BEGIN  WRITE(AUX,CHAR(DATA)) ; END ;


PROCEDURE SET_COM2;
```

```
BEGIN  com_base:=com2;  END;


PROCEDURE WAIT_COM1;
BEGIN REPEAT UNTIL (port[$3fd] AND 64) = 64;  {wait for tx buf empty}
END;


PROCEDURE SEND_PREFIX;
                       {sets even parity, sends prefix byte, sets odd parity}
                       {COM1 is the serial port being used}
BEGIN
  set_com1;
  wait_com1;
  PORT[$3fb]:=even;
  SENDBYTE(prefix) ;
  wait_com1;
  PORT[$3fb]:=odd;
END;     {of procedure SEND_PREFIX}


PROCEDURE SEND_CMD(cmdadr,cmdata_hi,cmdata_lo:byte);

VAR i: byte;  ch: CHAR;  cmdadr_hi: byte;

BEGIN
  cmdadr_hi := address_hi or $80;
   REWRITE(AUX) ; {opens write file to aux}
   ioctl(flush,com1);                {make sure buffer is empty}
    send_prefix;                     {send SYNC byte prefix}
     set_com1;                       {calls set_com1 & prepares port}
    Write(aux,char(cmdadr_hi));      {send address byte #1}
    Write(aux,char(cmdadr));         {send address byte #2}
    Write(aux,char(cmdata_hi));      {send data byte #1}
    Write(aux,char(cmdata_lo));      {send data byte #2}
     expected_length:=2;
      repeat
       begin
        sleep(1);
```

```
        IOCTL(buf_count,com1);        {buf_count is const @ 0 & calls ioctl}
        i := i +1;
       end;
      until ((ioctl_value >= expected_length) or (i = 10));
   CLOSE(AUX) ;   {closes write file to aux}

   RESET(AUX) ; {opens aux file so data can be read from the tipper}
     if ((i <= 10) AND (ioctl_value = expected_length)) then
    for i := 1 to ioctl_value do
   read(aux,ch);                      {reads ACK, DC1 in that order}
   CLOSE(AUX) ;  {This closes the aux file so it can be written or read from aga
in later}

END;  {of procedure SEND_CMD}



{*************************************************************************}


PROCEDURE RECEIVE_DATA(monadr:byte; var digval: single; var brdnc: boolean);

VAR i:byte; monch: array[1..3] of char; value: integer ;

BEGIN

  REWRITE(AUX) ;    {opens write file to aux}
IOCTL(flush,com1); send_prefix; set_com1;  {set up hardware for communication}
                                           {this trio is in send_cmd}

   Write(aux,char(address_hi));  {sends monit hi adres byte global: from MAIN}
    Write(aux,char(monadr));     {send monit low address byte}
     Write(aux,char($47)); Write(aux,char($48));  {required command bytes}
      expected_length := 3;
       i := 0;
  CLOSE(AUX) ;

  RESET(AUX) ;
  REPEAT
```

```
    BEGIN
      sleep(1);
      IOCTL(buf_count,com1);
      i:=i+1;
    END;
  UNTIL ((ioctl_value >= expected_length)  OR (i=10));

    If ((i =  10) OR (ioctl_value <> expected_length)) then brdnc := true
    else begin
      brdnc := false;
       For i:=1 to ioctl_value do
         Read(aux,monch[i]);  {reads ACK,MDH,MDL in that order}
         value := swap(integer(monch[2])) + integer(monch[3]);
                              {combine MDH & MDL}
         digval:= value*1.0 ; {this converts the negative integer into a
           negative real, direct conversion in TURBO 4.0 gives a large positive
          real number }
       End;
  CLOSE(AUX) ;
END;



PROCEDURE CONNECT_BOARD(var newbrd:boolean);

  Begin
    BAUD;
    port[$20] := $c3;
    IOCTL(set_binary,com1);
    address_hi := 0;
  End;  {procedure connect_board}



PROCEDURE RESPONDING(brdnc: boolean);

  Begin
    Window(50,1,80,2);
      gotoxy(1,1);
```

```
       if brdnc then write('No response from board')
      else clrscr;
   End;  {procedure responding}


PROCEDURE MONITOR_DATA(var raw_data:araw_data);

VAR nc:boolean; monadr:byte;

Begin
  RECEIVE_DATA($00,raw_data[1],nc);
  RESPONDING(nc);
  for i:=1 to 15 do
    RECEIVE_DATA(i,raw_data[i+1],nc);

End;  {procedure monitor_data}


PROCEDURE PROCESS_DATA(raw_data:  araw_data; var proc_data:  aproc_data;
                       var tproc_data: taproc_data);

VAR  b:byte;

Begin
    For b := 1 to 16 Do
      proc_data[b] := 5.000*(raw_data[b]/16);   {converts to mvolts}

  tproc_data[1] := proc_data[1]/20;             {sig-ref (switched output)}
  tproc_data[2] := proc_data[2]/200;            {hot-ref (gain monitor)}
  tproc_data[3] := proc_data[3]/2 + 5000;       {ref (total power)}
  tproc_data[4] := proc_data[4]/100;            {ref temp}
  tproc_data[5] := proc_data[5]/100;            {hot temp}
  tproc_data[6] := proc_data[6]/100;            {ambient temp}
  if (tproc_data[6]<-40) or (tproc_data[6]>40) then tproc_data[6]:=0;
  {This is a safeguard against the tendency for the temperature probe
  on the radiometer to break of malfunction}
  tproc_data[7] := proc_data[7]/100;            {chassis temp}
  tproc_data[8] := proc_data[8]/-1000;          {mixer current}
  tproc_data[9] := proc_data[9]/-1000;          {tripler current}
```

```
  tproc_data[10] := proc_data[10]/10000;        {gunn current}
  tproc_data[11] := proc_data[11]/500;          {supply voltage}
  tproc_data[12] := proc_data[12]/50;           {zenith angle}
  tproc_data[13] := proc_data[13]/1000;         {supply current}
  tproc_data[14] := proc_data[14]/13.89;        {wind direction}
  tproc_data[15] := proc_data[15]/25;           {wind speed}
  tproc_data[16] := proc_data[16]/500;          {weather station supply volts}
End;  {procedure process_data}


PROCEDURE DIRECT; {sends command to determine direction of mirrror rotation}
                        {clockwise or counterclockwise}
  Begin
    if posit then begin   {If posit is true, rotation direction is positive}
      comvar := comvar and cwzero;
       SEND_CMD(state_adr,0,comvar);    {sends a zero(mirror rotates CCW)}
       sleep(100);
     end
    else begin
      comvar := comvar or cw;
       SEND_CMD(state_adr,0,comvar);    {sends a one(mirror rotates CW)}
     end;
   sleep(100);    {allow time for command}
  End;  {procedure direct}


PROCEDURE GZEN;  {sends command to change mirror orientation to zenith}

VAR  y:integer;

 Begin
  zac := 0;
   comvar := comvar or gozen;
    SEND_CMD(state_adr,0,comvar);  {GZ bit=1,command sent to go zenith}
    sleep(100);
   comvar := comvar and gozero;
    SEND_CMD(state_adr,0,comvar);  {GZ bit reset to 0}
  sleep(16000);   {time delay to allow mirror to travel to zenith}
 End;  {procedure GZEN}
```

```
PROCEDURE ELEVATE(astep:integer);  {steps motor by 1.8 degree steps}

  VAR q:integer;

  BEGIN
      for q := 1 to astep do begin
       comvar := comvar or step;
        SEND_CMD(state_adr,0,comvar);      {sends command}
         sleep(100);
          comvar := comvar and stzero;
           SEND_CMD(state_adr,0,comvar);    {sends reset of zero}
          sleep(100);
       end;
  END;    {procedure ELEVATE}


PROCEDURE STEPBACK;

  Begin
    airstep := 39;
    posit := false;
    DIRECT;
    ELEVATE(airstep);
    sleep(1000);
  {  GZEN;}
  END;   {procedure stepback}


PROCEDURE DISPLAY_DATA( proc_data: aproc_data; tproc_data:taproc_data);

VAR b:byte; a:integer;

  Begin
    Window(1,1,80,25);Highvideo;
    b := 1;
    For b := 1 to 16 do begin
      a := b + 4;
```

```
    gotoxy(22,a); write(proc_data[b]:8:2);
    gotoxy(39,a); write(tproc_data[b]:9:3);

    end;
    b := 1;
 End;     {procdure display_data}

PROCEDURE SCREEN;      {generates CRT heading for tipper data}

 Begin
   WINDOW(1,1,80,25);
   CLRSCR ;
   GOTOXY(1,20); WRITE('
  ');
   gotoxy(4,1);  write('ZENITH ANGLE  =  ');
   gotoxy(4,3);  write('#     ITEM');
   gotoxy(20,3); write('    mVOLTS');
   gotoxy(41,3); write('  SCALED');
   gotoxy(4,5);  write('0    SIG-REF');   gotoxy(48,5);  write('K');
   gotoxy(4,6);  write('1    HOT-REF');   gotoxy(48,6);  write('K');
   gotoxy(4,7);  write('2      REF  ');   gotoxy(48,7);  write('K');
   gotoxy(4,8);  write('3    REF TEMP');  gotoxy(48,8);  write('C');
   gotoxy(4,9);  write('4    HOT TEMP');  gotoxy(48,9);  write('C');
   gotoxy(4,10); write('5    OUT TEMP');  gotoxy(48,10); write('C');
   gotoxy(4,11); write('6    IN TEMP');   gotoxy(48,11); write('C');
   gotoxy(4,12); write('7    MXR CUR');   gotoxy(48,12); write('mA');
   gotoxy(4,13); write('8    TRIP CUR');  gotoxy(48,13); write('mA');
   gotoxy(4,14); write('9    GUNN CUR');  gotoxy(48,14); write('A');
   gotoxy(4,15); write('10   SUP VOLTS'); gotoxy(48,15); write('V');
   gotoxy(4,16); write('11   ZEN ANG');   gotoxy(48,16); write('deg');
   gotoxy(4,17); write('12   SUP CUR');   gotoxy(48,17); write('A');
   gotoxy(4,18); write('13   WIND DIR');  gotoxy(48,18); write('deg');
   gotoxy(4,19); write('14   SPEED');     gotoxy(48,19); write('mph');
   gotoxy(4,20); write('15   WS SUP VOLTS'); gotoxy(48,20); write('V');
   gotoxy(70,23);
End;  {procedure SCREEN}

PROCEDURE START;
```

```
  var open : byte;

  Begin
    SCREEN;
    open := $04;
    SEND_CMD(state_adr,0,open);
    sleep(100);     {allow time for command}
  End;   {procedure START}



FUNCTION EXIST(nameoffile: string20): BOOLEAN;
          {TURBO manual p 96}
   VAR
   file1: FILE;
   temp:boolean;

   BEGIN
     ASSIGN(file1, nameoffile);
     {$I-}
     RESET(file1);
     {$I+}
     temp:=(IOresult = 0);
     if temp then close(file1);
     exist:=temp;

   END;   {function EXIST}

FUNCTION WHATFILE(answer : boolean): string16; {gets next available file name}
VAR
   t4: string[2];
   t0: INTEGER;

BEGIN
   t0:=0; t4:='00';
   WHILE (answer and EXIST(CONCAT('c:\data\',date2,t4,'.tx',sitenum))) DO BEGIN
{if answer is true then phitest is activated and search for '.TXT' is needed}
 {the truth table is correct if you short circuit the boolean expression}
```

```
    t0:=t0+1;
    STR(t0,t4);
    WHILE LENGTH(t4) < 2 DO t4:=concat('0',t4);
    END;
   WHATFILE:=CONCAT('C:\DATA\',DATE2,t4);
END;   {function whatfile}

FUNCTION WHATFILE2 : string16; {gets next available filename}

VAR
  t4: string[2];
  t0: INTEGER;

BEGIN
  t0:=0;  t4:='00';
  WHILE (EXIST(concat('c:/data/',date2,t4,'.da',sitenum))) DO
     BEGIN
     t0:=t0+1;
     STR(t0,t4);
     WHILE LENGTH(t4) < 2 DO t4:=concat('0',t4);
     END;
  WHATFILE2:=CONCAT('c:\data\',date2,t4);
END;  {function whatfile2}



PROCEDURE SAVE_MON(mm : mon_array; dmon : INTEGER);

  VAR
    mon_rec : rec2;          {dmon is not used in cvtipper}
                             {dmon allows consistency when using monread}
  BEGIN                      {monread reads monitoring data files}
    with mon_rec DO
      BEGIN
        time1 := time;
        m1    := mm;
        dmon2 := dmon;
      END; {* with *}
    RESET(mon_file);
```

```
    SEEK(mon_file,FILESIZE(mon_file));
    WRITE(mon_file,mon_rec);
    CLOSE(mon_file);
  END;   {procedure SAVE_MON}


PROCEDURE MON_CHECK(VAR mm:mon_array; VAR dmon:integer);

                   {mon_check sends the mirror to zenith and takes data}
Begin              {which is stored in a monitor data file}
  dmon := 0;
    If check then begin
      for b := 1 to 16 do begin
       raw_data[b] := 16*b/5;
       PROCESS_DATA(raw_data,proc_data,tproc_data);
      mm[b] := tproc_data[b];
      end;
      if (exist('c:\procomm\lock.txt')) then sleep(120000);
     SAVE_MON(mm,dmon);
    end
   Else begin
    MONITOR_DATA(raw_data);
     PROCESS_DATA(raw_data,proc_data,tproc_data);
      for b := 1 to 16 do begin
      mm[b] := tproc_data[b];
      end;
      if (exist('c:\procomm\lock.txt')) then sleep(120000);
     SAVE_MON(mm,dmon);
    end;
End;   {procedure MON_CHECK}


PROCEDURE ANALOG(etyb1:BYTE;zac1:SINGLE);

Begin
  If check then Begin
   raw_data[etyb1] := 16*EXP(-1/COS(PI*zac1/180));
    for b := 2 to 16 do begin
     raw_data[b] := 16*b/5;
```

```
    end;
     PROCESS_DATA(raw_data,proc_data,tproc_data);
     DISPLAY_DATA(proc_data,tproc_data);
   End
 Else
     MONITOR_DATA(raw_data);
     PROCESS_DATA(raw_data,proc_data,tproc_data);
     DISPLAY_DATA(proc_data,tproc_data);
     coldtemp := tproc_data[4] + 273;
     ambtemp := tproc_data[6] + 273;
     hottemp := tproc_data[5] +273;
     gain := (tproc_data[5] - tproc_data[4])/tproc_data[2];
End;   {procedure ANALOG}

FUNCTION VOLTAGE(etyb1:BYTE):SINGLE;

Begin
 ANALOG(etyb1,zac);
 voltage := - proc_data[etyb1]/20;
End;   {procedure VOLTAGE}


PROCEDURE AVERAGE(etyb1:BYTE;num_reads:INTEGER; VAR mean,rms:SINGLE);

VAR read:ARRAY[1..100] of SINGLE;

Begin
  mean := 0.0;  rms := 0.0;
    For n := 1 to num_reads Do Begin
     read[n] := voltage(etyb1);
     mean := mean + read[n]/num_reads;
     sleep(100);
    End;
  For n := 1 to num_reads do rms := rms+(read[n]-mean)*(read[n]-mean);
   rms := SQRT(rms/num_reads);
End;   {procedure AVERAGE}


PROCEDURE MEASURE(VAR x, y, z : tau_array);
```

```
{* Reads voltages at various elevations. *}

VAR  i : INTEGER;
     vsd, rms : SINGLE;

BEGIN
      zac := 0.0;

      For i := 0 to num_airs - 1  Do
        Begin

          Case i of
           0  : airstep :=  4;
           1  : airstep := 15;
           2  : airstep :=  6;
           3  : airstep :=  4;
           4  : airstep :=  3;
           5  : airstep :=  2;
           6  : airstep :=  1;
           7  : airstep :=  1;
           8  : airstep :=  1;
           9  : airstep :=  1;
           10 : airstep :=  1;
          End;    {of case}

          Case i of
           0  : zac :=  7.2;
           1  : zac := 34.2;
           2  : zac := 45.0;
           3  : zac := 52.2;
           4  : zac := 57.6;
           5  : zac := 61.2;
           6  : zac := 63.0;
           7  : zac := 64.8;
           8  : zac := 66.6;
           9  : zac := 68.4;
           10 : zac := 70.2;
          End;    {of case}
```

```
        if posit then begin
          zac := -zac;
         end
        else zac := zac;
        gotoxy(20,1); write(zac:4:1);
          elevate(airstep);
           sleep(10000);
              average(1,num_reads,vsd,z[i]);
              G[i]:=gain;
(*
 * changed by GSH on 4-17-87 to prevent log of negative numbers
 *)
              IF vsd <= 0.0 Then y[i] := -5.0 ELSE
                y[i] := LN(vsd);
              x[i] := 1/COS(pi*zac/180.0);
        END; {* for i *}

END;  {procedure MEASURE}


Procedure ERROR(x3,y3 : tau_array; constant1 : single);

VAR value : single;

Begin
  sigmasq := 0;
  For j := 0 to 10 Do Begin
    w := EXP(y3[j]) - constant1;
    if w <= 0 then w := default;
    jy := LN(w);
    jx := x3[j];
    value := int + slope*jx;
    sigmasq := sigmasq + (jy - value)*(jy - value);
  End;    {of j}
End;   {of procedure ERROR}


Procedure SUM( x,y : tau_array ; constant : single);
```

```
VAR
  prodxyw,prodxy,prodxsqw,prodxx,prodxsqy,prodxxy,sl,other : DOUBLE;

Begin

  sumw := 0;          sumxy  := 0;
  sumx := 0;          sumxsq := 0;
  sumy := 0;

  For i := 0 to 10 Do Begin

    w := EXP(y[i]) - constant;
    If w <= 0 then w := default;
    iy := LN(w);
    ix := x[i];

    sumw   := sumw + w;
    sumx   := sumx + ix*w;
    sumy   := sumy + iy*w;
    sumxy  := sumxy + ix*iy*w;
    sumxsq := sumxsq + ix*ix*w;

  End;         {of i}

  prodxyw:=sumxy*sumw; prodxy:=sumx*sumy; prodxsqw:=sumxsq*sumw;
  prodxx:=sumx*sumx; prodxsqy:=sumxsq*sumy; prodxxy:=sumx*sumxy;
  IF (prodxsqw-prodxx) = 0 THEN PRODXX:=0.9*PRODXSQW;
  sl := (prodxyw-prodxy)/(prodxsqw-prodxx);
  other := (prodxsqy-prodxxy)/(prodxsqw-prodxx);
  slope:=sl;
  int:= other;

End;    {of procedure SUM}


Procedure REGULAR(x4,y4 : tau_array; manipulate : single);

Begin
  SUM(x4,y4,manipulate);
```

```
  ERROR(x4,y4,manipulate);
End;   {procedure REGULAR}


Procedure SUBADJUST;

VAR delta1 : SINGLE;

Begin

  gain2 := Gfudge*(1/gain);
  if gain2 = 0.0 then tau := -99.0
   else begin
   Tat1 := ambtemp - Tfudge;
   {Tat1 := 260;}
    adjust1 := gain2*(coldtemp - Tat1) +  LOfudge;
     SUM(xx,yy,adjust1);
     Tat := EXP(int)/gain2;
    adjust := gain2*(coldtemp - Tat) +  LOfudge;

      If adjust < 0 then REGULAR(xx,yy,adjust1)
        else REGULAR(xx,yy,adjust);

   delta1 := (sumw*sumxsq) - (sumx*sumx);
   sigmatau := SQRT((sumw*sigmasq)/((num_airs - 2)*delta1));
  tau := -slope;
  end;

End;   {procedure subadjust}

Procedure SUBADJUST1;

VAR delta1 : SINGLE;

Begin

  gain2 := gain;
  if gain2 = 0.0 then tau := -99.0
   else begin
```

```
   Tat1 := ambtemp - Tfudge;
   {Tat1 := 260;}
    adjust1 := gain2*(coldtemp - Tat1) +  LOfudge;
     SUM(xx,yy,adjust1);
     Tat := EXP(int)/gain2;
    adjust := gain2*(coldtemp - Tat) +  LOfudge;

      If adjust < 0 then REGULAR(xx,yy,adjust1)
        else REGULAR(xx,yy,adjust);

   delta1 := (sumw*sumxsq) - (sumx*sumx);
   sigmatau := SQRT((sumw*sigmasq)/((num_airs - 2)*delta1));
  tau := -slope;
  end;

End;   {procedure subadjust1}


PROCEDURE SAVE(xx,yy,zz : tau_array; tau,sigmatau,tauz,gain,ambtemp,
               coldtemp : SINGLE);

  VAR
    tau_rec : rec;

  BEGIN
    with tau_rec DO
      BEGIN
        time1 := time;
        tau1  := tautip;
        sigmatau1 := sigmatautip;
        tauz1 := tauz;
        Vz1 := Vz;
        GainZ1:=GainZ;
        sigmaVZ1:=sigmaVZ;
        gain1  := gaintip;
        tauI1:=tau;
        sigtauI1:=sigmatau;
        gainI1:=gain;
```

```
        Tamb1 := ambtemptip;
        Tc    := coldtemp;
        Th:=hottemp;
        x1    := xx;
        y1    := yy;
        z1    := zz;
        G1:=G;
      END; {* with *}
    RESET(tau_file);
    SEEK(tau_file,FILESIZE(tau_file));
    WRITE(tau_file,tau_rec);
    CLOSE(tau_file);
  END;   {procedure SAVE}


PROCEDURE MAKE_HEADER;
VAR i:integer;

BEGIN
  ASSIGN(header_file,'C:STATUS.TXT');
  REWRITE(header_file);
  WRITELN(header_file,'Software version: ',softversion);
  WRITELN(header_file,'Hardware version: ',hardversion);
  WRITELN(header_file,num_reads,' points averaged at each airmass.');
  WRITELN(header_file,num_airs,' airmasses measured.');
  WRITELN(header_file);
  CLOSE(header_file);
END;   {procedure MAKE_HEADER}

PROCEDURE MAKE_LINE (title : STRING9 ;xyz : tau_array);

VAR i : INTEGER;

BEGIN
  WRITE(title);
  FOR i:=0 to num_airs - 1 DO WRITE(xyz[i]:5:2,' ');
  WRITELN(' ');
END;   {procedure MAKE_LINE}
```

```
Procedure ZENTAU;

var read:ARRAY[1..100] of SINGLE;
var NL,mean,rms:SINGLE;

Begin
  mean:=0.0; rms:=0.0;
  for n:=1 to num_reads do begin
    read[n]:=voltage(1);
    mean:=mean+read[n]/num_reads;
    sleep(100);
  end;
  for n:=1 to num_reads do rms:=rms+(read[n]-mean)*(read[n]-mean);
  rms:=SQRT(rms/num_reads);
  gainz :=  Gfudge/gain;
  NL := (mean/gainz + LOfudge + (ambtemp - coldtemp) )/ambtemp;
  If NL <= 0 then tauz := 0
    else tauz := -LN(NL);
  Vz:=mean;sigmaVz:=rms;
End;   {of procedure ZENTAU}

PROCEDURE CHOICE;

BEGIN

 {Writeln('Choose one of the following one hour observations.');
  Writeln('1) 3.5 second interval, 32 readings');
  Writeln('2) 7 second interval, 512 readings');
  Writeln('3) 14 second interval, 256 readings');
  Readln(run);}

  run := 1;

  If run = 1 then begin
    num_avg := 1024;
    interval := 50;
    TD1 := 380;
    TD2 := 365;
```

```
  end;

  If run = 2 then begin
    num_avg := 512;
    interval := 100;
    TD1 := 997;
    TD2 := 990;
  end;

  If run = 3 then begin
    num_avg := 256;
    interval := 200;
    TD1 := 1998;
    TD2 := 1990;
  end;
END; {of procedure CHOICE}

FUNCTION SECONDS:real;

var
   hour,minute,second,sec100 : word;

begin
   gettime(hour,minute,second,sec100);
   seconds:=hour*3600+minute*60+second+sec100/100;
end; { SECONDS }


FUNCTION HSECONDS : Integer;

Var
    t             : registers;
    secs,hsecs    : integer;

Begin
  t.ax := 256*$2C;
  intr($21,t);
  secs := hi(t.dx);
  hsecs := lo(t.dx);
```

```
  HSECONDS := 100*secs + hsecs;
End;        {of function HSECONDS}



PROCEDURE TIMESLOT;

Begin
  time1 := HSECONDS;
  time2 := time1 + interval;
End;      {of procedure TIMESLOT}



PROCEDURE TIMEOUT;

Begin
  If time2 >= 6000 then begin
   time3 := time2 - 6000;
    Repeat
     negtime := HSECONDS - time3;
      if negtime <= 0 then checktime := HSECONDS
     else checktime := -100;
    Until checktime >= time3;
  end
  else begin
    Repeat
     checktime := HSECONDS;
    Until checktime >= time2;
  end;
End;     {of procedure TIMEOUT}

PROCEDURE PHI_MONITOR_DATA(var raw_data: araw_data) ;

VAR nc : boolean ;

BEGIN
  RECEIVE_DATA($00,raw_data[1],nc) ;  {sig-ref}
  RESPONDING(nc) ;
```

```
  RECEIVE_DATA($01,raw_data[2],nc) ;  {hot-ref}
  RECEIVE_DATA($03,raw_data[3],nc) ;  {ref}
  RECEIVE_DATA($04,raw_data[4],nc) ;  {hot}
END ; {of procedure PHI_MONITOR_DATA}

PROCEDURE PHI_PROCESS_DATA(raw_data : araw_data; VAR proc_data : aproc_data;
                           VAR tproc_data : taproc_data) ;
VAR b : byte ;

BEGIN
 For b := 1 to 4 do
   proc_data[b] := 5.000*(raw_data[b]/16) ;  {converts to mVOLTS}
 {note: The subscripts on the following variables do not correspond to the
        same subscripts in procedure PROCESS_DATA.  See PHI_MONITOR_DATA
        for details.}
 tproc_data[1] := proc_data[1]/20 ; {sig-ref (switched outputs)}
 tproc_data[2] := proc_data[2]/200 ; {hot-ref (gain monitor)}
 tproc_data[3] := proc_data[3]/100 ; {ref temp}
 tproc_data[4] := proc_data[4]/100 ; {hot temp}
END ;  {of procedure PHI_PROCESS_DATA}

PROCEDURE PHISTEPBACK ;

BEGIN
  airstep := 10;
  ELEVATE(airstep) ;
  sleep(1000);
END ;  {of procedure PHISTEPBACK}

PROCEDURE PHIMEASURE(VAR w,x,y,t :   phi_array);

var
   i,j       : integer;
   t1,t0,t00 : real;
   fred      :  text;

BEGIN
   Writeln('Measurement beginning at ',stop_watch,' on ',date2);
   For j := 1 to num_avg Do Begin
```

```
      t0:=seconds;
      t00:=t0;
      w[j]:=0;x[j]:=0;y[j]:=0;t[j]:=0;
      For i := 1 to phi_num_reads Do Begin
         PHI_MONITOR_DATA(raw_data);
         PHI_PROCESS_DATA(raw_data,proc_data,tproc_data);
         w[j] := w[j] + tproc_data[1];
         x[j] := x[j] + tproc_data[2];
         y[j] := y[j] + tproc_data[4] - tproc_data[3];
         t[j] := t[j] + tproc_data[3] ;
         if i<phi_num_reads then
         begin
            repeat
               t1:=seconds;
               if t1-t0<0 then t1:=t1+86400;
            until(t1-t0>=0.5);
            t0:=t1;
         end;
      End;
      w[j] := w[j]/phi_num_reads;
      x[j] := x[j]/phi_num_reads;
      y[j] := y[j]/phi_num_reads;
      t[j] := t[j]/phi_num_reads;
      repeat
         t1:=seconds;
         if t1-t0<0 then t1:=t1+86400;
      until(t1-t00>=3.51);
      t00:=t1;
      t0:=t1;
   End;
   Writeln('Measurement complete at ',stop_watch);
END;  {of procedure PHIMEASURE}

PROCEDURE PHISAVE(sig1,hot1,ref1,cold1: phi_array);

VAR  phase_rec : phirec;

BEGIN
  with phase_rec DO BEGIN
```

```
       w2 := sig1;    {sigref}
       x2 := hot1;    {hotref}
       y2 := ref1;    {hotref-coldref}
       t2 := cold1;   {cold load}
     End;
  RESET(phase_file);
  SEEK(phase_file,FILESIZE(phase_file));
  WRITE(phase_file,phase_rec);
  CLOSE(phase_file);
END;   {of procedure PHISAVE}

PROCEDURE CONVERT ;

VAR   c,sides                : Integer ;
      phase_rec              : phirec ;
      dev                    : text ;
      Q                      : String[4] ;
      z                      : Single ;
      gavg,gsum              : Single ;

BEGIN  {of procedure convert}
      fil2 := CONCAT(COPY(phasefile,1,LENGTH(phasefile)-3),'TX',sitenum);
      Assign(dev,fil2);
      Rewrite(dev);
      if calibrate then writeln(dev,'Calibration Run');
      Assign(phase_file,phasefile);
      Reset(phase_file);
      Read(phase_file,phase_rec);
        With phase_rec Do Begin
{
          w2[1]:=w2[2];
}          {This is a temporary fix for a timing problem in the calibration
          measurements.  It should eventually be replaced by a longer delay
          after sending the mirror to 180 degrees to allow transient signals
          to die down.}

          gsum:=0;
          sides:=gfreq div 2;
          for c:=1 to gfreq+1 do gsum:=gsum+y2[c]/(x2[c]*GFudge);
```

```
          for c:=1 to 1024 do begin
            if (c>1+sides) and (c<1025-sides) then
              gsum:=gsum-(y2[c-1-sides]/(x2[c-1-sides]*GFudge))
                +(y2[c+sides]/(x2[c+sides]*GFudge));
            gavg:=gsum/(gfreq+1);
            z := (w2[c]*gavg) + t2[c] + 273.0 ;
            Writeln(dev,z:6:2);
          end;  {for}
        End;  {of with}
  Close(phase_file) ;
  erase(phase_file) ;
  Close(dev)
END;   {of PROCEDURE CONVERT}

PROCEDURE ALLAN_88 ;

VAR
      dev                       : text;
      num_avg                   : Integer;
      num,number,space          : Integer;
      diff,sigma,deviation,avg  : Single;
      point,temp                : phi_array;
      sigmaA,time               : Single;
      Tb                        : Single;
      hour,min,sec,sec100       : word ;
      junk                      : string[40];

  Procedure SUMMATION(pp : Integer; w : phi_array; VAR sigma : Single);

  VAR count   : integer ;

  Begin
    sigma := 0.0;
    FOR count:= 1 to pp-2 Do begin
      diff := ((w[count+2] + w[count])/2) - w[count+1];
      sigma := sigma + diff*diff;
    End;  {for}
    sigma := 2*sigma/(3*(pp-3));
    if sigma>0 then deviation:=sqrt(sigma)
```

```
    else deviation:=0; {note: this number is not passed thru the parm list}
  End; {of procedure SUMMATION}

  Procedure LONG_TIME(aa,cc,num_points : integer; x : phi_array);

  Begin
    space := cc;
    For n := 1 to num_points Do Begin
      avg := 0;
      for i := aa to cc do avg := avg + x[i];
      point[n] := avg/space;
      aa := aa + space;
      cc := cc + space;
    End; {for}
    SUMMATION(num_points,point,sigmaA);
    time:=space*3.5;
    Writeln(data2file,sigmaA:13:5,' ',deviation:8:3,' ',time:4:0);
  End; {of procedure LONG_TIME}

BEGIN   {of procedure allan_87}
  num_avg := 1024;
  fil1 := CONCAT(COPY(fil2,1,length(fil2)-3),'ou',sitenum);
  Assign(data2file,fil1) ;
  Rewrite(data2file) ;
  Assign(data1file,fil2);
  Reset(data1file);
  i := 1;
  if calibrate then begin
    writeln(data2file,'Calibration Run');
    readln(data1file,junk); {read calibrate label}
  end;
  Repeat
    Readln(data1file,Tb);
    temp[i] := Tb;
    i := i + 1;
  Until EOF(data1file);
  num := 1;
  number := num_avg;
  gettime(hour,min,sec,sec100) ;
```

```
  if (hour < 10) then write(data2file,'0',hour:1,':')
  else write(data2file,hour:2,':');
  if (min < 10) then writeln(data2file,'0',min:1)
  else writeln(data2file,min:2);
  LONG_TIME(1,num,number,temp);
    Repeat
      num_avg := num_avg div 2;
      num := num*2;
      LONG_TIME(1,num,num_avg,temp);
    Until num_avg = 4;
  Close(data1file) ;
  Close(data2file) ;
END ;   {of procedure ALLAN_88}

function cal:boolean;
{checks to see if a calibration phitest should be performed instead of a
sky test.}
var
  number      :integer;
begin
  cal:=false;
  if calfreq>0 then begin
    assign(phicount_file,phicount);
    reset(phicount_file);
    read(phicount_file,number);
    close(phicount_file);
    if number>=calfreq then begin
      cal:=true;
      number:=0;
    end;
    number:=number+1;
    rewrite(phicount_file);
    writeln(phicount_file,number);
    close(phicount_file);
  end;
end;  {function cal}

PROCEDURE PHITEST ;
```

```
(* This procedure measures fluctuations in the sky temperature and the
   rms in sky temperature with the Charlottesville radiometer. The data
   will be analyzed to see if phase stability can be related to sky
   fluctuations. *)

VAR  newbrd : boolean;  num : integer;  length : single;
     go_on  : string[2]; m : integer ;

BEGIN
        ISPHITEST := TRUE ;   {phitest is on}
        startdate := date2;
        phasefile := whatfile(ISPHITEST);
        phasefile := CONCAT(phasefile,'.PHI');
        ASSIGN(phase_file, phasefile);
        REWRITE(phase_file);
        CLOSE(phase_file);
        CONNECT_BOARD(newbrd);
        GZEN;
{***        Writeln('Install absorber - type go when ready');
        Readln(go_on);
        If go_on = 'go' then writeln('Here we go!');  ***}
        CHOICE;
        calibrate:=(calfreq>0) and (cal);
        if calibrate then begin
          elevate(100);
          sleep(20000);
        end;
        PHIMEASURE(sig1,hot1,ref1,cold1);
        for m := 1 to num_avg do begin
           writeln(m,'  ',sig1[m]:8:3,'  ',hot1[m]:8:3,'  ',ref1[m]:8:3);
        end;
        PHISAVE(sig1,hot1,ref1,cold1);
        STEPBACK;
        CONVERT ;
        ALLAN_88 ;
 END;   {of procedure PHITEST}

FUNCTION PHIDEFAULT: BOOLEAN ;
{ This function is the default action which is implemented if decision in
```

```
  the file ‘‘phitest.txt’’ is equal to 0.  This function will be true if
  the number of Julian hours is a multiple of phifreq read from fudge.pas.
  In addition, calibrate will be set to true one out of every calfreq
  phitests}

VAR
  year,month,day,day_of_week  : word ;
  hour,minute,second,sec100   : word ;
  Julianday,Julianhour        : REAL;
  remainder                   : REAL ;
  prejunk                : longint ;

BEGIN
  phidefault := FALSE;
  if phifreq>0 then begin
    GETDATE(year,month,day,day_of_week) ;
    GETTIME(hour,minute,second,sec100) ;
  (* Julian day 2440000 began at noon of May 23, 1968. *)
    Julianday := 365.25*(year-1968) + 30.6001*(month - 5) + 1.0*(day - 23) ;
    Julianday := Julianday + 2440000.0 ;
    Julianhour := Julianday*24.0 + hour*1.0 + minute/60.0 + second/3600.0 ;
    Julianhour := Julianhour + 12.0 ;
    { this accounts for the fact that the Julian day begins at noon }
    prejunk := TRUNC(Julianhour/phifreq) ;
  (* intermediate step in determining remainder *)
    remainder := Julianhour - phifreq*prejunk ;
    IF ((remainder >= 0.0) and (remainder <= 0.5)) then phidefault := TRUE ;
  end;
END ;  {of function PHIDEFAULT}

FUNCTION PHICHECK : BOOLEAN;
{ This function will check the file ‘‘a:phitime.txt’’ for phitest times. If the
  actual time is equal to or greater than the file time by 30 minutes, the
  procedure PHITEST will be implemented. JJH, 11 July 1988 }

VAR
  testday                          : string[6] ;
  timenow,testtime                 : string[4] ;
  blank                            : string[1] ;
```

```
  realday,realtime,today,nowtime    : SINGLE ;
  m,n,p,q                           : integer ;
  hour,minute,second,sec100         : word ;

BEGIN
  phicheck := FALSE ;
  while not EOF(phitest_file) do begin
     readln(phitest_file,testday,blank,testtime) ;
     VAL(testday,realday,m) ;
     VAL(testtime,realtime,n) ;
     VAL(DATE2,today,p) ;
     IF today = realday then BEGIN
       GETTIME(hour,minute,second,sec100) ;
       nowtime := hour*100.0 + minute*1.0 ;
       IF (nowtime >= realtime) and (nowtime <= (realtime + 30.0)) then
         phicheck := TRUE
       END ;         {if}
     END ;       {while}
  Close(phitest_file) ;
END ;   {of function PHICHECK}

{******************** MAIN PROGRAM ***************************}

VAR
        newbrd        : boolean;
        Q             : STRING[5];
        filesave        : integer ;

BEGIN
    caldelay;
    ASSIGN(AUX,'AUX') ;

    Assign(fudge_file,fil);
    Reset(fudge_file);
    Readln(fudge_file,Gfudge);
    readln(fudge_file,Tfudge);
    readln(fudge_file,LOfudge);
    readln(fudge_file,phifreq);
    readln(fudge_file,calfreq);
```

```
  readln(fudge_file,gfreq);
  read(fudge_file,sitenum);
  close(fudge_file);
  if 1024 mod gfreq <>0 then gfreq:=256;
  Writeln('Gfudge = ',Gfudge:4:2,'   Tfudge = ',Tfudge:4:0);
  Writeln('LOfudge =',LOfudge:4:0,'  PhiFreq =',phifreq:5:2);
  Writeln('calfreq =',calfreq,'  GFreq =',gfreq);
  writeln('sitenum =',sitenum);
  sleep(1000);

  Assign(phitest_file,phitime) ;
  Reset(phitest_file) ;
  readln(phitest_file,decision) ;
  readln(phitest_file,filesave); {if filesave=0 the converted data is erased
                                  and only the allan variance file remains}

  {This part of the program decides how to check if it is time for a phitest
  and then calls the procedure to do the checks.}
  if decision=0 then begin
    Close(phitest_file) ;
    IF PHIDEFAULT then PHITEST;
    end
  else if decision=1 then begin
    IF PHICHECK then PHITEST ;
    end
  else if decision=2 then begin
    if phicheck or phidefault then phitest;
    end
  else writeln('Illegal value for phitype.  Phitest deactivated.');
   { decision can be set to different integers for different options }
  IF (filesave=0) then erase(data1file) ;

ASSIGN(GLOB1,'C:\DATA\GLOB1.INT');
ASSIGN(GLOB2,'C:\DATA\GLOB2.REA');
ASSIGN(GLOB3,'C:\DATA\GLOB3.BOO');
ASSIGN(GLOB4,'C:\DATA\GLOB4.BYT');
ASSIGN(GLOB5,'C:\DATA\GLOB5.STR');

ASSIGN(STORBOOL,'C:STORBOOL.DAT');
```

```
RESET(STORBOOL);
READ(STORBOOL,WHATISIT);
CLOSE(STORBOOL);

IF NOT WHATISIT THEN
  begin
    WRITELN('THE SYSTEM WAS JUST BOOTED.');
    rewrite(storbool);
    WHATISIT:=TRUE;
    write(storbool,WHATISIT);
    close(storbool);
  end

ELSE BEGIN
  WRITELN;

  RESET(GLOB1);
  READ(GLOB1,i,n,p,j,thismin,dmon,adr,tautime);
  READ(GLOB1,montime,airstep,ioctl_value,mon_flag);
  CLOSE(GLOB1);

  RESET(GLOB2);
  READ(GLOB2,air,gain,zac,coldtemp,tau,slope,int);
  READ(GLOB2,ambtemp,ix,iy,Tamb,sumx,sumy,sumw,w,sumxy,sumxsq);
  READ(GLOB2,Tat,Tat1,gain2,adjust,adjust1,sigmatau,tauz,jy,jx,sigmasq);
  FOR NEWI:= 1 TO 10 DO
    READ(GLOB2,xx[NEWI],yy[NEWI],zz[NEWI]);
  FOR NEWI:= 1 TO 16 DO
    READ(GLOB2,mm[NEWI],raw_data[NEWI],proc_data[NEWI],tproc_data[NEWI]);
  CLOSE(GLOB2);

  RESET(GLOB3);
  READ(GLOB3,flag,achec,posit,stop);
  CLOSE(GLOB3);

  RESET(GLOB4);
  READ(GLOB4,b,address_hi,nadr,comvar,com_base);
  CLOSE(GLOB4);
```

```
    RESET(GLOB5);
    READ(GLOB5,taufile,monitorfile,startdate);
    CLOSE(GLOB5);

    TIPPING:= FALSE;
    ASSIGN(TAU_FILE,TAUFILE);
    ASSIGN(MON_FILE,MONITORFILE);
    GOTO beta;
  END;  { ELSE }

    WRITELN(prog,' Version : ',version,' ',revdate);

 alpha: WRITELN('Automatic tipping scans beginning ',time,' UT  ',date2);

        startdate:=date2;
        IF NOT exist('C:STATUS.TXT') THEN make_header;

        ISPHITEST := TRUE ;     {phitest is activated}
        taufile:=whatfile2;
          monitorfile:=CONCAT(taufile,'.MO',sitenum);
          taufile:=CONCAT(taufile,'.DA',sitenum);
        ASSIGN(TAU_FILE,TAUFILE);
        REWRITE(TAU_FILE);
        CLOSE(TAU_FILE);
        WRITELN('Opacity data to disk file ', taufile);
        ASSIGN(MON_FILE,MONITORFILE);
        REWRITE(MON_FILE);
        CLOSE(MON_FILE);
        montime:= minutes;
        WRITELN('Monitor data to disk file ', monitorfile);
        sleep(1000);

gamma:  CONNECT_BOARD(newbrd);
        CLRSCR;
      { START;} SCREEN;
      { STEPBACK;}
        posit := true;
        DIRECT;
        GZEN;
```

```
        ZENTAU;

        If minutes >= montime + monint then begin
          montime := minutes;
          mon_check(mm,dmon);
        end;

        tautime := minutes;
        TIPPING:= TRUE;
        MEASURE(xx, yy, zz);
        SUBADJUST;

        gaintip:=Gfudge/gain;
        tautip:=tau;
        sigmatautip:=sigmatau;
        tauz2:=tauz;
        ambtemptip:=ambtemp;
        gain:=Gfudge/gain;
        FOR COUNT:=1 to 20 do
          begin
            Gcorr:=((coldtemp-ambtemp+Tfudge)+(ambtemp-Tfudge)*exp(-tauz2));
            Gcorr:=Gcorr/((coldtemp-ambtemp+Tfudge)+(ambtemp-TFudge)*exp(-tau));
            gain:=gain*Gcorr;
            tauz2:=tau;
            subadjust1;
          end;

        CLRSCR;
        WRITELN(date2,' ',time,'UT TAU = ',tautip:6:3,' +/- ',sigmatau:6:4);
        WRITELN(' tauz =  ',tauz:6:3,' tau cali = ',tau:6:3);
        make_line('Airmass: ',xx);
        make_line('ln(vsd): ',yy);
        make_line('rms(vsd):',zz);
        sleep(15000);
      if (exist('c:\procomm\lock.txt')) then sleep(120000);
        save(xx,yy,zz,tau,sigmatau,tauz,gain,ambtemp,coldtemp);

writeln;
{   Writeln('Another tau scan? (y/n)');
```

```
Readln(Q); Q := copy(Q,1,1);
If (Q = 'Y') or (Q = 'y') then goto beta;
goto psi;                                        }

beta: IF date2 <> startdate THEN GOTO alpha;

     IF minutes >= montime + monint THEN BEGIN
       montime:= minutes;
       mon_check(mm,dmon);
     END;
     IF minutes >= tautime + tauint THEN GOTO gamma;

     WINDOW(1,1,80,25) ;
     thismin:=minutes;
     GOTOXY(1,20);
     write('                                                                    ');
     GOTOXY(1,21);
     write('                                                                    ');
     gotoxy(1,20);
     WRITELN('Minutes = ',thismin,' Next Tau scan at ',tautime + tauint);
     IF NOT TIPPING THEN BEGIN
delta: IF minutes < ((thismin+1) MOD 1440) THEN GOTO delta;
       GOTO beta;
     END;    {IF NOT TIPPING}

psi:

REWRITE(GLOB1);
WRITE(GLOB1,i,n,p,j,thismin,dmon,adr,tautime);
WRITE(GLOB1,montime,airstep,ioctl_value,mon_flag);
CLOSE(GLOB1);

REWRITE(GLOB2);
WRITE(GLOB2,air,gain,zac,coldtemp,tau,slope,int);
WRITE(GLOB2,ambtemp,ix,iy,Tamb,sumx,sumy,sumw,w,sumxy,sumxsq);
WRITE(GLOB2,Tat,Tat1,gain2,adjust,adjust1,sigmatau,tauz,jy,jx,sigmasq);
FOR NEWI:= 1 TO 10 DO
  WRITE(GLOB2,xx[NEWI],yy[NEWI],zz[NEWI]);
FOR NEWI:= 1 TO 16 DO
```

```
  WRITE(GLOB2,mm[NEWI],raw_data[NEWI],proc_data[NEWI],tproc_data[NEWI]);
CLOSE(GLOB2);

REWRITE(GLOB3);
WRITE(GLOB3,flag,achec,posit,stop);
CLOSE(GLOB3);

REWRITE(GLOB4);
WRITE(GLOB4,b,address_hi,nadr,comvar,com_base);
CLOSE(GLOB4);

REWRITE(GLOB5);
WRITE(GLOB5,taufile,monitorfile,startdate);
CLOSE(GLOB5);

END.
```

D WHRSTRT.PAS

```
PROGRAM WHRSTRT;

{THIS PROGRAM IS RUN BY THE COMPUTER CONTROLLING THE RADIOMETER ONLY}
{WHEN THAT COMPUTER IS FIRST BOOTED UP.  IT STORES A VALUE OF FALSE}
{INTO A FILE OF BOOLEAN CALLED 'STORBOOL.DAT.'  THIS DATA FILE IS}
{READ BY CVTIPPER TO DETERMINE WHICH COMMANDS NEED TO BE EXECUTED TO}
{INSURE PROPER RADIOMETER OPERATION.  WHEN THE COMPUTER IS FIRST
BOOTED,}
{FOR INSTANCE, A 'STORBOOL.DAT' VALUE OF 'FALSE' INDICATES THAT}
{CVTIPPER .DAT AND .MON FILES MUST BE INITIALIZED.  WHILE BEING RUN}
{THE FIRST TIME, CVTIPPER CHANGES THE 'STORBOOL.DAT' VALUE TO 'TRUE.'}
{THE NEXT TIME CVTIPPER IS EXECUTED, IT KNOWS TO READ IN VALUES OF}
{VARIABLES THAT WERE SAVED TO DISK UPON EXIT OF THE PREVIOUS RUN.}

VAR
  DATAFILE: FILE OF BOOLEAN;
  NOTBOOTED:BOOLEAN;

BEGIN
```

```
  NOTBOOTED:=FALSE;
  ASSIGN(DATAFILE,'STORBOOL.DAT');
  REWRITE(DATAFILE);
  WRITE(DATAFILE,NOTBOOTED);
  CLOSE(DATAFILE);
END.
```

E Watchdog Documentation

```
WATCHDOG.COM and WATCHDG1.COM

Written by James R.  Reinders, minor modifications by Jim Kovalsky.  Contact
either through The Sailboard, Highland, MI (313) 887-7429.  PC-Slave mods
by Doug Azzarito, TECHNOLOGY CONSULTANTS RBBS, 407-627-6969.

Purpose:
--------

To monitor the carrier status on the desginated serial port, and re-boot the
machine if carrier is dropped.  This reset WILL include the power-on
self-test.

Designed specifically for use with Bulletin Board systems allowing remote
users to exit to DOS, but not capable of monitoring the carrier.

Prevents the unwanted situation of a caller dropping to DOS and accidentally
(or not!) disconnecting, leaving the system 'hung' until a manual re-boot can
be executed.

Implementation:
---------------

Use WATCHDOG.COM to protect COM2:  and WATCHDG1.COM to protect COM1: (The
original version was for COM2!)

For correct usage, the following must be true:

  1)  Your AUTOEXEC.BAT file must be set to start your Bulletin Board
```

2) WATCHDOG MUST be activated and deactivated as instructed below

Installation:

If your batch file to change the serial port into the console is RCTTY.BAT, and it returns control to RBBS.BAT when it has finished, it MUST be set as follows:

```
WATCHDOG ON
.
.
. Whatever commands you have
.
.
WATCHDOG OFF
RBBS
```

The first line installs and activates watchdog, and the second-from-bottom line deactivates watchdog. WATCHDOG MUST BE TURNED OFF OR IT WILL RESET THE SYSTEM WHEN THE CALLER HANGS UP, EVEN IF HE HAS RETURNED TO THE BULLETIN BOARD!

Command Line Options:

WATCHDOG ON
... Activates, will also install if not already resident

WATCHDOG OFF
... Deactivates and remains resident if previously installed,
installs and leaves inactive if not already loaded.

Notes:

Any utility that uses the clock interrupt will cause Watchdog to malfunction.

Watchdog checks the carrier status with each cycle of the clock, (18.2 times per second) and if any other clock interrupts are executed AFTER installing Watchdog, it can no longer make its interrupt!

Distribution:

This program is in the Public Domain, feel free to copy and distribute it. Please DO NOT distribute any modified versions or alter any credits. Any suggestions or improvements should be addressed/uploaded to James Reinders on The Sailboard, Highland, Mi - 313-887-7429 [300/1200, 24 hours]

PC-Slave version:

A special version of WATCHDOG (WATCHDGS.COM) is provided for use by PC-Slave card users. Because the PC-Slave does not use standard BIOS locations, the method used by WATCHDOG will not work. WATCHDGS will monitor COM2: as WATCHDOG does, but it uses INT 19H (BOOT-STRAP) to reboot the PC-Slave if carrier is lost. Use WATCHDGS on PC-Slaves and other non standard MS-DOS systems, and WATCHDOG on standard PC's and compatibles.

F Zip Documentation

NAME

zip, zipcloak, zipnote, zipsplit - package and compress (archive) files

SYNOPSIS

```
zip [-cdDeEfFghjklLmoqrSTuvVwyz@$] [-b path] [-n suffixes]
[-t mmddyy] [ zipfile [ file1 file2 ...]] [-xi list]

zipcloak [-dhL] [-b path] zipfile

zipnote [-hwL] [-b path] zipfile

zipsplit [-hiLpst] [-n size] [-b path] zipfile
```

DESCRIPTION

zip is a compression and file packaging utility for Unix, VMS, MSDOS, OS/2, Windows NT, Minix, Atari and Macintosh. It is analogous to a combination of the UNIX commands tar(1) and compress(1) and is compatible with PKZIP (Phil Katz's ZIP for MSDOS systems).

A companion program (unzip(1L)), unpacks zip archives. The zip and unzip(1L) programs can work with archives produced by PKZIP, and PKZIP and PKUNZIP can work with archives produced by zip. zip version 2.0.1 is compatible with PKZIP 2.04 Note that PKUNZIP 1.10 cannot extract files produced by PKZIP 2.04 or zip 2.0.1. You must use PKUNZIP 2.04g or unzip 5.0p1 (or later versions) to extract them.

For a brief help on zip and unzip, run each without specifying any parameters on the command line.

The program is useful for packaging a set of files for distribution; for archiving files; and for saving disk space by temporarily compressing unused files or directories.

The zip program puts one or more compressed files into a single zip archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a zip archive with a single command. Compression ratios of 2:1 to 3:1 are common for text files. zip has one compression method (deflation) and can also store files without compression. zip automatically chooses the better of the two for each file to be compressed.

When given the name of an existing zip archive, zip will replace identically named entries in the zip archive or add entries for new names. For example, if foo.zip exists and contains foo/file1 and foo/file2, and the directory foo contains the files foo/file1 and foo/file3, then:

```
zip -r foo foo
```

will replace foo/file1 in foo.zip and add foo/file3 to foo.zip. After this, foo.zip contains foo/file1, foo/file2, and foo/file3, with foo/file2 unchanged from before.

If the file list is specified as -@, zip takes the list of input files from standard input. Under UNIX, this option can be used to powerful effect in conjunction with the find(1) command. For example, to archive all the C source files in the current directory and its subdirectories:

```
find . -name ``*.[ch]'' -print | zip source -@
```

(note that the pattern must be quoted to keep the shell from expanding it). zip will also accept a single dash (``-'') as the zip file name, in which case it will write the zip file to standard output, allowing the output to be piped to another program. For example:

```
zip -r - . | dd of=/dev/nrst0 obs=16k
```

would write the zip output directly to a tape with the specified block size for the purpose of backing up the current directory.

zip also accepts a single dash (``-'') as the name of a file to be compressed, in which case it will read the file from standard input, allowing zip to take input from another program. For example:

```
tar cf - . | zip backup -
```

would compress the output of the tar command for the purpose of backing up the current directory. This generally produces better compression than the previous example using the -r option, because zip can take advantage of redundancy between files. The backup can be restored using the command

```
unzip -p backup | tar xf -
```

When no zip file name is given and stdout is not a terminal, zip acts as a filter, compressing standard input to standard output. For example,

```
tar cf - . | zip | dd of=/dev/nrst0 obs=16k
```

is equivalent to

```
tar cf - . | zip - - | dd of=/dev/nrst0 obs=16k
```

zip archives created in this manner can be extracted with the program funzip which is provided in the unzip package, or by gunzip which is provided in the gzip package. For example:

```
dd if=/dev/nrst0  ibs=16k | funzip | tar xvf -
```

When changing an existing zip archive, zip will write a temporary file with the new contents, and only replace the old one when the process of creating the new version has been completed without error.

If the name of the zip archive does not contain an extension, the extension .zip is added. If the name already contains an extension other than .zip the existing extension is kept unchanged.

OPTIONS

-b path

Use the specified path for the temporary zip archive. For example:

```
zip -b /tmp stuff *
```

will put the temporary zip archive in the directory /tmp, copying over stuff.zip to the current directory

when done. This option is only useful when updating an existing archive, and the file system containing this old archive does not have enough space to hold both old and new archive at the same time.

-c Add one-line comments for each file. File operations (adding, updating) are done first, and the user is then prompted for a one-line comment for each file. Enter the comment followed by return, or just return for no comment.

-d Remove (delete) entries from a zip archive. For example:

```
zip -d foo foo/tom/junk foo/harry/\* \*.o
```

will remove the entry foo/tom/junk, all of the files that start with foo/harry/, and all of the files that end with .o (in any path). Note that shell pathname expansion has been inhibited with backslashes, so that zip can see the asterisks, enabling zip to match on the contents of the zip archive instead of the contents of the current directory.

Under MSDOS, -d is case sensitive when it matches names in the zip archive. This requires that file names be entered in upper case if they were zipped by PKZIP on an MSDOS system.

-D Do not create entries in the zip archive for directories. Directory entries are created by default so that their attributes can be saved in the zip archive. The environment variable ZIPOPT can be used to change the default options. For example under Unix with sh:

```
ZIPOPT=''-D''; export ZIPOPT
```

(The variable ZIPOPT can be used for any option except -i and -x and can include several options.) The option

-D is a shorthand for -x ‘‘*/’’ but the latter cannot be set as default in the ZIPOPT environment variable.

-e Encrypt the contents of the zip archive using a password which is entered on the terminal in response to a prompt (this will not be echoed; if standard error is not a tty, zip will exit with an error).

-ee Encrypt contents, prompting for the password twice, checking that the two entries are identical before using the password.

-f Replace (freshen) an existing entry in the zip archive only if it has been modified more recently than the version already in the zip archive; unlike the update option (-u) this will not add files that are not already in the zip archive. For example:

```
zip -f foo
```

This command should be run from the same directory from which the original zip command was run, since paths stored in zip archives are always relative.

-F Fix the zip archive. This option can be used if some portions of the archive are missing. It is not guaranteed to work, so you MUST make a backup of the original archive first.

When doubled as in -FF the compressed sizes given inside the damaged archive are not trusted and zip scans for special signatures to identify the limits between the archive members. The single -F is more reliable if the archive is not too much damaged, for example if it has only been truncated, so try this option first.

Neither option will recover archives that have been incorrectly transferred in ascii mode instead of

binary. After the repair, the -t option of unzip may show that some files have a bad CRC. Such files cannot be recovered; you can remove them from the archive using the -d option of zip.

-g Grow (append to) the specified zip archive, instead of creating a new one. If this operation fails, zip attempts to restore the archive to its original state. If the restoration fails, the archive might become corrupted.

-h Display the zip help information (this also appears if zip is run with no arguments).

-i files
Include only the specified files, as in:

```
zip -r foo . -i \*.c
```

which will include only the files that end in .c in the current directory and its subdirectories. (Note for PKZIP users: the equivalent command is

```
pkzip -r foo *.c
```

PKZIP does not allow recursion in directories other than the current one.) The backslash avoids the shell filename substitution, so that the name matching is performed by zip at all directory levels.

-j Store just the name of a saved file (junk the path), and do not store directory names. By default, zip will store the full path (relative to the current path).

-k Attempt to convert the names and paths to conform to MSDOS, store only the MSDOS attribute (just the user write attribute from UNIX), and mark the entry as made under MSDOS (even though it was not); for compatibility with PKUNZIP under MSDOS which cannot handle certain

names such as those with two dots.

-l Translate the Unix end-of-line character LF into the MSDOS convention CR LF. This option should not be used on binary files. This option can be used on Unix if the zip file is intended for PKUNZIP under MSDOS. If the input files already contain CR LF, this option adds an extra CR. This ensure that unzip -a on Unix will get back an exact copy of the original file, to undo the effect of zip -l.

-ll Translate the MSDOS end-of-line CR LF into Unix LF. This option should not be used on binary files. This option can be used on MSDOS if the zip file is intended for unzip under Unix.

-L Display the zip license.

-m Move the specified files into the zip archive; actually, this deletes the target directories/files after making the specified zip archive. If a directory becomes empty after removal of the files, the directory is also removed. No deletions are done until zip has created the archive without error. This is useful for conserving disk space, but is potentially dangerous so it is recommended to use it in combination with -T to test the archive before removing all input files.

-n suffixes
Do not attempt to compress files named with the given suffixes. Such files are simply stored (0% compression) in the output zip file, so that zip doesn't waste its time trying to compress them. The suffixes are separated by either colons or semicolons. For example:

```
zip -rn .Z:.zip:.tiff:.gif:.snd foo foo
```

will copy everything from foo into foo.zip, but will store any files that end in .Z, .zip, .tiff, .gif, or

.snd without trying to compress them (image and sound files often have their own specialized compression methods). By default, zip does not compress files with extensions in the list .Z:.zip:.zoo:.arc:.lzh:.arj. Such files are stored directly in the output archive. The environment variable ZIPOPT can be used to change the default options. For example under Unix with csh:

setenv ZIPOPT ``-n .gif:.zip''

To attempt compression on all files, use:

zip -n : foo

The maximum compression option -9 also attempts compression on all files regardless of extension.

-o Set the ``last modified'' time of the zip archive to the latest (oldest) ``last modified'' time found among the entries in the zip archive. This can be used without any other operations, if desired. For example:

zip -o foo

will change the last modified time of foo.zip to the latest time of the entries in foo.zip.

-q Quiet mode; eliminate informational messages and comment prompts. (Useful, for example, in shell scripts and background tasks).

-r Travel the directory structure recursively; for example:

zip -r foo foo

In this case, all the files and directories in foo are saved in a zip archive named foo.zip, including files with names starting with ``.'', since the recursion does

not use the shell's file-name substitution mechanism. If you wish to include only a specific subset of the files in directory foo and its subdirectories, use the -i option to the specify the pattern of files to be included. You should not use -r with the name ``.*'', since that matches ``..'' which will attempt to zip up the parent directory (probably not what was intended).

-S Include system and hidden files. This option is effective on some systems only; it is ignored on Unix.

-t mmddyy
Do not operate on files modified prior to the specified date, where mm is the month (0-12), dd is the day of the month (1-31), and yy are the last two digits of the year. For example:

```
zip -rt 120791 infamy foo
```

will add all the files in foo and its subdirectories that were last modified on or after 7 December 1991, to the zip archive infamy.zip.

-T Test the integrity of the new zip file. If the check fails, the old zip file is unchanged and (with the -m option) not input files are removed.

-u Replace (update) an existing entry in the zip archive only if it has been modified more recently than the version already in the zip archive. For example:

```
zip -u stuff *
```

will add any new files in the current directory, and update any files which have been modified since the zip archive stuff.zip was last created/modified (note that zip will not try to pack stuff.zip into itself when you do this).

Note that the -u option with no arguments acts like the -f (freshen) option.

-v Verbose mode. Display a progress indicator during compression.

-V Save VMS file attributes. This option is available on VMS only; zip archives created with this option will generally not be usable on other systems.

-w Append the version number of the files to the name, including multiple versions of files. (VMS only; default: use only the most recent version of a specified file).

-x files
Explicitly exclude the specified files, as in:

```
zip -r foo foo -x \*.o
```

which will include the contents of foo in foo.zip while excluding all the files that end in .o. The backslash avoids the shell filename substitution, so that the name matching is performed by zip at all directory levels.

-y Store symbolic links as such in the zip archive, instead of compressing and storing the file referred to by the link (UNIX only).

-z Prompt for a multi-line comment for the entire zip archive. The comment is ended by a line containing just a period, or an end of file condition (^D on UNIX, ^Z on MSDOS, OS/2, and VAX/VMS). The comment can be taken from a file:

```
zip -z foo < foowhat
```

-# Regulate the speed of compression using the specified

digit #, where -0 indicates no compression (store all files), -1 indicates the fastest compression method (less compression) and -9 indicates the slowest compression method (optimal compression, ignores the suffix list). The default compression level is -6.

-@ Take the list of input files from standard input.

-$ Include the volume label for the the drive holding the first file to be compressed. If you want to include only the volume label or to force a specific drive, use the drive name as first file name, as in:

```
zip -$ foo a: c:bar
```

This option is effective on some systems only (MSDOS and OS/2); it is ignored on Unix.

EXAMPLES

The simplest example:

```
zip stuff *
```

creates the archive stuff.zip (assuming it does not exist) and puts all the files in the current directory in it, in compressed form (the .zip suffix is added automatically, unless that archive name given contains a dot already; this allows the explicit specification of other suffixes).

Because of the way the shell does filename substitution, files starting with ‘‘.’’ are not included; to include these as well:

```
zip stuff .* *
```

Even this will not include any subdirectories from the current directory.

To zip up an entire directory, the command:

```
zip -r foo foo
```

creates the archive foo.zip, containing all the files and directories in the directory foo that is contained within the current directory.

You may want to make a zip archive that contains the files in foo, without recording the directory name, foo. You can use the -j option to leave off the paths, as in:

```
zip -j foo foo/*
```

If you are short on disk space, you might not have enough room to hold both the original directory and the corresponding compressed zip archive. In this case, you can create the archive in steps using the -m option. If foo contains the subdirectories tom, dick, and harry, you can:

```
zip -rm foo foo/tom
zip -rm foo foo/dick
zip -rm foo foo/harry
```

where the first command creates foo.zip, and the next two add to it. At the completion of each zip command, the last created archive is deleted, making room for the next zip command to function.

PATTERN MATCHING

This section applies only to UNIX. Watch this space for details on MSDOS and VMS operation.

The UNIX shells (sh(1) and csh(1)) do filename substitution on command arguments. The special characters are:

? match any single character

* match any number of characters (including none)

[] match any character in the range indicated within the brackets (example: [a-f], [0-9]).

When these characters are encountered (without being escaped with a backslash or quotes), the shell will look for files relative to the current path that match the pattern, and replace the argument with a list of the names that matched.

The zip program can do the same matching on names that are in the zip archive being modified or, in the case of the -x (exclude) or -i (include) options, on the list of files to be operated on, by using backslashes or quotes to tell the shell not to do the name expansion. In general, when zip encounters a name in the list of files to do, it first looks for the name in the file system. If it finds it, it then adds it to the list of files to do. If it does not find it, it looks for the name in the zip archive being modified (if it exists), using the pattern matching characters described above, if present. For each match, it will add that name to the list of files to be processed, unless this name matches one given with the -x option, or does not match any name given with the -i option.

The pattern matching includes the path, and so patterns like *.o match names that end in ``.o'', no matter what the path prefix is. Note that the backslash must precede every special character (i.e. ?*[]), or the entire argument must be enclosed in double quotes (``'').

In general, use backslash to make zip do the pattern matching with the -f (freshen) and -d (delete) options, and sometimes after the -x (exclude) option when used with an appropriate operation (add, -u, -f, or -d).

SEE ALSO

compress(1), shar(1L), tar(1), unzip(1L), gzip(1L)

BUGS

zip 2.0.1 is not compatible with PKUNZIP 1.10. Use zip 1.1

to produce zip files which can be extracted by PKUNZIP 1.10.

zip files produced by zip 2.0.1 must not be updated by zip 1.1 or PKZIP 1.10, if they contain encrypted members or if they have been produced in a pipe or on a non-seekable device. The old versions of zip or PKZIP would create an archive with an incorrect format. The old versions can list the contents of the zip file but cannot extract it anyway (because of the new compression algorithm). If you do not use encryption and use regular disk files, you do not have to care about this problem.

Under VMS, not all of the odd file formats are treated properly. Only stream-LF format zip files are expected to work with zip. Others can be converted using Rahul Dhesi's BILF program. This version of zip handles some of the conversion internally. When using Kermit to transfer zip files from Vax to MSDOS, type ''set file type block'' on the Vax. When transfering from MSDOS to Vax, type ''set file type fixed'' on the Vax. In both cases, type ''set file type binary'' on MSDOS.

Under VMS, zip hangs for file specification that uses DECnet syntax foo::*.*.

On OS/2, zip cannot match some names, such as those including an exclamation mark or a hash sign. This is a bug in OS/2 itself: the 32-bit DosFindFirst/Next don't find such names. Other programs such as GNU tar are also affected by this bug.

Under OS/2, the amount of External Attributes displayed by DIR is (for compatibility) the amount returned by the 16-bit version of DosQueryPathInfo(). Otherwise OS/2 1.3 and 2.0 would report different EA sizes when DIRing a file. However, the structure layout returned by the 32-bit DosQueryPathInfo() is a bit different, it uses extra padding bytes and link pointers (it's a linked list) to have all fields on 4-byte boundaries for portability to future RISC

OS/2 versions. Therefore the value reported by zip (which uses this 32-bit-mode size) differs from that reported by DIR. zip stores the 32-bit format for portability, even the 16-bit MS-C-compiled version running on OS/2 1.3, so even this one shows the 32-bit-mode size.

AUTHORS

Please send bug reports and comments by email to: zip-bugs@wkuvx1.bitnet. For bug reports, please include the version of zip, the make options used to compile it, the machine and operating system in use, and as much additional information as possible.

ACKNOWLEDGEMENTS

Thanks to R. P. Byrne for his Shrink.Pas program, which inspired this project, and from which the shrink algorithm was stolen; to Phil Katz for placing in the public domain the zip file format, compression format, and .ZIP filename extension, and for accepting minor changes to the file format; to Steve Burg for clarifications on the deflate format; to Haruhiko Okumura and Leonid Broukhis for providing some useful ideas for the compression algorithm; to Keith Petersen, Rich Wales, Hunter Goatley and Mark Adler for providing a mailing list and ftp site for the INFO-ZIP group to use; and most importantly, to the INFO-ZIP group itself (listed in the file infozip.who) without whose tireless testing and

bug-fixing efforts a portable zip would not have been possible. Finally we should thank (blame) the first INFO-ZIP moderator, David Kirschbaum, for getting us into this mess in the first place. The manual page was rewritten for UNIX by R. P. C. Rodgers.

G Unzip Documentation

```
UNZIP(1L)          MISC. REFERENCE MANUAL PAGES          UNZIP(1L)

NAME
     unzip - list, test and extract compressed files in a ZIP
     archive

SYNOPSIS
     unzip [-Z] [-cflptuvz[abjnoqsCLV$]] file[.zip] [file(s) ...]
     [-x xfile(s) ...] [-d exdir]

DESCRIPTION
     unzip will list, test, or extract files from a ZIP archive,
     commonly found on MS-DOS systems. The default behavior
     (with no options) is to extract into the current directory
     (and subdirectories below it) all files from the specified
     ZIP archive. A companion program, zip(1L), creates ZIP
     archives; both programs are compatible with archives created
     by PKWARE's PKZIP and PKUNZIP for MS-DOS, but in many cases
     the program options or default behaviors differ.

ARGUMENTS
     file[.zip]
          Path of the ZIP archive(s). If the file specification
          is a wildcard, each matching file is processed in an
          order determined by the operating system (or file sys-
          tem). Only the filename can be a wildcard; the path
          itself cannot. Wildcard expressions are similar to
          Unix egrep(1) (regular) expressions and may contain:
```

```
        *    matches a sequence of 0 or more characters

        ?    matches exactly 1 character

        [...]
             matches any single character found inside the
             brackets; ranges are specified by a beginning
             character, a hyphen, and an ending character.  If
             an exclamation point or a caret ('!' or '^') fol-
             lows the left bracket, then the range of charac-
             ters within the brackets is complemented (that is,
             anything except the characters inside the brackets
             is considered a match).

        (Be sure to quote any character which might otherwise
        be interpreted or modified by the operating system,
        particularly under Unix and VMS.) If no matches are
        found, the specification is assumed to be a literal
        filename; and if that also fails, the suffix .zip is
        appended.  Note that self-extracting ZIP files are sup-
        ported, as with any other ZIP archive; just specify the
        .exe suffix (if any) explicitly.

    [file(s)]
        An optional list of archive members to be processed,

Info-ZIP         Last change: 28 Aug 94 (v5.12)                1

UNZIP(1L)         MISC. REFERENCE MANUAL PAGES          UNZIP(1L)

        separated by spaces.  (VMS versions compiled with
        VMSCLI defined must delimit files with commas instead.
        See -v in OPTIONS below.) Regular expressions (wild-
        cards) may be used to match multiple members; see
        above.  Again, be sure to quote expressions that would
        otherwise be expanded or modified by the operating sys-
        tem.

    [-x xfile(s)]
```

An optional list of archive members to be excluded from processing. Since wildcard characters match directory separators (‘/’), this option may be used to exclude any files which are in subdirectories. For example, ‘‘unzip foo *.[ch] -x */*’’ would extract all C source files in the main directory, but none in any subdirectories. Without the -x option, all C source files in all directories within the zipfile would be extracted.

[-d exdir]

An optional directory to which to extract files. By default, all files and subdirectories are recreated in the current directory; the -d option allows extraction in an arbitrary directory (always assuming one has permission to write to the directory). This option need not appear at the end of the command line; it is also accepted immediately after the zipfile specification, or between the file(s) and the -x option. The option and directory may be concatenated without any white space between them, but note that this may cause normal shell behavior to be suppressed. In particular, ‘‘-d ~’’ (tilde) is expanded by Unix C shells into the name of the user’s home directory, but ‘‘-d~’’ is treated as a literal subdirectory ‘‘~’’ of the current directory.

OPTIONS

Note that, in order to support obsolescent hardware, unzip’s usage screen is limited to 22 or 23 lines and should therefore be considered a reminder of the basic unzip syntax rather than an exhaustive list of all possible flags.

-Z zipinfo(1L) mode. If the first option on the command line is -Z, the remaining options are taken to be zipinfo(1L) options. See the appropriate manual page for a description of these options.

-c extract files to stdout/screen (‘‘CRT’’). This option is similar to the -p option except that the name of

each file is printed as it is extracted, the -a option is allowed, and ASCII-EBCDIC conversion is automatically performed if appropriate. This option is not listed in the unzip usage screen.

-f freshen existing files, i.e., extract only those files which already exist on disk and which are newer than the disk copies. By default unzip queries before overwriting, but the -o option may be used to suppress the queries. Note that under many operating systems, the TZ (timezone) environment variable must be set correctly in order for -f and -u to work properly (under Unix the variable is usually set automatically). The reasons for this are somewhat subtle but have to do with the differences between DOS-format file times (always local time) and Unix-format times (always in GMT) and the necessity to compare the two. A typical TZ value is ``PST8PDT'' (US Pacific time with automatic adjustment for Daylight Savings Time or ``summer time'').

-l list archive files (short format). The names, uncompressed file sizes and modification dates and times of the specified files are printed, along with totals for all files specified. In addition, the zipfile comment and individual file comments (if any) are displayed. If a file was archived from a single-case file system (for example, the old MS-DOS FAT file system) and the -L option was given, the filename is converted to lowercase and is prefixed with a caret (^).

-p extract files to pipe (stdout). Nothing but the file data is sent to stdout, and the files are always extracted in binary format, just as they are stored (no conversions).

-t test archive files. This option extracts each specified file in memory and compares the CRC (cyclic redundancy check, an enhanced checksum) of the expanded file with the original file's stored CRC value.

-u update existing files and create new ones if needed. This option performs the same function as the -f option, extracting (with query) files which are newer than those with the same name on disk, and in addition it extracts those files which do not already exist on disk. See -f above for information on setting the timezone properly.

-v be verbose or print diagnostic version info. This option has evolved and now behaves as both an option and a modifier. As an option it has two purposes: when a zipfile is specified with no other options, -v lists archive files verbosely, adding to the -l info the compression method, compressed size, compression ratio and 32-bit CRC. When no zipfile is specified (that is, the complete command is simply ‘‘unzip -v’’),

a diagnostic screen is printed. In addition to the normal header with release date and version, unzip lists the home Info-ZIP ftp site and where to find a list of other ftp and non-ftp sites; the target operating system for which it was compiled, as well as (possibly) the hardware on which it was compiled, the compiler and version used, and the compilation date; any special compilation options which might affect the program's operation (see also DECRYPTION below); and any options stored in environment variables which might do the same (see ENVIRONMENT OPTIONS below). As a modifier it works in conjunction with other options

(e.g., -t) to produce more verbose or debugging output; this is not yet fully implemented but will be in future releases.

-z display only the archive comment.

MODIFIERS

-a convert text files. Ordinarily all files are extracted exactly as they are stored (as ‘‘binary’’ files). The -a option causes files identified by zip as text files (those with the ‘t’ label in zipinfo listings, rather than ‘b’) to be automatically extracted as such, converting line endings, end-of-file characters and the character set itself as necessary. (For example, Unix files use line feeds (LFs) for end-of-line (EOL) and have no end-of-file (EOF) marker; Macintoshes use carriage returns (CRs) for EOLs; and most PC operating systems use CR+LF for EOLs and control-Z for EOF. In addition, IBM mainframes and the Michigan Terminal System use EBCDIC rather than the more common ASCII character set, and NT supports Unicode.) Note that zip’s identification of text files is by no means perfect; some ‘‘text’’ files may actually be binary and vice versa. unzip therefore prints ‘‘[text]’’ or ‘‘[binary]’’ as a visual check for each file it extracts when using the -a option. The -aa option forces all files to be extracted as text, regardless of the supposed file type.

-b treat all files as binary (no text conversions). This is a shortcut for ---a.

-C match filenames case-insensitively. unzip’s philosophy is ‘‘you get what you ask for’’ (this is also responsible for the -L/-U change; see the relevant options below). Because some filesystems are fully case-sensitive (notably those under the Unix operating system) and because both ZIP archives and unzip itself are portable across platforms, unzip’s default behavior is

to match both wildcard and literal filenames case-

sensitively. That is, specifying ‘‘makefile’’ on the command line will only match ‘‘makefile’’ in the archive, not ‘‘Makefile’’ or ‘‘MAKEFILE’’ (and similarly for wildcard specifications). Since this does not correspond to the behavior of many other operating/file systems (for example, OS/2 HPFS which preserves mixed case but is not sensitive to it), the -C option may be used to force all filename matches to be case-insensitive. In the example above, all three files would then match ‘‘makefile’’ (or ‘‘make*’’, or similar). The -C option affects files in both the normal file list and the excluded-file list (xlist).

-j junk paths. The archive’s directory structure is not recreated; all files are deposited in the extraction directory (by default, the current one).

-L convert to lowercase any filename originating on an uppercase-only operating system or filesystem. (This was unzip’s default behavior in releases prior to 5.11; the new default behavior is identical to the old behavior with the -U option, which is now obsolete and will be removed in a future release.) Depending on the archiver, files archived under single-case filesystems (VMS, old MS-DOS FAT, etc.) may be stored as all-uppercase names; this can be ugly or inconvenient when extracting to a case-preserving filesystem such as OS/2 HPFS or a case-sensitive one such as under Unix. By default unzip lists and extracts such filenames exactly as they’re stored (excepting truncation, conversion of unsupported characters, etc.); this option causes the names of all files from certain systems to be converted to lowercase.

-n never overwrite existing files. If a file already exists, skip the extraction of that file without prompting. By default unzip queries before extracting any file which already exists; the user may choose to overwrite only the current file, overwrite all files, skip extraction of the current file, skip extraction of all existing files, or rename the current file.

-o overwrite existing files without prompting. This is a dangerous option, so use it with care. (It is often used with -f, however.)

-q perform operations quietly (-qq = even quieter). Ordinarily unzip prints the names of the files it's extracting or testing, the extraction methods, any file or zipfile comments which may be stored in the archive, and possibly a summary when finished with each archive. The -q[q] options suppress the printing of some or all

of these messages.

-s [OS/2, NT, MS-DOS] convert spaces in filenames to underscores. Since all PC operating systems allow spaces in filenames, unzip by default extracts filenames with spaces intact (e.g., ''EA DATA. SF''). This can be awkward, however, since MS-DOS in particular does not gracefully support spaces in filenames. Conversion of spaces to underscores can eliminate the awkwardness in some cases.

-U (obsolete; to be removed in a future release) leave filenames uppercase if created under MS-DOS, VMS, etc. See -L above.

-V retain (VMS) file version numbers. VMS files can be stored with a version number, in the format file.ext;##. By default the ‘‘;##’’ version numbers are stripped, but this option allows them to be retained. (On filesystems which limit filenames to particularly short lengths, the version numbers may be truncated or stripped regardless of this option.)

-X [VMS] restore owner/protection info (may require system privileges). Ordinary file attributes are always restored, but this option allows UICs to be restored as well. [The next version of unzip will support Unix UID/GID info as well, and possibly NT permissions.]

-$ [MS-DOS, OS/2, NT, Amiga] restore the volume label if the extraction medium is removable (e.g., a diskette). Doubling the option (-$$) allows fixed media (hard disks) to be labelled as well. By default, volume labels are ignored.

ENVIRONMENT OPTIONS

unzip's default behavior may be modified via options placed in an environment variable. This can be done with any option, but it is probably most useful with the -a, -L, -C, -q, -o, or -n modifiers: make unzip auto-convert text files by default, make it convert filenames from uppercase systems to lowercase, make it match names case-insensitively, make it quieter, or make it always overwrite or never overwrite files as it extracts them. For example, to make unzip act as quietly as possible, only reporting errors, one would use one of the following commands:

```
UNZIP=-qq; export UNZIP    Unix Bourne shell
setenv UNZIP -qq           Unix C shell
set UNZIP=-qq              OS/2 or MS-DOS
define UNZIP_OPTS "-qq"    VMS (quotes for lowercase)
```

Environment options are, in effect, considered to be just like any other command-line options, except that they are effectively the first options on the command line. To override an environment option, one may use the ‘‘minus operator’’ to remove it. For instance, to override one of the quiet-flags in the example above, use the command

```
unzip --q[other options] zipfile
```

The first hyphen is the normal switch character, and the second is a minus sign, acting on the q option. Thus the effect here is to cancel one quantum of quietness. To cancel both quiet flags, two (or more) minuses may be used:

```
unzip -t--q zipfile
unzip ---qt zipfile
```

(the two are equivalent). This may seem awkward or confusing, but it is reasonably intuitive: just ignore the first hyphen and go from there. It is also consistent with the behavior of Unix nice(1).

As suggested by the examples above, the default variable names are UNZIP_OPTS for VMS (where the symbol used to install unzip as a foreign command would otherwise be confused with the environment variable), and UNZIP for all other operating systems. For compatibility with zip(1L), UNZIPOPT is also accepted (don’t ask). If both UNZIP and UNZIPOPT are defined, however, UNZIP takes precedence. unzip’s diagnostic option (-v with no zipfile name) can be used to check the values of all four possible unzip and zipinfo environment variables.

The timezone variable (TZ) should be set according to the local timezone in order for the -f and -u to operate correctly. See the description of -f above for details. This variable may also be necessary in order for timestamps

on extracted files to be set correctly.

DECRYPTION

Encrypted archives are fully supported by Info-ZIP software, but due to United States export restrictions, the encryption and decryption sources are not packaged with the regular unzip and zip distributions. Since the crypt sources were written by Europeans, however, they are freely available at sites throughout the world; see the file ‘‘Where’’ in any Info-ZIP source or binary distribution for locations both inside and outside the US.

Because of the separate distribution, not all compiled versions of unzip support decryption. To check a version for crypt support, either attempt to test or extract an

encrypted archive, or else check unzip’s diagnostic screen (see the -v option above) for ‘‘[decryption]’’ as one of the special compilation options.

There are no runtime options for decryption; if a zipfile member is encrypted, unzip will prompt for the password without echoing what is typed. unzip continues to use the same password as long as it appears to be valid; it does this by testing a 12-byte header. The correct password will always check out against the header, but there is a 1-in-256 chance that an incorrect password will as well. (This is a security feature of the PKWARE zipfile format; it helps prevent brute-force attacks which might otherwise gain a large speed advantage by testing only the header.) In the case that an incorrect password is given but it passes the header test anyway, either an incorrect CRC will be generated for the extracted data or else unzip will fail during the extraction because the ‘‘decrypted’’ bytes do not constitute a valid compressed data stream.

If the first password fails the header check on some file, unzip will prompt for another password, and so on until all files are extracted. If a password is not known, entering a null password (that is, just a carriage return) is taken as a signal to skip all further prompting. Only unencrypted files in the archive(s) will thereafter be extracted. (Actually that's not quite true; older versions of zip(1L) and zipcloak(1L) allowed null passwords, so unzip checks each encrypted file to see if the null password works. This may result in ``false positives'' and extraction errors, as noted above.)

Note that there is presently no way to avoid interactive decryption. This is another security feature: plaintext passwords given on the command line or stored in files constitute a risk because they may be seen by others. Future releases may (under protest, with great disapproval) support such shenanigans.

EXAMPLES

To use unzip to extract all members of the archive letters.zip into the current directory and subdirectories below it, creating any subdirectories as necessary:

```
unzip letters
```

To extract all members of letters.zip into the current directory only:

```
unzip -j letters
```


To test letters.zip, printing only a summary message indicating whether the archive is OK or not:

```
unzip -tq letters
```

To test all zipfiles in the current directory, printing only the summaries:

```
unzip -tq \*.zip
```

(The backslash before the asterisk is only required if the shell expands wildcards, as in Unix; double quotes could have been used instead, as in the source examples below.) To extract to standard output all members of letters.zip whose names end in .tex, auto-converting to the local end-of-line convention and piping the output into more(1):

```
unzip -ca letters \*.tex | more
```

To extract the binary file paper1.dvi to standard output and pipe it to a printing program:

```
unzip -p articles paper1.dvi | dvips
```

To extract all FORTRAN and C source files--*.f, *.c, *.h, and Makefile--into the /tmp directory:

```
unzip source.zip "*.[fch]" Makefile -d /tmp
```

(the double quotes are necessary only in Unix and only if globbing is turned on). To extract all FORTRAN and C source files, regardless of case (e.g., both *.c and *.C, and any makefile, Makefile, MAKEFILE or similar):

```
unzip -C source.zip "*.[fch]" makefile -d /tmp
```

To extract any such files but convert any uppercase MS-DOS or VMS names to lowercase and convert the line-endings of all of the files to the local standard (without respect to any files which might be marked ‘‘binary’’):

```
    unzip -aaCL source.zip "*.[fch]" makefile -d /tmp
```

To extract only newer versions of the files already in the current directory, without querying (NOTE: be careful of unzipping in one timezone a zipfile created in another--ZIP archives to date contain no timezone information, and a ‘‘newer’’ file from an eastern timezone may, in fact, be older):


```
    unzip -fo sources
```

To extract newer versions of the files already in the current directory and to create any files not already there (same caveat as previous example):

```
    unzip -uo sources
```

To display a diagnostic screen showing which unzip and zipinfo options are stored in environment variables, whether decryption support was compiled in, the compiler with which unzip was compiled, etc.:

```
    unzip -v
```

In the last five examples, assume that UNZIP or UNZIP_OPTS is set to -q. To do a singly quiet listing:

```
    unzip -l file.zip
```

To do a doubly quiet listing:

```
    unzip -ql file.zip
```

(Note that the ‘‘.zip’’ is generally not necessary.) To do a standard listing:

```
        unzip --ql file.zip
    or
        unzip -l-q file.zip
    or
        unzip -l--q file.zip       (extra minuses don't hurt)
```

TIPS

The current maintainer, being a lazy sort, finds it very useful to define a pair of aliases: tt for ``unzip -tq'' and ii for ``unzip -Z'' (or ``zipinfo''). One may then simply type ``tt zipfile'' to test an archive, something which is worth making a habit of doing. With luck unzip will report ``No errors detected in zipfile.zip,'' after which one may breathe a sigh of relief.

The maintainer also finds it useful to set the UNZIP environment variable to ``-aL'' and is tempted to add ``-C'' as well. His ZIPINFO variable is set to ``-z''.

DIAGNOSTICS

The exit status (or error level) approximates the exit codes defined by PKWARE and takes on the following values, except under VMS:

0 normal; no errors or warnings detected.

1 one or more warning errors were encountered, but processing completed successfully anyway. This includes zipfiles where one or more files was skipped due to unsupported compression method or encryption with an unknown password.

2 a generic error in the zipfile format was detected. Processing may have completed success-

fully anyway; some broken zipfiles created by other archivers have simple work-arounds.

3 a severe error in the zipfile format was detected. Processing probably failed immediately.

4-8 unzip was unable to allocate memory for one or more buffers.

9 the specified zipfiles were not found.

10 invalid options were specified on the command line.

11 no matching files were found.

50 the disk is (or was) full during extraction.

51 the end of the ZIP archive was encountered prematurely.

VMS interprets standard Unix (or PC) return values as other, scarier-looking things, so by default unzip always returns 0 (which reportedly gets converted into a VMS status of 1--i.e., success). There are two compilation options available to modify or expand upon this behavior: defining RETURN_CODES results in a human-readable explanation of what the real error status was (but still with a faked ‘‘success’’ exit value), while defining RETURN_SEVERITY causes unzip to exit with a ‘‘real’’ VMS status. The latter behavior will become the default in future versions unless it is found to conflict with officially defined VMS codes. The current mapping is as follows: 1 (success) for normal exit, 0x7fff0001 for warning errors, and (0x7fff000? + 16*normal_unzip_exit_status) for all other errors, where the ‘?’ is 2 (error) for unzip values 2 and 9-11, and 4 (fatal error) for the remaining ones (3-8, 50, 51). Check the ‘‘unzip -v’’ output to see whether RETURN_SEVERITY was defined at compilation time.

BUGS

When attempting to extract a corrupted archive, unzip may go into an infinite loop and, if not stopped quickly enough, fill all available disk space. Compiling with CHECK_EOF

should fix this problem for all zipfiles, but the option was introduced too late in the testing process to be made the default behavior. Future versions will be robust enough to fail gracefully on damaged archives. Check the ‘‘unzip -v’’ output to see whether CHECK_EOF was defined during compilation.

[MS-DOS] When extracting or testing files from an archive on a defective floppy diskette, if the ‘‘Fail’’ option is chosen from DOS’s ‘‘Abort, Retry, Fail?’’ message, unzip may hang the system, requiring a reboot. Instead, press control-C (or control-Break) to terminate unzip.

Under DEC Ultrix, unzip will sometimes fail on long zipfiles (bad CRC, not always reproducible). This is apparently due either to a hardware bug (cache memory) or an operating system bug (improper handling of page faults?).

Dates and times of stored directories are not restored.

[OS/2] Extended attributes for existing directories are never updated. This is a limitation of the operating system; unzip has no way to determine whether the stored attributes are newer or older than the existing ones.

[VMS] When extracting to another directory, only the [.foo] syntax is accepted for the -d option; the simple Unix foo syntax is silently ignored (as is the less common VMS foo.dir syntax).

[VMS] When the file being extracted already exists, unzip's query only allows skipping, overwriting or renaming; there should additionally be a choice for creating a new version of the file. In fact, the ''overwrite'' choice does create a new version; the old version is not overwritten or deleted.

SEE ALSO

funzip(1L), zip(1L), zipcloak(1L), zipgrep(1L), zipinfo(1L), zipnote(1L), zipsplit(1L)

AUTHORS

The primary Info-ZIP authors (current zip-bugs workgroup) are: Jean-loup Gailly (Zip); Greg R. Roelofs (UnZip); Mark Adler (decompression, fUnZip); Kai Uwe Rommel (OS/2); Igor Mandrichenko and Hunter Goatley (VMS); John Bush and Paul Kienitz (Amiga); Antoine Verheijen (Macintosh); Chris Herborth (Atari); Henry Gessau (NT); Karl Davis, Sergio Monesi and Evan Shattock (Acorn Archimedes); and Robert Heath (Windows). The author of the original unzip code upon which Info-ZIP's is based was Samuel H. Smith; Carl Mascott did the first Unix port; and David P. Kirschbaum organized and

led Info-ZIP in its early days. The full list of contributors to UnZip has grown quite large; please refer to the CONTRIBS file in the UnZip source distribution for a relatively complete version.

VERSIONS

v1.2	15 Mar 89	Samuel H. Smith
v2.0	9 Sep 89	Samuel H. Smith
v2.x	fall 1989	many Usenet contributors
v3.0	1 May 90	Info-ZIP (DPK, consolidator)
v3.1	15 Aug 90	Info-ZIP (DPK, consolidator)

```
     v4.0    1 Dec 90   Info-ZIP (GRR, maintainer)
     v4.1   12 May 91   Info-ZIP
     v4.2   20 Mar 92   Info-ZIP (zip-bugs subgroup, GRR)
     v5.0   21 Aug 92   Info-ZIP (zip-bugs subgroup, GRR)
     v5.01  15 Jan 93   Info-ZIP (zip-bugs subgroup, GRR)
     v5.1    7 Feb 94   Info-ZIP (zip-bugs subgroup, GRR)
     v5.11   2 Aug 94   Info-ZIP (zip-bugs subgroup, GRR)
     v5.12  28 Aug 94   Info-ZIP (zip-bugs subgroup, GRR)

Info-ZIP         Last change: 28 Aug 94 (v5.12)               13
```

H dat2text.c

```
#include <stdio.h>

struct rec
{
  char time1[4];   /*UT in HHMM format*/
  float tau1;      /*tipping scan opacity*/
  float sigmatau1; /*tipping scan rms*/
  float tauz1;     /*Zenith Opacity*/
  float Vz1;       /*Zenith mean voltage*/
  float GainZ1;    /*Zenith gain correction*/
  float sigmaVz1;  /*Zenith rms voltage measurement*/
  float gain1;     /*assumed gain*/
  float tauI1;     /*iterated opacity*/
  float sigtauI1;  /*iterated opacity rms*/
  float gainI1;    /*iterated gain*/
  float Tamb1;     /*Ambient Temperature*/
  float Tc;        /*Cold load temperature*/
  float Th;        /*Hot load temperature*/
  float x1[11],y1[11],z1[11],G1[11]; /*airmass, ln(vsd), rms(vsd)*/
};

float swapbytes();
void usage();

main(argc,argv)
```

```
int argc;
char *argv[];
{
  struct rec r;
  FILE * fp;
  char padding;
  float time;
  float lasttime = 0;
  char num[3];
  int i;

  if(argc!=2) usage();
  num[2] = '\0';
  if((fp = fopen(argv[1],''rb''))==NULL)
    {
      fprintf(stderr,''File not found: %s\n'',argv[1]);
      exit(1);
    }
  while((fread(&padding,sizeof(char),1,fp)==1) &&
        (fread(&r,sizeof(struct rec),1,fp))==1)
    {
      num[0] = r.time1[0];
      num[1] = r.time1[1];
      time = (float) atoi(num);
      num[0] = r.time1[2];
      num[1] = r.time1[3];
      time += ((float) atoi(num))/60;
      if((time>22)&&(time<lasttime)) time = 24;
      printf(``%7.3f %8.4f %8.4f %8.4f %8.4f %8.4f  %7.3f  %7.3f
       %%7.3f\n'',
             time,swapbytes(&r.tau1),swapbytes(&r.sigmatau1),
             swapbytes(&r.tauz1),
             swapbytes(&r.tauI1),swapbytes(&r.sigtauI1),swapbytes(&r.gain1),
             swapbytes(&r.GainZ1),swapbytes(&r.gainI1));
      lasttime = time;
    }
  fclose(fp);
}
```

```
void usage()
{
  fprintf(stderr,''Usage:\n  dat2text filename\n'');
  exit(1);
}

float swapbytes(x)
float *x;
{
  char *ptr;
  char c;

  ptr = (char *) x;
  c = ptr[0];
  ptr[0] = ptr[3];
  ptr[3] = c;
  c = ptr[1];
  ptr[1] = ptr[2];
  ptr[2] = c;
  return(*x);
}
```

I mon2text.c

```
#include <stdio.h>

struct rec
{
  char time1[4];  /*UT in HHMM format*/

  float m1[16];   /*Analog monitor data*/

};

float swapbytes();
void usage();

main(argc,argv)
```

```
int argc;
char *argv[];
{
  struct rec r;
  FILE * fp;
  char padding;
  short dmon2;
  float time;
  float lasttime = 0;
  char num[3];
  int i;

  if(argc!=2) usage();
  num[2] = '\0';
  if((fp = fopen(argv[1],''rb''))==NULL)
    {
      fprintf(stderr,''File not found: %s\n'',argv[1]);
      exit(1);
    }
  while((fread(&padding,sizeof(char),1,fp)==1) &&
        (fread(&r,sizeof(struct rec),1,fp)==1) &&
        (fread(&dmon2,sizeof(short),1,fp)==1))
    {
      num[0] = r.time1[0];
      num[1] = r.time1[1];
      time = (float) atoi(num);
      num[0] = r.time1[2];
      num[1] = r.time1[3];
      time += ((float) atoi(num))/60;
      if((time>22)&&(time<lasttime)) time = 24;
      printf(``%7.3f  ``,time);
      for(i=0;i<16;i++)
        printf(``%7.3f ``,swapbytes(&r.m1[i]));
      printf(``\n'');
      lasttime = time;
    }
  fclose(fp);
}
```

```
void usage()
{
  fprintf(stderr,''Usage:\n  mon2text filename\n'');
  exit(1);
}

float swapbytes(x)
float *x;
{
  char *ptr;
  char c;

  ptr = (char *) x;
  c = ptr[0];
  ptr[0] = ptr[3];
  ptr[3] = c;
  c = ptr[1];
  ptr[1] = ptr[2];
  ptr[2] = c;
  return(*x);
}
```

J Cplots.csh

```
#!/bin/csh

echo -n ``Chile Automatic Data Plotting: Started ``
date

set tmp = tmp$$

cd $work/tipper/Chile

set dates
set dates = `awk '{printf(``%s ``,$1)}' newdat.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  set flen = `wc -l archive/$yymm/$d.dat | awk '{print $1}'`
```

```
  if ( $flen > 10 ) then
    tauplot.csh $d $tmp.1/gif
    giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
    giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
    giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
    giftrans -t '#ffffff' $tmp.4 > $d.tau.C.gif
    rm $tmp.*
    mv $d.tau.C.gif /home/dietcoke/sfoster/public_html/tipper
  endif
end
rm newdat.log

set dates
set dates = `awk '{printf(``%s ``,$1)}' newmon.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  set flen = `wc -l archive/$yymm/$d.mon | awk '{print $1}'`
  if ( $flen > 10 ) then
    monplot.csh $d $tmp.1/vgif $tmp.5/vgif
    giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
    giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
    giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
    giftrans -t '#ffffff' $tmp.4 > $d.mona.C.gif
    giftrans -g '#ffffff=#ff0000' $tmp.5 > $tmp.6
    giftrans -g '#000000=#ffffff' $tmp.6 > $tmp.7
    giftrans -g '#ff0000=#000000' $tmp.7 > $tmp.8
    giftrans -t '#ffffff' $tmp.8 > $d.monb.C.gif
    rm $tmp.*
    mv $d.mona.C.gif /home/dietcoke/sfoster/public_html/tipper
    mv $d.monb.C.gif /home/dietcoke/sfoster/public_html/tipper
  endif
end
rm newmon.log

set dates
set dates = `awk '{printf(``%s ``,$1)}' newtxt.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  txtplot.csh $d $tmp.1/gif
```

```
  giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
  giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
  giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
  giftrans -t '#ffffff' $tmp.4 > $d.txt.C.gif
  rm $tmp.*
  mv $d.txt.C.gif /home/dietcoke/sfoster/public_html/tipper
end
rm newtxt.log

rm newout.log

set dates
set dates = `awk '{printf(``%s ``,$1)}' newwnd.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  wndplot.csh $d $tmp.1/gif
  giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
  giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
  giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
  giftrans -t '#ffffff' $tmp.4 > $d.wnd.C.gif
  rm $tmp.*
  mv $d.wnd.C.gif /home/dietcoke/sfoster/public_html/tipper
end

set dates
set dates = `awk '{printf(``%s ``,$1)}' newwnd.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  wdrplot.csh $d $tmp.1/gif
  giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
  giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
  giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
  giftrans -t '#ffffff' $tmp.4 > $d.wdr.C.gif
  rm $tmp.*
  mv $d.wdr.C.gif /home/dietcoke/sfoster/public_html/tipper
end
rm newwnd.log

echo -n ``Chile Automatic Data Plotting: Finished ``
```

```
date
```

K MKdata.csh

```
#!/bin/csh

echo -n ``Mauna Kea Automatic Data Retrieval: Started ``
date

umask 002
set tmp = tmp$$

cd $work/tipper/MaunaKea
ftp tipper.vlba.nrao.edu >& /dev/null
if (!(-e pickup.zip)) then
  echo Error: ftp did not recover a data file from the tipper.
  ping tipper.vlba.nrao.edu
  goto endscript
endif
if (-e pickup.zip) then
  unzip -a -L -o pickup.zip >& /dev/null
  rm pickup.zip
  rm glob*
  set phi = `ls | grep -i ``.phi'' | wc -l`
  if ( $phi != 0 ) rm *.phi
endif

if ( -e newmon.log ) mv newmon.log $tmp.newmon
if ( -e newdat.log ) mv newdat.log $tmp.newdat
if ( -e newtxt.log ) mv newtxt.log $tmp.newtxt
if ( -e newout.log ) mv newout.log $tmp.newout

set mon = `ls | grep ``.mom'' | wc -l`
if ($mon != 0) then
  foreach f (*.mom)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newmon
```

```
    if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
    mv $f rawarchive/$yymm
  end
endif

set dat = `ls | grep ``.dam'' | wc -l`
if ($dat != 0) then
  foreach f (*.dam)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    echo $date >> $tmp.newdat
    mv $f rawdata
  end
endif

set text = `ls | grep ``.txm'' | wc -l`
if ($text != 0) then
  foreach f (*.txm)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set base = `echo $f | awk '{printf(``%s\n'',substr($1,1,8))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newtxt
    if(! -e archive/$yymm) mkdir archive/$yymm
    cp $base.txm archive/$yymm/$base.txt
    if(! -e rawarchive/$yymm) mkdir rawarchive/$yymm
    mv $f rawarchive/$yymm
  end
endif

set out = `ls | grep ``.oum'' | wc -l`
if ($out != 0) then
  foreach f (*.oum)
    set date = `echo $f | awk '{printf(``%s\n'',substr($1,1,6))}'`
    set base = `echo $f | awk '{printf(``%s\n'',substr($1,1,8))}'`
    set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`
    echo $date >> $tmp.newout
    if(! -e archive/$yymm) mkdir archive/$yymm
    cp $f archive/$yymm/$base.out
  end
  mv *.oum rawdata
```

```
endif

awk -f plotlog.awk $tmp.newdat > newdat.log
awk -f plotlog.awk $tmp.newmon > newmon.log
awk -f plotlog.awk $tmp.newtxt > newtxt.log
awk -f plotlog.awk $tmp.newout > newout.log

if (-e newmon.log) then
  set dates = `awk '{printf(``%s ``,$1)}' newmon.log`
  foreach d ($dates)
    set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
    set mon = `ls rawarchive/$yymm | grep $d | grep ``.mom'' | wc -l`
    if ($mon != 0) then
      foreach f (rawarchive/$yymm/$d*.mom)
        mon2text $f >> $d.mon
      end
    endif
    if(! -e archive/$yymm) mkdir archive/$yymm
    mv $d.mon archive/$yymm
  end
endif

if (-e newdat.log) then
  set dates = `awk '{printf(``%s ``,$1)}' newdat.log`
  foreach d ($dates)
    set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
    set dat = `ls rawarchive/$yymm | grep $d | grep ``.dam'' | wc -l`
    if ($dat != 0) then
      foreach f (rawarchive/$yymm/$d*.dam)
        dat2text $f >> $d.dat
      end
    endif
    set dat = `ls rawdata | grep $d | grep ``.dam'' | wc -l`
    if ($dat != 0) then
      foreach f (rawdata/$d*.dam)
        dat2text $f >> $d.dat
      end
    endif
    if(! -e archive/$yymm) mkdir archive/$yymm
```

```
    mv $d.dat archive/$yymm
  end
endif

#
# Retrieve MK Weather Data from Socorro (see .netrc)
#
ftp -i ftp.aoc.nrao.edu >& /dev/null
foreach f (*.wea)
  set yymm = `echo $f | awk '{printf(``%s'',substr($1,1,4))}'`
  mv $f archive/$yymm
end

endscript:
rm $tmp.*

echo -n ``Mauna Kea Automatic Data Retrieval: Finished ``
date
```

L MKwdata.csh

```
#!/bin/csh

echo -n ``MK Automatic Weather Data Retrieval: Started ``
date

umask 002

cd $work/tipper/MaunaKea
#
# Retrieve MK Weather Data from Socorro (see .netrc)
#
ftp -i ftp.aoc.nrao.edu >& /dev/null
foreach f (*.wea)
  set yymm = `echo $f | awk '{printf(``%s'',substr($1,1,4))}'`
  if(! -e archive/$yymm) mkdir archive/$yymm
  mv $f archive/$yymm
end
```

```
endscript:

echo -n ``MK Automatic Weather Data Retrieval: Finished ``
date
```

M MKplots.csh

```
#!/bin/csh

echo -n ``Mauna Kea Automatic Data Plotting: Started ``
date

set tmp = tmp$$

cd $work/tipper/MaunaKea

umask 002
set dates
set dates = `awk '{printf(``%s ``,$1)}' newdat.log`
foreach d ($dates)
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  set flen = `wc -l archive/$yymm/$d.dat | awk '{print $1}'`
  if ( $flen > 10 ) then
    tauplot.csh $d $tmp.1/gif
    giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
    giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
    giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
    giftrans -t '#ffffff' $tmp.4 > $d.tau.MK.gif
    mv $d.tau.MK.gif /home/dietcoke/sfoster/public_html/tipper
    rm $tmp.*
  endif
end
rm newdat.log

set dates
set dates = `awk '{printf(``%s ``,$1)}' newmon.log`
foreach d ($dates)
```

```
  set yymm = `echo $d | awk '{printf(``%s'',substr($1,1,4))}'`
  set flen = `wc -l archive/$yymm/$d.mon | awk '{print $1}'`
  if ( $flen > 10 ) then
    monplot.csh $d $tmp.1/vgif $tmp.5/vgif
    giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
    giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
    giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
    giftrans -t '#ffffff' $tmp.4 > $d.mona.MK.gif
    giftrans -g '#ffffff=#ff0000' $tmp.5 > $tmp.6
    giftrans -g '#000000=#ffffff' $tmp.6 > $tmp.7
    giftrans -g '#ff0000=#000000' $tmp.7 > $tmp.8
    giftrans -t '#ffffff' $tmp.8 > $d.monb.MK.gif
    rm $tmp.*
    mv $d.mona.MK.gif /home/dietcoke/sfoster/public_html/tipper
    mv $d.monb.MK.gif /home/dietcoke/sfoster/public_html/tipper
  endif
end
rm newmon.log

set dates
set dates = `awk '{printf(``%s ``,$1)}' newtxt.log`
foreach d ($dates)
  txtplot.csh $d $tmp.1/gif
  giftrans -g '#ffffff=#ff0000' $tmp.1 > $tmp.2
  giftrans -g '#000000=#ffffff' $tmp.2 > $tmp.3
  giftrans -g '#ff0000=#000000' $tmp.3 > $tmp.4
  giftrans -t '#ffffff' $tmp.4 > $d.txt.MK.gif
  rm $tmp.*
  mv $d.txt.MK.gif /home/dietcoke/sfoster/public_html/tipper
end
rm newtxt.log

rm newout.log

echo -n ``Mauna Kea Automatic Data Plotting: Finished ``
date
```

N monitor.csh

```
#!/bin/csh

set tmp = tmp$$

cd $work/public_html/tipper

#
# Clean Up Chile Data
#
foreach f (`ls *.tau.C.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.mona.C.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.monb.C.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.txt.C.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.wnd.C.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.wdr.C.gif | tail -r | tail +8`)
  rm $f
end
#
# Clean Up Mauna Kea Data
#
foreach f (`ls *.tau.MK.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.mona.MK.gif | tail -r | tail +8`)
  rm $f
end
foreach f (`ls *.monb.MK.gif | tail -r | tail +8`)
```

```
  rm $f
end
foreach f (`ls *.txt.MK.gif | tail -r | tail +8`)
  rm $f
end
#
# Chile Tau Plots
#
foreach f (*.tau.C.gif)
  set date = `echo $f | awk '{printf(``%s'',substr($1,1,6))}'`
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Tau Plot</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
  echo $date >> $tmp.1
end
#
# Chile Mon Plots
#
foreach f (*.mona.C.gif)
  set date = `echo $f | awk '{printf(``%s'',substr($1,1,6))}'`
  cat << EOF > $date.mon.C.html
<HEAD>
<TITLE>Chile Monitor Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$date.monb.C.gif''>
<P ALIGN=CENTER>
<ADDRESS>
```

```
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
#
# Chile Txt Plots
#
foreach f (*.txt.C.gif)
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Text Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
#
# Chile Wnd Plots
#
foreach f (*.wnd.C.gif)
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Wind Speed Data Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
```

```
#
# Chile Wdr Plots
#
foreach f (*.wdr.C.gif)
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Wind Direction Data Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
#
# Mauna Kea Tau Plots
#
foreach f (*.tau.MK.gif)
  set date = `echo $f | awk '{printf(``%s'',substr($1,1,6))}'`
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Tau Plot</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
  echo $date >> $tmp.2
end
#
# Mauna Kea Mon Plots
#
```

```
foreach f (*.mona.MK.gif)
  set date = `echo $f | awk '{printf(``%s'',substr($1,1,6))}'`
  cat << EOF > $date.mon.MK.html
<HEAD>
<TITLE>Chile Monitor Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<IMG
SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$date.monb.MK.gif''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
#
# Mauna Kea Txt Plots
#
foreach f (*.txt.MK.gif)
  cat << EOF > ``$f:r''.html
<HEAD>
<TITLE>Chile Text Plots</TITLE><P ALIGN=CENTER>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<IMG SRC=''http://www.tuc.nrao.edu/~sfoster/tipper/$f''>
<P ALIGN=CENTER>
<ADDRESS>
<a href=''mailto:sfoster@nrao.edu''>sfoster@nrao.edu</a>
</ADDRESS>
</BODY>
EOF
end
#
# Build monitor.html
#
cat << EOF > monitor.html
```

```
<HEAD>
<TITLE>NRAO 225 GHz Tipping Radiometer Raw Data Page</TITLE>
</HEAD>
<BODY bgcolor=#f0f0f0 vlink=30b420>
<H1 ALIGN=CENTER>NRAO 225 GHz Tipping Radiometer Raw Data</H1>
<H1 ALIGN=CENTER>Chile Monitor Data</H1>
<P ALIGN=CENTER>
<CENTER>
<TABLE Border=1>
EOF
foreach date (`cat $tmp.1`)
  cat << EOF >> monitor.html
<TR><TD>$date</TD>
<TD><A HREF=''$date.tau.C.html''>Opacity</A></TD>
<TD><A HREF=''$date.mon.C.html''>Monitor</A></TD>
<TD><A HREF=''$date.txt.C.html''>Phase</A></TD>
<TD><A HREF=''$date.wnd.C.html''>Wind Speed</A></TD>
<TD><A HREF=''$date.wdr.C.html''>Wind Direction</A></TD></TR>
EOF
end
cat << EOF >> monitor.html
</TABLE>
</CENTER>
</P>
<H1 ALIGN=CENTER>Mauna Kea Monitor Data</H1>
<P ALIGN=CENTER>
<CENTER>
<TABLE Border=1>
EOF
foreach date (`cat $tmp.2`)
  cat << EOF >> monitor.html
<TR><TD>$date</TD>
<TD><A HREF=''$date.tau.MK.html''>Opacity</A></TD>
<TD><A HREF=''$date.mon.MK.html''>Monitor</A></TD>
<TD><A HREF=''$date.txt.MK.html''>Phase</A></TD></TR>
EOF
end
cat << EOF >> monitor.html
</TABLE>
```

```
</CENTER>
</P>
</BODY>
EOF
rm $tmp.*
```

O MKweather.csh

```
#!/bin/csh

cd ~sfoster/MKweather

set tmp = tmp$$

set yr = `date +%y`
set mo = `date +%m`
set dy = `date +%d`

set days = (01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31)

set mon = `echo $mo | awk -f month.awk`
foreach d ($days)
  if ( $d > $dy ) break
  echo antennas MK > $tmp.sara
  echo mondat /jansky/mdata/ > $tmp.sara
  echo uttimerange 19$yr$mon$d 00:00:00 to d 24:00:00 >> $tmp.sara
  echo typecode WEA >> $tmp.sara
  echo sound off >> $tmp.sara
  echo outfile $tmp.1 >> $tmp.sara
  echo go >> $tmp.sara
  echo quit >> $tmp.sara
  sara < $tmp.sara >& /dev/null
  awk '($1==''MK'')&&($5==''7F'') {print}' $tmp.1 > $yr$mo$d.wea
  rm $tmp.1
end

set line = `echo $yr $mo | awk -f prev.awk`
```

```
set yr = $line[1]
set mo = $line[2]
set max = $line[3]
set mon = `echo $mo | awk -f month.awk`
foreach d ($days)
  if ( $d > $max ) break
  echo antennas MK > $tmp.sara
  echo mondat /jansky/mdata/ > $tmp.sara
  echo uttimerange 19$yr$mon$d 00:00:00 to d 24:00:00 >> $tmp.sara
  echo typecode WEA >> $tmp.sara
  echo sound off >> $tmp.sara
  echo outfile $tmp.1 >> $tmp.sara
  echo go >> $tmp.sara
  echo quit >> $tmp.sara
  sara < $tmp.sara >& /dev/null
  awk '($1==''MK'')&&($5==''7F'') {print}' $tmp.1 > $yr$mo$d.wea
  rm $tmp.1
end

rm /var/spool/ftp/pub/sfoster/*.wea
foreach f (*.wea)
  if (!(-z $f)) mv $f /var/spool/ftp/pub/sfoster
end

rm $tmp.* *.wea
```

P tauplot.csh

```
#!/bin/csh

if(($#argv > 2 )||($#argv == 0 )) then
  echo ``Usage: tauplot.csh yymmdd <device>''
  exit(1)
endif
if( $#argv == 1 ) then
  set dev = ``/ps -x''
endif
if( $#argv == 2 ) then
```

```
  set dev = $argv[2]
endif
set date = $argv[1]
set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`

set tmp = tmp$$

echo data archive/$yymm/$date.dat > $tmp.1
cat tauplot1.aux >> $tmp.1
echo data archive/$yymm/$date.mon >> $tmp.1
cat tauplot2.aux >> $tmp.1
echo mtext B 3.2 0.5 0.5 Chile Opacities for $date >> $tmp.1
if( $#argv == 2 ) then
  echo quit >> $tmp.1
endif
wip -d $dev $tmp.1 >& /dev/null

rm $tmp.*
```

Q monplot.csh

```
#!/bin/csh

if(($#argv > 3 )||($#argv == 0 )) then
  echo ``Usage: monplot.csh yymmdd <device> <device>''
  exit(1)
endif
if( $#argv == 1 ) then
  set deva = ``/vps -x''
  set devb = ``/vps -x''
endif
if( $#argv == 2 ) then
  set deva = $argv[2]
  set devb = $argv[2]
endif
if( $#argv == 3 ) then
  set deva = $argv[2]
  set devb = $argv[3]
```

```
endif

set date = $argv[1]
set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`

set tmp = tmp$$

echo data archive/$yymm/$date.mon > $tmp.1
cat monplot1.aux >> $tmp.1
echo mtext R 2.2 0.5 0.5 Chile Monitor File Data for $date >> $tmp.1
if(( $#argv == 2 )||( $#argv == 3 )) then
  echo quit >> $tmp.1
endif
wip -d $deva $tmp.1 >& /dev/null
sleep 5
echo data archive/$yymm/$date.mon > $tmp.1
cat monplot2.aux >> $tmp.1
echo mtext R 2.2 0.5 0.5 Chile Monitor File Data for $date >> $tmp.1
if(( $#argv == 2 )||( $#argv == 3 )) then
  echo quit >> $tmp.1
endif
wip -d $devb $tmp.1 >& /dev/null

rm $tmp.*
```

R txtplot.csh

```
#!/bin/csh

if(($#argv > 2 )||($#argv == 0 )) then
  echo ``Usage: txtplot.csh yymmdd <device>''
  exit(1)
endif
if( $#argv == 1 ) then
  set dev = ``/ps -x''
endif
if( $#argv == 2 ) then
  set dev = $argv[2]
```

```
endif

set date = $argv[1]
set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`

set tmp = tmp$$

set count = 1
foreach f (archive/$yymm/$date*.txt)
  set fl = $f:t
  set base = `echo $fl | awk '{printf(``%s\n'',substr($1,1,8))}'`
  set line1 = `head -1 $f`
  if ($line1[1] == Calibration) then
    set crun = 1
    awk '(NR>1) {printf(``%d\t%f\n'',NR-1,$1)}' $f > $tmp.$fl
    if ( -e archive/$yymm/$base.out ) then
      set endtime = `head -2 archive/$yymm/$base.out | tail -1`
    else
      set endtime = ``??:??''
    endif
  else
    set crun = 0
    awk '{printf(``%d\t%f\n'',NR,$1)}' $f > $tmp.$fl
    if ( -e archive/$yymm/$base.out ) then
      set endtime = `head -1 archive/$yymm/$base.out`
    else
      set endtime = ``??:??''
    endif
  endif
  switch ($count)
  case 1:
    echo paper 11 0.77 > $tmp.wip
    echo symbol 17 >> $tmp.wip
    echo font 2 >> $tmp.wip
    echo expand 0.5 >> $tmp.wip
    echo vsize 1.5 4 5.5 7 >> $tmp.wip
    echo data $tmp.$fl >> $tmp.wip
    echo xcol 1 >> $tmp.wip
    echo ycol 2 >> $tmp.wip
```

```
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 2
  breaksw
case 2:
  echo vsize 4.5 7 5.5 7 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 3
  breaksw
case 3:
  echo vsize 7.5 10 5.5 7 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 4
```

```
  breaksw
case 4:
  echo vsize 1.5 4 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 5
  breaksw
case 5:
  echo vsize 4.5 7 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 6
  breaksw
case 6:
  echo vsize 7.5 10 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
```

```
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 7
  breaksw
case 7:
  echo vsize 1.5 4 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 8
  breaksw
case 8:
  echo vsize 4.5 7 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  if ($crun == 1) then
    echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
  endif
  set count = 9
  breaksw
case 9:
  echo vsize 7.5 10 1.5 3 >> $tmp.wip
```

```
    echo data $tmp.$fl >> $tmp.wip
    echo xcol 1 >> $tmp.wip
    echo ycol 2 >> $tmp.wip
    echo limits >> $tmp.wip
    echo box BCNST BCNST >> $tmp.wip
    echo points >> $tmp.wip
    echo limits 0 1 0 1 >> $tmp.wip
    echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
    if ($crun == 1) then
      echo mtext T 0.5 0.05 0.0 Calibration Run >> $tmp.wip
    endif
    echo vsize 1.5 10 1.5 7 >> $tmp.wip
    echo expand 1 >> $tmp.wip
    echo mtext T 2.0 0.5 0.5 Chile Phase Stability Tests for $date >> $tmp.wip
    if( $#argv == 2 ) then
      echo quit >> $tmp.wip
    endif
    wip -d $dev $tmp.wip >& /dev/null
    rm $tmp.wip
    set count = 1
    breaksw
  endsw
end
if (-e $tmp.wip) then
  echo vsize 1.5 10 1.5 7 >> $tmp.wip
  echo expand 1 >> $tmp.wip
  echo mtext T 2.0 0.5 0.5 Chile Phase Stability Tests for $date >> $tmp.wip
  if( $#argv == 2 ) then
    echo quit >> $tmp.wip
  endif
  wip -d $dev $tmp.wip >& /dev/null
endif
rm $tmp.*
cd ..
```

S wndplot.csh

```
#!/bin/csh
```

```
if(($#argv > 2 )||($#argv == 0 )) then
  echo ``Usage: wndplot.csh yymmdd <device>''
  exit(1)
endif
if( $#argv == 1 ) then
  set dev = ``/ps -x''
endif
if( $#argv == 2 ) then
  set dev = $argv[2]
endif

set date = $argv[1]
set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`

set tmp = tmp$$

set count = 1
foreach f (archive/$yymm/$date*.wnd)
  set fl = $f:t
  set base = `echo $fl | awk '{printf(``%s\n'',substr($1,1,8))}'`
  if ( -e archive/$yymm/$base.out ) then
    set line1 = `head -1 archive/$yymm/$base.out`
    if ($line1[1] == Calibration) then
      set endtime = `head -2 archive/$yymm/$base.out | tail -1`
    else
      set endtime = `head -1 archive/$yymm/$base.out`
    endif
  else
    set endtime = ``??:??''
  endif
  awk '{printf(``%d\t%f\t%f\n'',NR,$1,$2)}' $f > $tmp.$fl
  switch ($count)
  case 1:
    echo paper 11 0.77 > $tmp.wip
    echo font 2 >> $tmp.wip
    echo expand 0.5 >> $tmp.wip
    echo vsize 1.5 4 5.5 7 >> $tmp.wip
    echo data $tmp.$fl >> $tmp.wip
```

```
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 2
  breaksw
case 2:
  echo vsize 4.5 7 5.5 7 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 3
  breaksw
case 3:
  echo vsize 7.5 10 5.5 7 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 4
  breaksw
case 4:
  echo vsize 1.5 4 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
```

```
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 5
  breaksw
case 5:
  echo vsize 4.5 7 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 6
  breaksw
case 6:
  echo vsize 7.5 10 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 7
  breaksw
case 7:
  echo vsize 1.5 4 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
```

```
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 8
  breaksw
case 8:
  echo vsize 4.5 7 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 9
  breaksw
case 9:
  echo vsize 7.5 10 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  echo vsize 1.5 10 1.5 7 >> $tmp.wip
  echo expand 1 >> $tmp.wip
  echo mtext T 2.0 0.5 0.5 Chile Wind Speed Data for $date >> $tmp.wip
  if( $#argv == 2 ) then
    echo quit >> $tmp.wip
```

```
    endif
    wip -d $dev $tmp.wip >& /dev/null
    rm $tmp.wip
    set count = 1
    breaksw
  endsw
end
if (-e $tmp.wip) then
  echo vsize 1.5 10 1.5 7 >> $tmp.wip
  echo expand 1 >> $tmp.wip
  echo mtext T 2.0 0.5 0.5 Chile Wind Speed Data for $date >> $tmp.wip
  if( $#argv == 2 ) then
    echo quit >> $tmp.wip
  endif
  wip -d $dev $tmp.wip >& /dev/null
endif
rm $tmp.*
cd ..
```

T wdrplot.csh

```
#!/bin/csh

if(($#argv > 2 )||($#argv == 0 )) then
  echo ``Usage: wdrplot.csh yymmdd <device>''
  exit(1)
endif
if( $#argv == 1 ) then
  set dev = ``/ps -x''
endif
if( $#argv == 2 ) then
  set dev = $argv[2]
endif

set date = $argv[1]
set yymm = `echo $date | awk '{printf(``%s'',substr($1,1,4))}'`

set tmp = tmp$$
```

```
set count = 1
foreach f (archive/$yymm/$date*.wnd)
  set fl = $f:t
  set base = `echo $fl | awk '{printf(``%s\n'',substr($1,1,8))}'`
  if ( -e archive/$yymm/$base.out ) then
    set line1 = `head -1 archive/$yymm/$base.out`
    if ($line1[1] == Calibration) then
      set endtime = `head -2 archive/$yymm/$base.out | tail -1`
    else
      set endtime = `head -1 archive/$yymm/$base.out`
    endif
  else
    set endtime = ``??:??''
  endif
  awk '{printf(``%d\t%f\t%f\n'',NR,$1,$2)}' $f > $tmp.$fl
  switch ($count)
  case 1:
    echo paper 11 0.77 > $tmp.wip
    echo font 2 >> $tmp.wip
    echo expand 0.5 >> $tmp.wip
    echo vsize 1.5 4 5.5 7 >> $tmp.wip
    echo data $tmp.$fl >> $tmp.wip
    echo xcol 1 >> $tmp.wip
    echo ycol 3 >> $tmp.wip
    echo limits >> $tmp.wip
    echo box BCNST BCNST >> $tmp.wip
    echo symbol 17 >> $tmp.wip
    echo points >> $tmp.wip
    echo limits 0 1 0 1 >> $tmp.wip
    echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
    set count = 2
    breaksw
  case 2:
    echo vsize 4.5 7 5.5 7 >> $tmp.wip
    echo data $tmp.$fl >> $tmp.wip
    echo xcol 1 >> $tmp.wip
    echo ycol 3 >> $tmp.wip
    echo limits >> $tmp.wip
```

```
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 3
  breaksw
case 3:
  echo vsize 7.5 10 5.5 7 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 4
  breaksw
case 4:
  echo vsize 1.5 4 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 5
  breaksw
case 5:
  echo vsize 4.5 7 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
```

```
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 6
  breaksw
case 6:
  echo vsize 7.5 10 3.5 5 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 7
  breaksw
case 7:
  echo vsize 1.5 4 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
  echo box BCNST BCNST >> $tmp.wip
  echo symbol 17 >> $tmp.wip
  echo points >> $tmp.wip
  echo limits 0 1 0 1 >> $tmp.wip
  echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
  set count = 8
  breaksw
case 8:
  echo vsize 4.5 7 1.5 3 >> $tmp.wip
  echo data $tmp.$fl >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 3 >> $tmp.wip
  echo limits >> $tmp.wip
```

```
    echo box BCNST BCNST >> $tmp.wip
    echo symbol 17 >> $tmp.wip
    echo points >> $tmp.wip
    echo limits 0 1 0 1 >> $tmp.wip
    echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
    set count = 9
    breaksw
  case 9:
    echo vsize 7.5 10 1.5 3 >> $tmp.wip
    echo data $tmp.$fl >> $tmp.wip
    echo xcol 1 >> $tmp.wip
    echo ycol 3 >> $tmp.wip
    echo limits >> $tmp.wip
    echo box BCNST BCNST >> $tmp.wip
    echo symbol 17 >> $tmp.wip
    echo points >> $tmp.wip
    echo limits 0 1 0 1 >> $tmp.wip
    echo mtext T 0.5 0.95 1.0 $endtime >> $tmp.wip
    echo vsize 1.5 10 1.5 7 >> $tmp.wip
    echo expand 1 >> $tmp.wip
    echo mtext T 2.0 0.5 0.5 Chile Wind Direction Data for $date >> $tmp.wip
    if( $#argv == 2 ) then
      echo quit >> $tmp.wip
    endif
    wip -d $dev $tmp.wip >& /dev/null
    rm $tmp.wip
    set count = 1
    breaksw
  endsw
end
if (-e $tmp.wip) then
  echo vsize 1.5 10 1.5 7 >> $tmp.wip
  echo expand 1 >> $tmp.wip
  echo mtext T 2.0 0.5 0.5 Chile Wind Direction Data for $date >> $tmp.wip
  if( $#argv == 2 ) then
    echo quit >> $tmp.wip
  endif
  wip -d $dev $tmp.wip >& /dev/null
endif
```

```
rm $tmp.*
cd ..
```

U hist.csh

```
#!/bin/csh

if($#argv != 10) then
  echo Usage: hist.csh dbase col xmin xmax hbins cdfflag upflag xlabel title outfile
  exit(1)
endif

set dbase = $argv[1]
set col = $argv[2]
set xmin = $argv[3]
set xmax = $argv[4]
set hbins = $argv[5]
set cdfflag = $argv[6]
set upflag = $argv[7]
set xlabel = ``$argv[8]''
set title = ``$argv[9]''
set outfile = $argv[10]

set ymin = 0
set tmp = tmp$$

set ymax = `peak $dbase $col $hbins $xmin $xmax | awk '{print $2*1.1}'`
echo data $dbase > $tmp.wip
echo font 2 >> $tmp.wip
echo limits $xmin $xmax $ymin $ymax >> $tmp.wip
echo xcol $col >> $tmp.wip
echo expand 0.75 >> $tmp.wip
echo vsize 1 7 5.5 9 >> $tmp.wip
echo box BCNST,BCNST >> $tmp.wip
echo histogram $xmin $xmax $hbins >> $tmp.wip
echo xlabel $xlabel >> $tmp.wip
echo ylabel Counts >> $tmp.wip
echo expand 1 >> $tmp.wip
```

```
echo mtext T 2.0 0.5 0.5 $title >> $tmp.wip
if($upflag) then
  set up = `tipper_uptime $dbase $col -999 | awk '{print $2}'`
  echo expand 0.75 >> $tmp.wip
  echo mtext T 1.5 0.5 0.5 The instrument was running $up% of the time >> $tmp.wip
  echo expand 1 >> $tmp.wip
endif
echo id >> $tmp.wip
if($cdfflag) then
  set line = `cdf $dbase $col 2000 $xmin $xmax $tmp.1 -999`
  echo data $tmp.1 >> $tmp.wip
  echo  expand 0.75 >> $tmp.wip
  echo  vsize 1 7 1 4.5 >> $tmp.wip
  echo xcol 1 >> $tmp.wip
  echo ycol 2 >> $tmp.wip
  echo limits $xmin $xmax 0 1.1 >> $tmp.wip
  echo box BCNST,BCNST >> $tmp.wip
  echo connect >> $tmp.wip
  echo xlabel $xlabel >> $tmp.wip
  echo ylabel Cumulative Distribution Function >> $tmp.wip
  echo limits 0 10 0 10 >> $tmp.wip
  echo move 8 1 >> $tmp.wip
  echo label 25% ... $line[2] >> $tmp.wip
  echo move 8 2 >> $tmp.wip
  echo label 50% ... $line[3] >> $tmp.wip
  echo move 8 3 >> $tmp.wip
  echo label 75% ... $line[4] >> $tmp.wip
  echo move 8 4 >> $tmp.wip
  echo label Quartiles >> $tmp.wip
endif
echo quit >> $tmp.wip

wip -d $outfile/vps $tmp.wip
rm $tmp.*
```

V time.csh

```
#!/bin/csh
```

```
if(($#argv != 9)&&($#argv != 7)) then
 echo ‘‘Usage: time.csh dbase xcol ycol [ymin ymax] xlabel ylabel title outfile’’
 exit(1)
endif

if($#argv == 7) then
  set dbase = $argv[1]
  set xcol = $argv[2]
  set ycol = $argv[3]
  set ymin = auto
  set ymax = auto
  set xlabel = ‘‘$argv[6]’’
  set ylabel = ‘‘$argv[7]’’
  set title = ‘‘$argv[8]’’
  set outfile = $argv[9]
endif

if($#argv == 9) then
  set dbase = $argv[1]
  set xcol = $argv[2]
  set ycol = $argv[3]
  set ymin = $argv[4]
  set ymax = $argv[5]
  set xlabel = ‘‘$argv[6]’’
  set ylabel = ‘‘$argv[7]’’
  set title = ‘‘$argv[8]’’
  set outfile = $argv[9]
endif

set tmp = tmp$$

echo data $dbase > $tmp.wip
echo font 2 >> $tmp.wip
echo xcol $xcol >> $tmp.wip
echo ycol $ycol >> $tmp.wip
if(ymin == auto) then
  echo limits >> $tmp.wip
else
```

```
  echo limits 0 31 $ymin $ymax >> $tmp.wip
endif
echo box BCNST,BCNST >> $tmp.wip
echo symbol 1 >> $tmp.wip
echo points >> $tmp.wip
echo xlabel $xlabel >> $tmp.wip
echo ylabel $ylabel >> $tmp.wip
echo mtext T 2.0 0.5 0.5 $title >> $tmp.wip
echo id >> $tmp.wip
echo quit >> $tmp.wip

wip -d $outfile/ps $tmp.wip
rm $tmp.*
```

W Cdb.csh

```
#!/bin/csh -f

set tmp = tmp$$

if(-e Makefile) grep -v ``all:'' Makefile | grep -v ``dbcat.csh'' >
$tmp.1

set months = `archive_dates.perl archive`

echo $months
foreach yymm ($months)
  set n = `grep $yymm $tmp.1 | wc -l`
  if( $n < 1 ) then
    echo ``# $yymm'' >> $tmp.1
    echo $yymm | awk
'{printf(``database/Merged.%s.C:\tarchive/%s/%s*.dat archive
/%s/%s*.wea\n\tmonth.csh %s\n'',$1,$1,$1,$1,$1,$1)}' >> $tmp.1
    echo $yymm | awk '{printf(``\tsummary.csh %s\n\n'',$1)}' >> $tmp.1
  endif
end

echo -n ``all: `` >> $tmp.1
```

```
foreach yymm ($months)
  echo -n ``database/Merged.$yymm.C `` >> $tmp.1
end
echo ``'' >> $tmp.1
echo 1 | awk '{printf(``\tdbcat.csh\n'')}' >> $tmp.1

mv $tmp.1 Makefile

make all
```

X tipper.csh

```
#!/bin/csh
#
# This script goes and finds all of the plots which need to be added to the
# site testing page and copies them to the appropriate directories.  It should
# be run before mksites.csh.
#
# SMF 10/1/95
#

echo ``Do not run this script unless you meet one or more of the following''
echo ``criteria:''
echo ``''
echo ``  A. You are Scott Foster.''
echo ``  B. You are acting under Scott Foster's instructions.''
echo ``  C. You are really sure you know what you are doing and are willing to''
echo ``     risk Scott Foster's wrath.''
echo ``''
echo -n ``Continue (y/n)? ``
set flag = $<

if (($flag != 'y')&&($flag != 'Y')) exit(0)

set Cdir = $work/tipper/Chile/figures
set MKdir = $work/tipper/MaunaKea/figures
set gifdir = /home/heineken/ftp/observerinfo/mma/sites
set ftpdir = /home/heineken/ftp/mma/sites
```

```
set tmp = tmp$$

set maxdate = `date -u +%y%m`

#
# Add Opacity Histograms
#
set months = `ls $Cdir/opacity.*.C.gif | awk '{printf(``%s\n'',substr($1,index($1,``.
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $Cdir/opacity.$yymm.C.gif $gifdir
  cp $Cdir/opacity.$yymm.C.ps $ftpdir/Chile
end
set months = `ls $MKdir/opacity.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1,'
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $MKdir/opacity.$yymm.MK.gif $gifdir
  cp $MKdir/opacity.$yymm.MK.ps $ftpdir/MaunaKea
end
#
# Add Temperature Histograms
#
set months = `ls $Cdir/temperature.*.C.gif | awk '{printf(``%s\n'',substr($1,index($1
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $Cdir/temperature.$yymm.C.gif $gifdir
  cp $Cdir/temperature.$yymm.C.ps $ftpdir/Chile
end
set months = `ls $MKdir/temperature.*.MK.gif | awk '{printf(``%s\n'',substr($1,index(
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $MKdir/temperature.$yymm.MK.gif $gifdir
  cp $MKdir/temperature.$yymm.MK.ps $ftpdir/MaunaKea
end
#
# Add Wind Speed Histograms
#
set months = `ls $Cdir/wind_speed.*.C.gif | awk '{printf(``%s\n'',substr($1,index($1,
foreach yymm ($months)
```

```
  if($yymm == $maxdate) break
  cp $Cdir/wind_speed.$yymm.C.gif $gifdir
  cp $Cdir/wind_speed.$yymm.C.ps $ftpdir/Chile
end
set months = `ls $MKdir/wind_speed.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $MKdir/wind_speed.$yymm.MK.gif $gifdir
  cp $MKdir/wind_speed.$yymm.MK.ps $ftpdir/MaunaKea
end
#
# Add Wind Direction Histograms
#
set months = `ls $Cdir/wind_direction.*.C.gif | awk '{printf(``%s\n'',substr($1,index
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $Cdir/wind_direction.$yymm.C.gif $gifdir
  cp $Cdir/wind_direction.$yymm.C.ps $ftpdir/Chile
end
set months = `ls $MKdir/wind_direction.*.MK.gif | awk '{printf(``%s\n'',substr($1,ind
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $MKdir/wind_direction.$yymm.MK.gif $gifdir
  cp $MKdir/wind_direction.$yymm.MK.ps $ftpdir/MaunaKea
end
#
# Add Dew Point Histograms
#
set months = `ls $MKdir/dew_point.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1
foreach yymm ($months)
  if($yymm == $maxdate) break
  cp $MKdir/dew_point.$yymm.MK.gif $gifdir
  cp $MKdir/dew_point.$yymm.MK.ps $ftpdir/MaunaKea
end
#
# Add Pressure Histograms
#
set months = `ls $MKdir/pressure.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1,
foreach yymm ($months)
```

```
  if($yymm == $maxdate) break
  cp $MKdir/pressure.$yymm.MK.gif $gifdir
  cp $MKdir/pressure.$yymm.MK.ps $ftpdir/MaunaKea
end
```

Y mksites.csh

```
#!/bin/csh -f
#
# This script updates the site testing web page.
# It should be run each time data is added.
#
# Scott Foster 9/18/95
#

set gifdir = /home/heineken/ftp/observerinfo/mma/sites
set gifurl = http://www.tuc.nrao.edu/mma/sites
set ftpdir = ftp://ftp.tuc.nrao.edu/mma/sites

set tmp = tmp$$

cat sites.1 > $tmp.html

#
# Add Chile Opacity Histograms
#
echo ``<LI>Monthly Opacity Distributions for: `` >> $tmp.html
set months = `ls $gifdir/opacity.*.C.gif | awk '{printf(``%s\n'',substr($1,index($1,'
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$yymm.C.html''\"'
  echo -n ``<TITLE>Chile Opacity Data for $mlabel</TITLE>'' > opacity.$yymm.C.html
```

```
  echo ``<P ALIGN=CENTER>'' >> opacity.$yymm.C.html
  echo ``<IMG SRC=''\"''$gifurl/opacity.$yymm.C.gif''\"''>'' >> opacity.$yymm.C.html
  echo ``<P ALIGN=CENTER>'' >> opacity.$yymm.C.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/opacity.$prev.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$prev.C.html''\
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/opacity.$next.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$next.C.html''\
  endif
  if(-e $gifdir/phase.$yymm.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$yymm.C.html''\"'
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  opacity.$yymm.C.html
  echo ``<ADDRESS>'' >>  opacity.$yymm.C.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  opaci
  echo ``</ADDRESS>'' >>  opacity.$yymm.C.html
end

#
# Add Chile Phase Histograms
#
echo ``<LI>Monthly Phase Stability Distributions for: `` >> $tmp.html
set months = `ls $gifdir/phase.*.C.gif | awk '{printf(``%s\n'',substr($1,index($1,''.
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$yymm.C.html''\"''>
  echo -n ``<TITLE>Chile Phase Stability Data for $mlabel</TITLE>'' > phase.$yymm.C.h
  echo ``<P ALIGN=CENTER>'' >> phase.$yymm.C.html
  echo ``<IMG SRC=''\"''$gifurl/phase.$yymm.C.gif''\"''>'' >> phase.$yymm.C.html
  echo ``<P ALIGN=CENTER>'' >> phase.$yymm.C.html
```

```
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/phase.$prev.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$prev.C.html''\"'
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/phase.$next.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$next.C.html''\"'
  endif
  if(-e $gifdir/temperature.$yymm.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$yymm.C.htm
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  phase.$yymm.C.html
  echo ``<ADDRESS>'' >>  phase.$yymm.C.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  phase
  echo ``</ADDRESS>'' >>  phase.$yymm.C.html
end

#
# Add Chile Temperature Histograms
#
echo ``<LI>Monthly Temperature Distributions for: `` >> $tmp.html
set months = `ls $gifdir/temperature.*.C.gif | awk '{printf(``%s\n'',substr($1,index(
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$yymm.C.html'
  echo -n ``<TITLE>Chile Temperature Data for $mlabel</TITLE>'' > temperature.$yymm.C
  echo ``<P ALIGN=CENTER>'' >> temperature.$yymm.C.html
  echo ``<IMG SRC=''\"''$gifurl/temperature.$yymm.C.gif''\"''>'' >> temperature.$yymm
  echo ``<P ALIGN=CENTER>'' >> temperature.$yymm.C.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/temperature.$prev.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$prev.C.htm
```

```
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/temperature.$next.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$next.C.htm
  endif
  if(-e $gifdir/wind_speed.$yymm.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$yymm.C.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  temperature.$yymm.C.html
  echo ``<ADDRESS>'' >>  temperature.$yymm.C.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  tempe
  echo ``</ADDRESS>'' >>  temperature.$yymm.C.html
end

#
# Add Chile Wind Speed Histograms
#
echo ``<LI>Monthly Wind_Speed Distributions for: `` >> $tmp.html
set months = `ls $gifdir/wind_speed.*.C.gif | awk '{printf(``%s\n'',substr($1,index($
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$yymm.C.html''
  echo -n ``<TITLE>Chile Wind Speed Data for $mlabel</TITLE>'' > wind_speed.$yymm.C.h
  echo ``<P ALIGN=CENTER>'' >> wind_speed.$yymm.C.html
  echo ``<IMG SRC=''\"''$gifurl/wind_speed.$yymm.C.gif''\"''>'' >> wind_speed.$yymm.C
  echo ``<P ALIGN=CENTER>'' >> wind_speed.$yymm.C.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/wind_speed.$prev.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$prev.C.html
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/wind_speed.$next.C.gif) then
```

```
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$next.C.html
  endif
  if(-e $gifdir/wind_direction.$yymm.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$yymm.C.
C.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  wind_speed.$yymm.C.html
  echo ``<ADDRESS>'' >>  wind_speed.$yymm.C.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  wind_
  echo ``</ADDRESS>'' >>  wind_speed.$yymm.C.html
end

#
# Add Chile Wind_Direction Histograms
#
echo ``<LI>Monthly Wind Direction Distributions for: `` >> $tmp.html
set months = `ls $gifdir/wind_direction.*.C.gif | awk '{printf(``%s\n'',substr($1,ind
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$yymm.C.ht
  echo -n ``<TITLE>Chile Wind Direction Data for $mlabel</TITLE>'' > wind_direction.$
  echo ``<P ALIGN=CENTER>'' >> wind_direction.$yymm.C.html
  echo ``<IMG SRC=''\"''$gifurl/wind_direction.$yymm.C.gif''\"''>'' >> wind_direction
  echo ``<P ALIGN=CENTER>'' >> wind_direction.$yymm.C.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/wind_direction.$prev.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$prev.C.
html
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/wind_direction.$next.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$next.C.
```

```
  endif
  if(-e $gifdir/opacity.$yymm.C.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$yymm.C.html''\
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  wind_direction.$yymm.C.h
  echo ``<ADDRESS>'' >>  wind_direction.$yymm.C.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  wind_
  echo ``</ADDRESS>'' >>  wind_direction.$yymm.C.html
end

cat sites.2 >> $tmp.html

#
# Add Mauna Kea Opacity Histograms
#
echo ``<LI>Monthly Opacity Distributions for: `` >> $tmp.html
set months = `ls $gifdir/opacity.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1,
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$yymm.MK.html''\"
  echo -n ``<TITLE>Mauna Kea Opacity Data for $mlabel</TITLE>'' > opacity.$yymm.MK.ht
  echo ``<P ALIGN=CENTER>'' >> opacity.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/opacity.$yymm.MK.gif''\"''>'' >> opacity.$yymm.MK.htm
  echo ``<P ALIGN=CENTER>'' >> opacity.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/opacity.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$prev.MK.html''
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/opacity.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$next.MK.html''
  endif
```

```
  if(-e $gifdir/phase.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$yymm.MK.html''\"
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  opacity.$yymm.MK.html
  echo ``<ADDRESS>'' >>  opacity.$yymm.MK.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  opaci
  echo ``</ADDRESS>'' >>  opacity.$yymm.MK.html
end

#
# Add Mauna Kea Phase Stability Histograms
#
echo ``<LI>Monthly Phase Stability Distributions for: `` >> $tmp.html
set months = `ls $gifdir/phase.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1,''
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$yymm.MK.html''\"''
  echo -n ``<TITLE>Mauna Kea Phase Stability Data for $mlabel</TITLE>'' > phase.$yymm
  echo ``<P ALIGN=CENTER>'' >> phase.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/phase.$yymm.MK.gif''\"''>'' >> phase.$yymm.MK.html
  echo ``<P ALIGN=CENTER>'' >> phase.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/phase.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$prev.MK.html''\"
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/phase.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/phase.$next.MK.html''\"
  endif
  if(-e $gifdir/temperature.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$yymm.MK.ht
  endif
```

```
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  phase.$yymm.MK.html
  echo ``<ADDRESS>'' >>  phase.$yymm.MK.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  phase
  echo ``</ADDRESS>'' >>  phase.$yymm.MK.html
end

#
# Add Mauna Kea Temperature Histograms
#
echo ``<LI>Monthly Temperature Distributions for: `` >> $tmp.html
set months = `ls $gifdir/temperature.*.MK.gif | awk '{printf(``%s\n'',substr($1,index
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$yymm.MK.html
  echo -n ``<TITLE>Mauna Kea Temperature Data for $mlabel</TITLE>'' > temperature.$yy
  echo ``<P ALIGN=CENTER>'' >> temperature.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/temperature.$yymm.MK.gif''\"''>'' >> temperature.$yym
  echo ``<P ALIGN=CENTER>'' >> temperature.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/temperature.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$prev.MK.ht
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/temperature.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/temperature.$next.MK.ht
  endif
  if(-e $gifdir/dew_point.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/dew_point.$yymm.MK.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  temperature.$yymm.MK.htm
  echo ``<ADDRESS>'' >>  temperature.$yymm.MK.html
```

```
  echo ‘‘<a href=’’\"’’mailto:sfoster@nrao.edu’’\"’’>sfoster@nrao.edu</a>’’ >>  tempe
  echo ‘‘</ADDRESS>’’ >>  temperature.$yymm.MK.html
end

#
# Add Mauna Kea Dew Point Histograms
#
echo ‘‘<LI>Monthly Dew Point Distributions for: ‘‘ >> $tmp.html
set months = ‘ls $gifdir/dew_point.*.MK.gif | awk ’{printf(‘‘%s\n’’,substr($1,index($
set flag = 1
foreach yymm ($months)
  set mlabel = ‘echo $yymm | awk ’{printf(‘‘%s %s’’,substr($1,1,2),substr($1,3,2))}’
  if($flag) then
    set flag = 0
  else
    echo -n ‘‘, ‘‘ >> $tmp.html
  endif
  echo -n ‘‘<A HREF=’’\"’’http://www.tuc.nrao.edu/mma/sites/dew_point.$yymm.MK.html’’
  echo -n ‘‘<TITLE>Mauna Kea Dew Point Data for $mlabel</TITLE>’’ > dew_point.$yymm.M
  echo ‘‘<P ALIGN=CENTER>’’ >> dew_point.$yymm.MK.html
  echo ‘‘<IMG SRC=’’\"’’$gifurl/dew_point.$yymm.MK.gif’’\"’’>’’ >> dew_point.$yymm.MK
  echo ‘‘<P ALIGN=CENTER>’’ >> dew_point.$yymm.MK.html
  set prev = ‘echo $yymm | awk -f prev.awk‘
  if(-e $gifdir/dew_point.$prev.MK.gif) then
    echo -n ‘‘<A HREF=’’\"’’http://www.tuc.nrao.edu/mma/sites/dew_point.$prev.MK.html
  endif
  set next = ‘echo $yymm | awk -f next.awk‘
  if(-e $gifdir/dew_point.$next.MK.gif) then
    echo -n ‘‘<A HREF=’’\"’’http://www.tuc.nrao.edu/mma/sites/dew_point.$next.MK.html
  endif
  if(-e $gifdir/pressure.$yymm.MK.gif) then
    echo -n ‘‘<A HREF=’’\"’’http://www.tuc.nrao.edu/mma/sites/pressure.$yymm.MK.html’
  endif
  echo -n ‘‘<A HREF=’’\"’’http://www.tuc.nrao.edu/mma/sites/sites.html’’\"’’>Site Tes
  echo ‘‘<A HREF=’’\"’’$ftpdir’’\"’’>FTP Directory</A>’’ >>  dew_point.$yymm.MK.html
  echo ‘‘<ADDRESS>’’ >>  dew_point.$yymm.MK.html
  echo ‘‘<a href=’’\"’’mailto:sfoster@nrao.edu’’\"’’>sfoster@nrao.edu</a>’’ >>  dew_p
  echo ‘‘</ADDRESS>’’ >>  dew_point.$yymm.MK.html
end
```

```
#
# Add Mauna Kea Pressure Histograms
#
echo ``<LI>Monthly Pressure Distributions for: `` >> $tmp.html
set months = `ls $gifdir/pressure.*.MK.gif | awk '{printf(``%s\n'',substr($1,index($1
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/pressure.$yymm.MK.html''\
  echo -n ``<TITLE>Mauna Kea Pressure Data for $mlabel</TITLE>'' > pressure.$yymm.MK.
  echo ``<P ALIGN=CENTER>'' >> pressure.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/pressure.$yymm.MK.gif''\"''>'' >> pressure.$yymm.MK.h
  echo ``<P ALIGN=CENTER>'' >> pressure.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/pressure.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/pressure.$prev.MK.html'
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/pressure.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/pressure.$next.MK.html'
  endif
  if(-e $gifdir/wind_speed.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$yymm.MK.htm
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  pressure.$yymm.MK.html
  echo ``<ADDRESS>'' >>  pressure.$yymm.MK.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  press
  echo ``</ADDRESS>'' >>  pressure.$yymm.MK.html
end

#
# Add Mauna Kea Wind Speed Histograms
```

```
#
echo ``<LI>Monthly Wind Speed Distributions for: `` >> $tmp.html
set months = `ls $gifdir/wind_speed.*.MK.gif | awk '{printf(``%s\n'',substr($1,index(
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$yymm.MK.html'
  echo -n ``<TITLE>Mauna Kea Wind Speed Data for $mlabel</TITLE>'' > wind_speed.$yymm
  echo ``<P ALIGN=CENTER>'' >> wind_speed.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/wind_speed.$yymm.MK.gif''\"''>'' >> wind_speed.$yymm.
  echo ``<P ALIGN=CENTER>'' >> wind_speed.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/wind_speed.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$prev.MK.htm
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/wind_speed.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_speed.$next.MK.htm
  endif
  if(-e $gifdir/wind_direction.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$yymm.MK
.MK.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  wind_speed.$yymm.MK.html
  echo ``<ADDRESS>'' >>  wind_speed.$yymm.MK.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu''\"''>sfoster@nrao.edu</a>'' >>  wind_
  echo ``</ADDRESS>'' >>  wind_speed.$yymm.MK.html
end

#
# Add Mauna Kea Wind Direction Histograms
#
echo ``<LI>Monthly Wind Direction Distributions for: `` >> $tmp.html
```

```
set months = `ls $gifdir/wind_direction.*.MK.gif | awk '{printf(``%s\n'',substr($1,in
`
set flag = 1
foreach yymm ($months)
  set mlabel = `echo $yymm | awk '{printf(``%s %s'',substr($1,1,2),substr($1,3,2))}'
  if($flag) then
    set flag = 0
  else
    echo -n ``, `` >> $tmp.html
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$yymm.MK.h
  echo -n ``<TITLE>Mauna Kea Wind Direction Data for $mlabel</TITLE>'' > wind_directi
  echo ``<P ALIGN=CENTER>'' >> wind_direction.$yymm.MK.html
  echo ``<IMG SRC=''\"''$gifurl/wind_direction.$yymm.MK.gif''\"''>'' >> wind_directio
  echo ``<P ALIGN=CENTER>'' >> wind_direction.$yymm.MK.html
  set prev = `echo $yymm | awk -f prev.awk`
  if(-e $gifdir/wind_direction.$prev.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$prev.MK
K.html
  endif
  set next = `echo $yymm | awk -f next.awk`
  if(-e $gifdir/wind_direction.$next.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/wind_direction.$next.MK
ml
  endif
  if(-e $gifdir/opacity.$yymm.MK.gif) then
    echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/opacity.$yymm.MK.html''
  endif
  echo -n ``<A HREF=''\"''http://www.tuc.nrao.edu/mma/sites/sites.html''\"''>Site Tes
  echo ``<A HREF=''\"''$ftpdir''\"''>FTP Directory</A>'' >>  wind_direction.$yymm.MK.
  echo ``<ADDRESS>'' >>  wind_direction.$yymm.MK.html
  echo ``<a href=''\"''mailto:sfoster@nrao.edu,mholdawa@nrao.edu''\"''>sfoster@nrao.e
tml
  echo ``</ADDRESS>'' >>  wind_direction.$yymm.MK.html
end

cat sites.3 >> $tmp.html

mv $tmp.html sites.html
```