A NEW CONTINUUM MAPPING PROCEDURE AT HARTRAO

THESIS
Submitted in fulfilment of the
requirements for the Degree of
MASTER OF SCIENCE
of Rhodes University

by

SARAH BÜCHNER

January 1998

Abstract

A basket weaving technique for making radio continuum maps has been developed at the Hartebeesthoek Radio Astronomy Observatory (HartRAO). This data reduction technique significantly reduces scanning effects by using independent maps scanned in orthogonal directions. The observation and data analysis procedures that were developed are presented.

The technique was used to map the supernova remnant MSH 15-52 at frequencies of 5000 MHz and 8500 MHz. The flux spectral index for this supernova remnant was found to be 0.83 ± 0.02 in this frequency range

Two regions (A and B) of the Galactic plane were observed at 8500 MHz with a resolution of 6`. Region A covered the 5°×5° area $47.5° < l < 52.5°$, $|b| < 2.5°$, and region B was the 4.2°×3° area $320.4° < l < 334.6°$, $|b| < 1.5°$. Far infrared observations at 60 μm were used in conjunction with the radio maps to separate the thermal and non-thermal components of the radio emission.

The technique can be used to map the Galactic plane at 8500 MHz using dual polarisation once the receiver at HartRAO has been upgraded. This would fulfil a need for a medium resolution, high frequency survey of the southern Galactic plane.

Table of contents

Acknowledgments

I am very grateful to all those who have supported and encouraged me during the writing of this thesis.

I would like to thank my supervisors, Prof. Eddie Baart and Justin Jonas for their support, assistance and patience; Eddie for arranging financial assistance and Justin for his advice, encouragement and valuable criticism.

A special thanks to George Nicolson for his support and helpful advice.

Lastly I would like to thank my family and friends for their unfailing support, love and encouragement, and most importantly for believing in me.

Chapter 1.

Introduction

There are relatively few radio frequency maps of the southern sky because of the lack of radio observatories in the southern hemisphere. The radio astronomy group of the Rhodes University Department of Physics and Electronics completed a survey of the southern sky in 1992. The 26 m antenna of the Hartebeesthoek radio observatory (HartRAO) in Gauteng, South Africa, was used to map extended emission at high and medium Galactic latitudes at 2326 MHz. This survey, the Rhodes/HartRAO 2326 MHz survey (Jonas et al., 1998), is the highest frequency and highest resolution (20`) large-area radio continuum survey in existence (Jonas, 1995). Increased competition for telescope time at HartRAO and increasing levels of interference from radio transmitters, both satellite and ground-based, make further all-sky continuum surveys unlikely (Jonas, 1995).

Table 1-1 shows a list of surveys made of the Southern Galactic plane. This list indicates the need for a medium resolution (< 10`) survey of the Galactic plane at a frequency above 5000 MHz. This would give us more information about the spectral index of the Galactic emission - knowledge which would be valuable in the study of the Galactic cosmic ray behaviour and would enable the identification of supernovae and thermal sources.

We therefore decided to map the Galactic plane at a frequency of 8500 MHz at which the HartRAO antenna has a resolution of 6`. This survey would be a valuable addition to the existing Galactic surveys mentioned above. Telescope time would be more easily obtained than for an all-sky survey as a much smaller area would be mapped. By using a polarimeter, valuable information can be obtained about magnetic fields of Galactic sources. Although a polarimeter is not yet available at HartRAO it was expected to become available in the near

future. Since the project was started, a survey of the Southern Galactic sky, at 2.4 GHz, with a resolution of 10.4`, has been completed by Duncan et al. (1995) using the Parkes telescope.

Authors	Date	Frequency	Resolution
Cane et al.	1977	2 - 20 MHz	2.5°
Shain et al.	1961	19.7 MHz	1.4°
Jones et al.	1974	29.9 MHz	0.8°
Hill et al.	1958	85 MHz	50`
Green	1974	408 MHz	2.86`
Komesaroff	1966	408 MHz	47.5`
Hill et al.	1968	1410 MHz	14`
Mathewson et al.	1962	1440 MHz	50`
Jonas et al.	1998	2326 MHz	20`
Duncan et al.	1995	2400 MHz	10.4`
Day et al.	1972	2700 MHz	8.2`
Haynes et al.	1978	5000 MHz	4.1`

Table 1-1 Surveys of the Southern Galactic plane (including some all sky surveys)

In radio astronomy, the telescope records the integrated emission from a single region at a time (unlike optical astronomy, in which the intensity of millions of points can be recorded simultaneously). In order to build up a radio image it is necessary to use a scanning technique in which the antenna is scanned across the sky while continuously recording the strength of the emission. A number of these scans are made at intervals of less than half the antenna beam width. The scans can then be combined into an image in the same way as a television picture is built up as a raster scan pattern. However each scan follows a different path and is made at a different time and so is affected by variable atmospheric contributions and by drift in the receiver output. Each scan therefore has a different baseline which leads to artefacts which appear as striations in the map. Even after a linear baseline has been removed from each scan these striations dominate the map masking the underlying structure.

To remove these "scanning effects" I have developed and implemented a mapping technique, known as basket weaving, that uses two maps of the same area - one made by scanning in

latitude, the other in longitude. Backgrounds can be determined by comparing these orthogonal scans at their intersection points. I have tested the method by making four maps of three regions, and found it to be effective.

In this thesis I discuss the development, implementation and testing of the observation and data reduction techniques.

Chapter 2 discusses the design of the basket weaving method, comparing it with other methods of background removal. I developed an observing system, PC-MAP, which makes use of an improved antenna control system (Jonas, 1991) at HartRAO to map small areas of sky. The development and operation of this observation system are discussed in Chapter 3.

Chapter 4 discusses the implementation and development of the basket weaving technique. It is customary to convert radio images from antenna temperatures into full-beam brightness temperatures. This brightness calibration procedure is discussed in Chapter 5.

To test the observing and data analysis systems I observed three regions. First I mapped the supernova remnant MSH 15-52 at 3.5 cm and 6 cm. These results are presented in Chapter 6. Then as a test of Galactic plane mapping I observed two areas of the Galactic plane. These results are presented and discussed in Chapter 7. Finally, Chapter 8 evaluates the system and discusses future developments.

Chapter 2.

Basket weaving design

I wanted to develop a mapping technique which would eventually be used to map the Galactic plane at 8500 MHz. In order to make optimum use of telescope time I wanted to be able to spread observations of a map over a number of observing sessions. Background determination should not therefore rely on a number of scans having the same background and so scans should be independent. Observations of individual scans could be scheduled in between other observing tasks, and later combined into a map.

As mentioned in the previous chapter a raw map made using a scanning technique inevitably contains striations in the scanning direction. Figure 2-1 shows a map which was made using latitude scans. Although independent linear baselines have been fitted to each scan and subtracted, the map is dominated by vertical stripes.

The central concern of the data reduction technique is to remove these scanning effects but to leave the real data unaffected. There are two main causes of scanning effects:

a) Terrestrial contributions to the antenna temperature, which vary with antenna orientation and time.
b) Fluctuations in the gain and system temperature of the receiver.

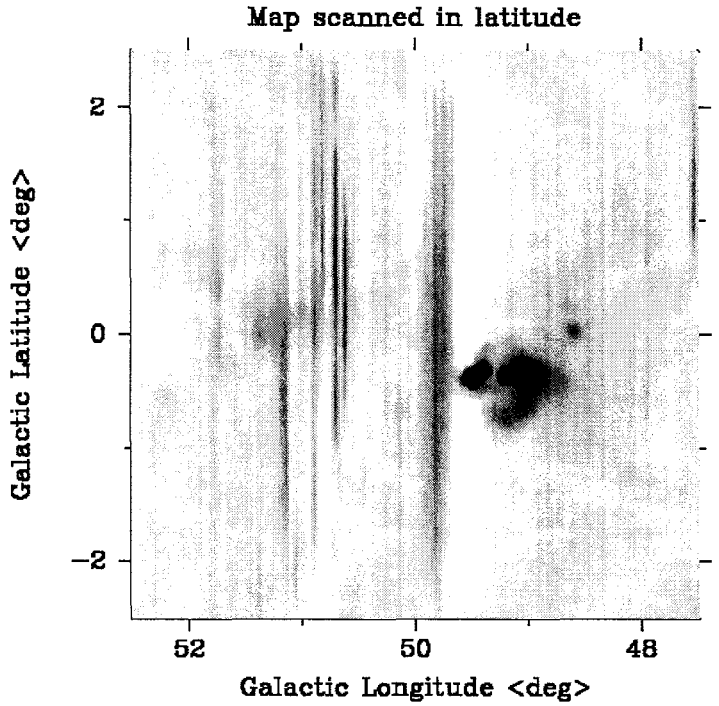


Figure 2-1 Map made at 3.5 cm by making a number of latitude scans (spaced at 0.04° intervals) . The vertical striations are instrumental effects which are still present even after a linear baseline has been subtracted.

In this chapter I will first examine some methods used to remove these backgrounds, including the Skymap technique used in the 2236 MHz survey. Then, in section 2-2, I will outline the basket weaving technique I decided to use.

2.1. Background removal methods

The simplest way of removing background is to subtract a linear baseline, found by fitting a straight line to the end points, from each scan. There are a number of limitations to this method (Emerson and Grave, 1988). If there is emission at the end points of the scan this will distort the baselevel. Noise in the data at the scan ends leads to uncertainties in the baseline which then leads to large scale fluctuations in the data. A linear baseline is not a good approximation for the background profile and if higher order fits are used there is a danger of distorting real features. Although linear baselines remove any slope in the data it leaves scanning effects as shown in figure 2-1.

Skymap was developed by P Mountfort (1989) to map very extended radio sources. Further extensions to the observing and data analysis techniques were made by Jonas (1982). I shall give a brief description, taken from Mountfort (1989) and Jonas (1982), of the background removal techniques used in Skymap.

The maps were made by making a number of scans in declination tracking a constant right ascension. A fast scanning rate was used and three repeat observations were made of each raster to improve the signal to noise ratio of the data. The scans were made alternately up and down in declination. Each upscan in a raster was started at the same hour angle and followed the same path in the sky. All downscans followed the same path, although this was different from the path of the upscans. The terrestrial background is dependent on antenna altitude, and hence on declination and hour angle. The background profile is therefore the same for all scans following the same path in the sky and so the same for all scans in the same direction. If the baseline drift is slow enough it can be assumed to be a function of right ascension, independent of declination. The contributions from the terrestrial background and drift are therefore orthogonal and so can be removed separately.

The drift was removed first. In order to remove it each raster needed to include an area of "cold" (source-free) sky. The data were binned into wide (1.6°) declination bins and drift free sections were selected visually from the three sets of observations. These sections were combined to give a highly smoothed drift-free map. By comparing the wide-binned data to this smoothed map an approximation to the drift could be obtained.

The terrestrial background was then found. Each raster was split into sets of up and down scans. For each set of scans the wide-binned data were compared and a least mean square polynomial of temperature as a function of declination was fitted to the tenth lowest point (to avoid noise) at each declination. This polynomial, representing terrestrial background, was then subtracted together with the drift and the data binned using median binning onto a regular grid.

Another method of determining backgrounds is by making use of reference scans. For example, in the 5 GHz survey by Haynes et al. (1978) tie-down scans were used. These scans ended in source-free areas far from the Galactic plane, where the temperature was set to zero

(ignoring the cosmic microwave background and diffuse Galactic emission), and were used to set the background level in the map. Hill (1968) used reference scans made parallel to the Galactic plane to set baselevels for the 1410 MHz survey of the Galactic plane.

Another method of background determination involves the use of intersecting scans. The 408 MHz survey made by Haslam et al. (1974) used nodding scans. The telescope is scanned in elevation using the rotation of the earth to move it in right ascension. Each upscan intersects with a number of downscans and vice versa. At these intersection points the scans should agree. A least squares polynomial is fitted to the difference between temperatures of an individual scan and those of intersecting scans to determine a baseline. Each intersecting scan has errors in its own baseline but these are statistically independent and so the baseline of each individual scan can be determined. These backgrounds are subtracted and this process iterated to improve the background determination.

This method was later refined by CJ Salter (Emerson and Grave, 1988) into the method, known as basket weaving, which uses a pair of maps scanned in orthogonal directions. This method was used by Sieber et al. (1979) to map supernova CTA1. A basket weaving data analysis program, WEAVE, was developed at the Max Planck Institute for Radio Astronomy (Reich, private communication).

This was later replaced by the program PLAIT, which removes scanning effects in the Fourier domain. The two orthogonal maps are Fourier transformed. In the Fourier domain scanning effects appear in a narrow band along one spatial frequency axis. PLAIT weights down these scanning effects in the Fourier domain and then combines the weighted transforms of the orthogonal maps. The inverse Fourier transform is then taken to give a final map.

I decided to use the basket weaving method for processing the 3.5 cm maps. Advantages of this method are that the processing is automated, does not rely on scans in an observation having a similar background and there is no need for additional reference scans. However the technique eliminates large scale structures (of the order of the extent of the map and bigger) and does not produce an absolutely calibrated radio image. However this is not a problem for my purposes

2.2. Basket weaving

I have adapted and extended WEAVE for use at HartRAO. To make a map of a region of sky I observe the same area twice, first scanning in Galactic latitude, the y-map, and then in Galactic longitude, the x-map. Both of these maps are dominated by striations in the scanning direction. I subtract a linear baseline from each scan. Each scan in the y-map intersects every scan in the x-map. At these intersection points the same position in the sky is observed and hence the values in the x-map and y-map should be identical. However, in practice the maps do not agree at these points because of the difference in backgrounds. The idea of the basket weaving is to correct the background of each scan in the x-map by using corresponding pixels from the y-map and vice versa. Non-real peaks in the latitude background are pulled down and troughs raised by the intersecting longitude scans. By iterating this process the two sets of scans are "woven" together until the differences at the intersection points are consistent with the map noise level.

The y-map and x-map are subtracted from each other to form a difference map. In this difference map the latitude and longitude backgrounds are functions of latitude and longitude respectively. A least squares polynomial is fitted to each row in the difference map to approximate the background of that x-scan. The same is done for the y-scans. These baseline approximations are scaled by some stability factor, S, and then subtracted from the original x and y maps. S is similar to the damping term in a servo equation (Haslam, 1974). A new difference map is then formed using the corrected maps and the process iterated until the difference map has a rms value less than some pre-defined limit. I have set this limit to one tenth of the nominal rms noise of the maps (1 mK at 3.5 cm).

I now needed to develop a technique for implementing this basket weaving method. This consisted of two sets of programs. First, an observing program, PC-MAP, which is written in Turbo Pascal and runs on a PC. This generates commands to map a specified area and record the data. PC-MAP is described in the next chapter. The analysis procedure consists of a suite of programs written in FORTRAN and running on a UNIX machine. These data reduction programs, which implement the basket weaving technique, are described in Chapter 4.

Chapter 3.

Observation Method

In this chapter I describe the observation method developed to map sections of the Galactic plane for processing using the basket weaving method. Major considerations in developing this method were: efficient and flexible use of telescope time and a user friendly interface. In the Skymap program it was necessary to have a large block of uninterrupted telescope time to complete an observation. Because of telescope time constraints I wanted to be able to spread an observation over a number of short observing sessions. Each scan should therefore be independent and be recorded in such a way that "bad" scans could be easily repeated. At the time I developed this observing process Jonas had developed a new PC based control system for the HartRAO antenna , PC-STEER (Jonas, 1991). My observing program would serve as a test for this as it would be the first full scale observing program written to use PC-STEER.

In this chapter I briefly describe the HartRAO telescope and control system and then discuss details of the observing programs.

3.1. HartRAO telescope and control system

The HartRAO antenna has a 26 m parabolic dish with an equatorial mount. The feed optics have a Cassegrain configuration.

As PC-STEER runs under a DOS environment there is no multi-tasking capability. This necessitates the use of separate dedicated computers for the antenna control system, PC-STEER, and the observing program, PC-MAP. Most of the instrumentation used for the

mapping observations was linked via the industry standard IEEE-488 bus. The observing computer is master on the IEEE-488 bus and PC-STEER, the digital voltmeter connected to the radiometer and other I/O are slave devices. The noise diodes can be controlled through a parallel interface on the observing PC.

Commands are passed from the observing computer to the steering computer on the IEEE-488 bus in command blocks written in a structured "pseudo English". Each command block starts with a "BEGIN *name of task*" command and ends with "END". One positioning or scanning task is performed by each block. Blocks may be executed immediately, or up to eight may be queued in a first in first out (FIFO) buffer. The main commands used are listed below:

BEGIN	*name*	start a command block
CLEAR		clear all commands in the buffer
APPEND		append command to buffer
DURATION	*time*	set *time* as duration before command expires
COORD	*system*	set base coordinate system e.g. Galactic
POINT	*on/off*	telescope pointing map is used or not used
FEED	*number*	select receiver feed system *number*
A0	*value*	set *value* as base longitude for this scan
B0	*value*	set *value* as base latitude for this scan
A1	*value*	set longitude scan rate
B1	*value*	set latitude scan rate
END		end a command block

On demand, the steering computer returns an ASCII string containing a number of fields on the IEEE bus, including antenna status, sidereal time and Julian day number, beam offsets and antenna coordinates. All coordinate calculations and corrections for beam offsets and pointing (optional) are done within the control computer and the antenna latitude and longitude are returned in the coordinate system selected by the observer.

The DVM is synchronised to sample the radiometer output whenever the steering computer reads the antenna axis encoders.

3.2. Observing programs

I chose to write the observation programs in Turbo Pascal as I was familiar with the language and found it to be fast, versatile and easy to program. In addition Turbo Pascal drivers were available for the IEEE-488 card.

The observing programs were designed to provide a user friendly interface allowing the observer to continuously monitor the observation. As each scan is completed it is plotted to the screen so problems may be detected immediately and the observation restarted or the scan repeated. Each scan is independent so a map may be observed over a period of several nights, and the scans combined. "Bad" scans due to interference or equipment malfunctions can be easily repeated after the initial completion of the set of scans.

The central mapping program, PC-MAP, is used to map a rectangular area of a specified size using any of the receiver systems. The area is mapped by moving the antenna in either latitude or longitude (the scan coordinate) while keeping the other coordinate constant. A map is built up by making a number of scans in the scan coordinate spaced by some "scan spacing" in the orthogonal non-scan coordinate.

For example, if a map is scanned in Galactic latitude then a number of scans are made at constant Galactic longitude at regular intervals. The scan spacing is chosen so that the map is oversampled. The scan tracks for a map made by scanning in latitude are shown in figure 3-1. During each scan the radiometer voltages and antenna positions are sampled and recorded. Scans are made alternately up and down in the scanning coordinate.

The basic algorithm for the mapping program is shown below.

```
get parameters of map from observer using interactive menu
set up antenna
move antenna to corner of map
for scan = first to scan = last do
        if (scan = first) then
                move to start of first scan
        else
                write previous scan to file
                plot previous scan on screen
        end if

        noise diode calibration

        repeat
                read antenna position
        until (antenna on source)

        begin scanning at scan rate
        append command to move to start of next scan

        while (still scanning)
                read antenna position
                read DVM
        end while
end do
output last scan to file
plot last scan to screen
end
```

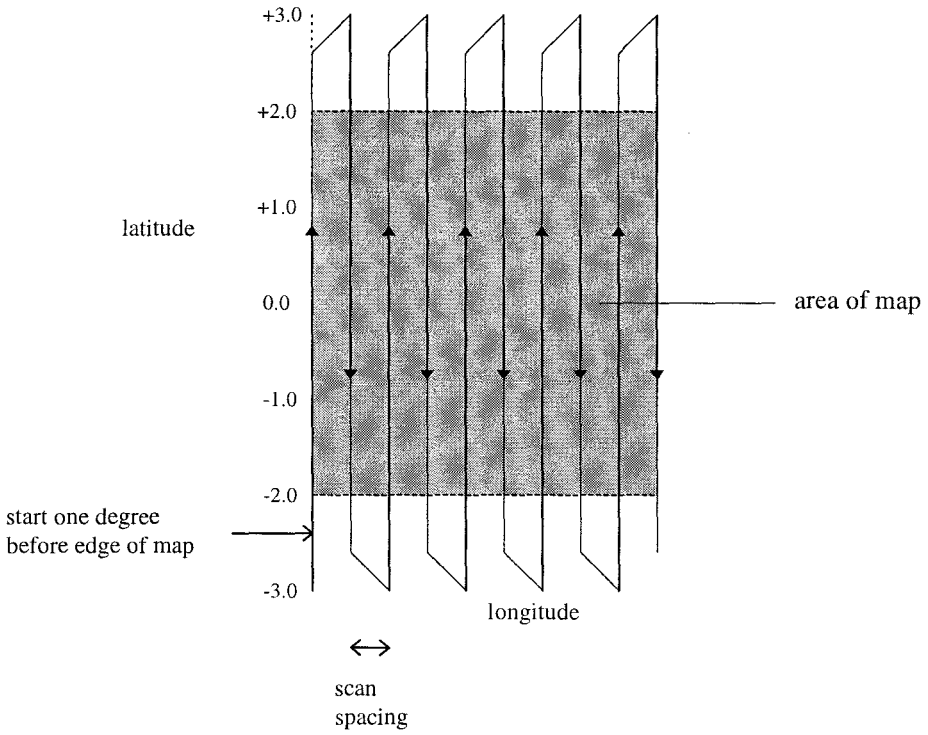


Figure 3-1 Scan pattern for latitude scans. Each scan starts a degree before the edge of the map to give the antenna time to settle down.

The mapping program, MAP, is listed in appendix A (pg. A-1). For clarity I shall divide the mapping procedure into six components and discuss each in turn. First, the observer needs to set the parameters of the map i.e. size, coordinate system, etc. (section 3.2.1). The observing program then generates commands for the control system to map the area (section 3.2.2). During the scan the radiometer is sampled and the voltage readings are stored along with coordinate information (section 3.2.3). The radiometer readings are recorded as voltages and need to be converted to antenna temperatures in Kelvin. This conversion factor is found using a noise diode for calibration (section 3.2.4). At the end of each scan the data values are stored in a text file (section 3.2.5). The scan is displayed on the screen (section 3.2.6).

3.2.1. Parameter input

The observer is prompted for an observation name which is used as a prefix to name data files and for the directory in which the files are stored. The observer then uses an interactive menu (see Fig 3-2) to set the following parameters:

- map coordinates (StartLat, StartLong, EndLat, EndLong) - latitude and longitude boundaries of the map
- scan rate in degrees/sec
- scan spacing - spacing between the scans in the non-scanning coordinate
- range of scans to observe
- feed system
- scan direction - latitude or longitude
- coordinate system - Galactic, equatorial, field centered equatorial

Fig 3-2 shows a menu with responses for making a 4° by 5° map of the area, with latitude range $|b| \leq 2.0°$ and longitude range $50.0° \leq l \leq 55.0°$ in Galactic coordinates. The antenna will be scanned at 0.1° per second in latitude and the scans spaced by 0.04° in longitude. The map is to be made using the 3.5 cm feed system. Only the first 30 scans are to be made in this observing session. The "Read from DVM" parameter can be set to "NO" to allow a dry run, in which the DVM is not sampled.

From these parameters, the length of each scan and the number of scans necessary to complete the map can be calculated. In the above example, each scan has a scan length of 4.0° and there are a total of 126 scans in the map. The expected duration of each scan, 50 s in this case, can be calculated from the scan length and scan rate. A map can be split into smaller sections and the observer may select a group of scans, scans 1 to 30 in this example, to complete in any observing session. These parameters are recorded in an observation log file.

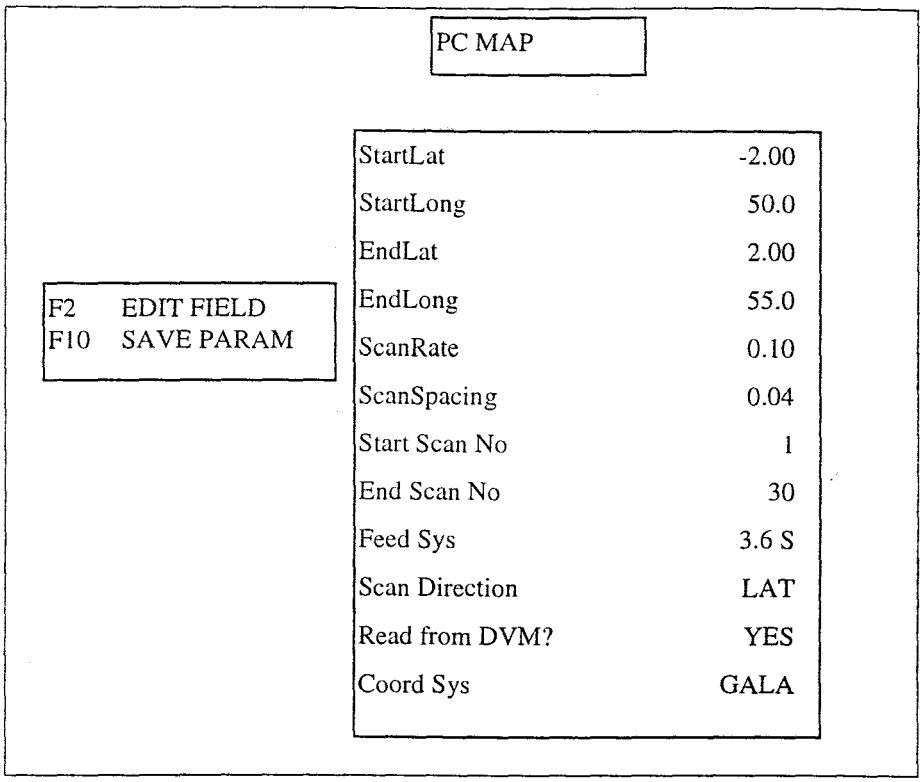


Figure 3-2 Sample of interactive menu from MAP to map the area -2.0° to 2.0° latitude and 50° to 55° longitude. The scan spacing is 0.04°, scan rate is 0.1°.

3.2.2. Mapping command generation

If, for example, the antenna is scanning in latitude then it should track a constant longitude. However the antenna takes a while to settle down at the start of each scan and its path wanders from an ideal straight line by as much as two scan spacings in longitude. Therefore the antenna needs to start scanning one degree before the edge of the map (see figure 3-1) so that when it reaches the edge of the map it is tracking the command longitude within a reasonable tolerance.

At the start of the mapping procedure the antenna is positioned at the start of the first scan. In the example this will be at l = 50° and b = -3° i.e. one degree before the edge of the map. While the antenna is moving into position a noise diode calibration (see section 3.2.4) is performed. Once the antenna is tracking the start position, the following sequence of commands is sent to start the antenna scanning:

BEGIN Scan *ScanNumber*
CLEAR
DURATION (Scan time + 5)
A0 = Long
B0 = Latitude
A1 = Longitude scan rate (zero if scanning in latitude)
B1 = Latitude scan rate (zero if scanning in longitude)
END

This command sequence clears the FIFO of any queued commands. The antenna then starts scanning at the given scan rate. Using the parameters given in figure 3-2, the base longitude (A0) is 50° and the base latitude (B0) is -3°. The longitude and latitude scan rates are 0 and 0.1 degrees/s respectively. The antenna would then scan in latitude at a scan rate of 0.1 degrees/s. The expected time of this scan would be 50 seconds and so the command is set to expire after 55 seconds.

A key issue in designing the observing program is to make the best possible use of telescope time. A large proportion of the total mapping time is spent in turning around at the end of each scan. To reduce the effect of this I put the following command (to drive to the start of the next scan) into the FIFO buffer for immediate execution once the Scan command expires.

BEGIN GoToNextScan
APPEND
DURATION 400s
A0 = Long
A1 = 0
B0 = Lat
B1 = 0
END

In the example the coordinates of the start of the second scan are A0 = 50.04° and B0 = +3.0°. When the Scan command expires (after 55 seconds) the "GoToNextScan" command immediately comes into effect. The antenna overshoots the edge of the map and then continues driving to the start of the next scan. While the antenna is moving, noise diode

calibrations are made and the previous scan is plotted on the screen and the data recorded to disk. The procedure AntennaOnSource (unit IO_UTILS) is then called to wait until the antenna is positioned at the start of the next scan.

3.2.3. Data collection

While the antenna is scanning, the antenna angles and radiometer output voltage need to be sampled. The antenna reply string is read and the antenna status and its latitude and longitude are extracted from this string using the function AntField (unit IO_UTILS). The digital voltmeter attached to the radiometer is also sampled. The voltmeter is sampled 9.1 times per second (each time the antenna angles are read). The antenna coordinates and antenna voltage are buffered in an array within the observing program.

This sampling continues while the scan command is the active command. When this command expires and the GotoNextScan command becomes active then sampling ceases, and the buffered data are written to disk.

3.2.4. Noise diode calibration

It is useful to quantify power received by the antenna in terms of an equivalent temperature - the antenna temperature. The radiometer voltage, in units of mV, can be converted to antenna temperature in K using a calibration noise diode. The noise diodes have a known noise temperature which is calibrated in the laboratory using hot and cold reference loads. The antenna temperature scale can be calibrated by measuring the increase in radiometer output voltage due to the noise diode (Kraus, 1966 Chapter 7). This calibration is done at the start of each scan while waiting for the antenna to settle down at the source using the procedure Calibrate (unit IO_UTILS). The DVM readings are averaged over 10 seconds with the noise diode off, 20 seconds with it on and finally another 10 s with the noise diode off. The average radiometer voltage due to the noise diode is recorded in the scan data file header.

3.2.5. Data storage

At the end of each scan the data are stored in a file as ASCII text. The files are named using the observation name and the scan number. Information including Julian day, sidereal time, coordinate type, beam offsets and feed system is stored in the file header together with the results of the noise calibration. Following the header one sample (DVM voltage, latitude, longitude) is stored per line.

The data are stored in text files as this makes it easier to process and troubleshoot the data. The scans may be conveniently analysed using a spreadsheet. ASCII files may be easily transported between computers and operating systems (e.g. from DOS to UNIX).

3.2.6. On screen display

One of my aims of the observing program has been to make it user-friendly. At the end of each scan the previous scan is plotted on the screen. The scans are plotted in different colours. This enables the user to quickly identify "bad" scans to be repeated at later date. It is useful and interesting for the observer to be able to monitor the progress of the observation in real time.

3.3. Program details

Listings of all the observing programs are provided in appendix A. I have summarised the most important procedure in each unit below.

UNIT IO_Utils (pg. A-16)

Procedure AntOnSource

The antenna status is continuously polled until the antenna is on the source.

Procedure Calibrate

The noise diode calibration is performed.

Function AntField (field, reply string)

The relevant field is picked from the antenna reply string. For example, the beam latitude or longitude is returned.

UNIT MapUtils (pg. A-20)

Procedure SetUp

Sets up the antenna to drive to the bottom corner of the map.

Procedure SetParams

Write the interactive menu on the screen and read in new parameters.

UNIT Screen (pg. A-23)

Procedure PlotScan (I)

Plots scan no i in a new colour on the screen.

UNIT Utils (pg. A-28)

Procedure OutScan

Output scan to the screen and write to a text file

Procedure PutFlHead

Write Julian day, sidereal time, beam offsets etc. to the text file.

Program Map (pg. A-1)

Procedure DoScan

While the antenna is scanning, the voltmeter and antenna status are sampled.

Procedure DoScans

Sets up commands to move the antenna to the start of each scan and to start scanning. Then calls the procedure DoScan and finally plots the scan on the screen and writes it to disk.

3.4. Other programs

3.4.1. Beam

The program BEAM (pg. A-4) makes a map in a field-centered coordinate projection. This is a coordinate system centered on the source. It is used for making maps of the antenna beam or for maps of small sources. The RA and DEC of the source are entered together with the scan length and scan rate. This program follows the same procedure as MAP.

3.4.2. Source

A number of up and down scans are made through a given position in both the longitude and latitude directions. This is useful for checking the offset between the up and down scans caused by the radiometer time constant. SOURCE is listed on pg. A-7.

.

3.4.3. PlotScan

PLOTSCAN is a utility program to plot scans on the observing computer at the observatory. It reads scans off disk and plots them to the screen. The scans can be binned before plotting and a linear baseline can be subtracted. This program is intended to allow the observer to analyse a night's observation and determine which, if any, scans need to be repeated. It gives the observer a preview of the data.

Chapter 4.

Data reduction method

Once two orthogonal maps have been made of an area, I use the basket weaving technique to determine the background of the individual scans in each map. These backgrounds are then subtracted and the maps combined into a final map which is free of significant scanning effects.

In this chapter I first discuss the development of the data reduction technique, highlighting some of the problems I encountered, and then discuss its implementation.

4.1. Development of the data reduction technique

Initially I concentrated on developing the core procedure - the implementation of the basket weaving technique itself. I extensively adapted a program, WEAVE, which was used at the Max Planck Institute for Radio Astronomy (MPIFR). I tested my program by using computer generated orthogonal maps. These model maps contained a gaussian source profile and zero-mean random noise with sinusoidal corrugations running along the scanning direction to simulate the scanning effects. I developed and refined the weave procedure until it was able to remove the known background profiles from these model maps. To monitor this process I displayed images of the intermediate maps after each iteration of the basket weaving procedure.

As a more stringent test, I added "scanning effects", in the form of random second order polynomials, to a map containing real data. WEAVE was able to successfully remove these polynomials in the presence of noise.

I was now ready to test my adaptation of the WEAVE program on real data. I observed orthogonal maps of a 5° by 5° section of the Galactic plane at HartRAO in June 1992. These observations also served as a first test of the new observing program (see Chapter 3). I made two orthogonal maps of the area, binned the data in square bins and tested WEAVE on it. The results looked promising - the final map was relatively free of scanning effects. Unfortunately this first map was lost in a disk crash of the departmental VAX computer.

I now had the core of the data reduction procedure - a working program to determine the coefficients of the background polynomials. Although the scanning effects were greatly reduced, this initial test did not produce an accurate map. I had assumed that the scan trajectories were straight lines which intersected on a regular grid (i.e. that the first latitude scan intersected the first bin of each longitude scan). However, despite starting scanning a degree before the edge of the map, the antenna does not accurately track a constant angle but wanders by as much as one scan spacing from its nominal coordinate. The scans do not therefore fall on a regular square grid and the actual intersection points need to be determined accurately. This proved to be a taxing and time consuming task, the solution to which is described in section 4.2.3.

Another error that was evident in this initial map arises because the post detection time constant in the radiometer leads to the radiometer voltage being delayed relative to the antenna angle. This delay resulted in a hysteresis effect, the removal of which is described in section 4.2.2.

The following section discusses the final version of the data reduction process.

4.2. The data reduction process

The processing of each observation from the original raw scans to the final map involves the following steps:

1. A parameter file is created for each observation
2. The hysteresis effect of the post detection time constant in the radiometer is removed from the raw scan data.
3. The coefficients of the initial linear baseline are found for each scan.
4. Radiometer voltages in mV are converted to antenna temperatures in K using the noise diode calibration.
5. The coordinates of the intersection points of the longitude and latitude scans are determined.
6. Scan temperatures are interpolated at the intersection points.
7. The coefficients of the background polynomials for each scan are determined using the basket weaving technique.
8. The polynomial backgrounds are subtracted from the scans.
9. The scans are interpolated onto a regular grid.
10. A map in FITS format is produced.

These steps are performed by a suite of FORTRAN programs running under the Unix operating system. The process is represented in the flow diagram in figure 4-1.

The program WRITEPARAM creates the parameter file (obs.prm) used by subsequent programs. SCALE takes the original scan files, corrects them for the hysteresis effect and produces a new set of corrected scans in units of antenna temperature. It also produces the files px1.dat and py1.dat containing the coefficients of the linear baselines for each scan. CROSS determines the intersection points of the scans and stores their coordinates in xy.dat. BIN1 interpolates the scans at the intersection points, producing files binx.dat and biny.dat. WEAVE uses the binned scans to find the coefficients of the background polynomials. These coefficients are stored in px.dat and py.dat. FINALBIN subtracts backgrounds from the scans

and stores the resultant map in obs.dat. The corrected latitude and longitude maps are stored in obsx.dat and obsy.dat.

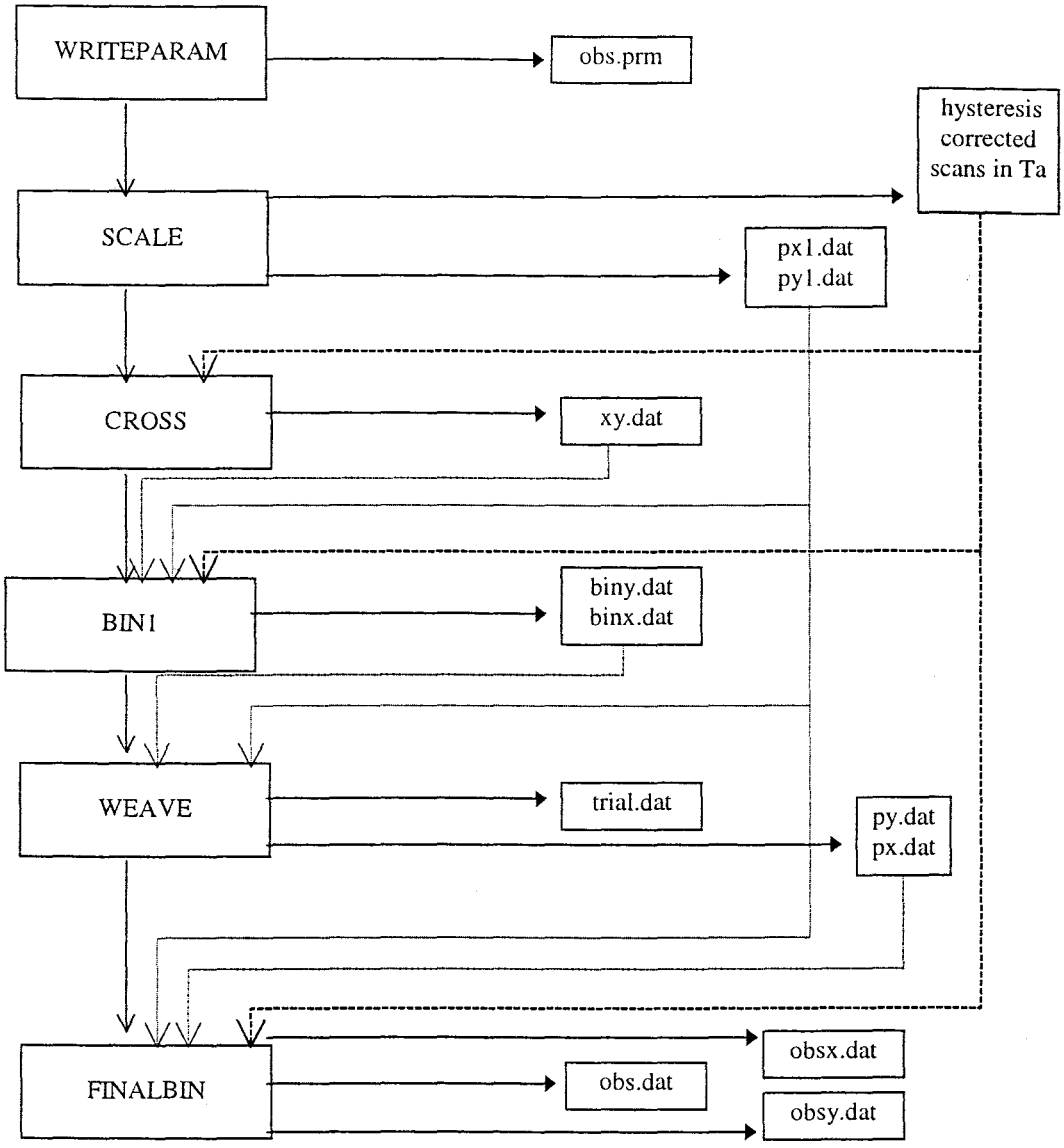


Figure 4-1 Flow diagram of data reduction procedure.

The operation of these program is described in the following sections. Program listings are provided in Appendix B for reference.

4.2.1. Parameter file

Each observation has an associated parameter file which is used by all subsequent analysis programs. This file is created by WRITEPARAM (pg. B-1) and contains the coordinates of the map boundaries together with the location of the various data files needed by the analysis programs.

WRITEPARAM prompts the user for each of the following parameters:

- M - the number of latitude scans
- N - the number of longitude scans
- Latmin - the minimum latitude value
- Longmin - the minimum longitude value
- YSize - the latitude range of the map
- XSize - the longitude range of the map
- Spacing - the spacing between the scans
- XScan - the directory containing the hysteresis-corrected and scaled longitude scans
- YScan - the directory containing the hysteresis-corrected and scaled latitude scans
- ObsName - a character string (length < 8) containing the name of the observation
- DirString - the directory containing all output files

The parameters are stored in the file ObsName.prm and read by each of the subsequent analysis programs.

4.2.2. Hysteresis removal

As mentioned previously, the post detection time constant in the radiometer causes the radiometer reading to be delayed by some time relative to the antenna angle. The removal of this hysteresis effect is done by the program SCALE (pg. B-2).

Each scan is stored in a separate text file (section 3.2.5), one sample per line. Each sample consists of a radiometer voltage reading and a coordinate pair (latitude and longitude). Antenna coordinates recorded at time t, correspond to antenna temperatures $T_a(t-\Delta t)$. To remove this delay, which is compounded into a hysteresis effect as the antenna scans in both directions, the antenna position must be shifted to an earlier time so that the antenna angle and radiometer voltage are synchronous.

To make this correction I need to determine the exact value of the time constant, and to find the scan angle as a function of time. Using SOURCE (section 3.4.2) I made two declination scans, at constant right ascension, in opposite directions through the strong source Virgo A, shown in figure 4-2.

As the radiometer voltage reading is delayed, the peak of the source occurs at a later time. The difference in declination between the peaks of the northwards and southwards scans is 20 mdeg (each peak has been shifted by 10 mdeg). As the scan was made at a scan rate of 0.1 deg/s, a 10 mdeg shift corresponds to a post detection time constant of 0.1 s, which is consistent with the value calculated for the post detection filter.

Now that I had found the post detection time constant, the next step was to delay the scan coordinate (latitude or longitude) by this time constant. In order to do this it was necessary to fit local polynomials to the scan angle/time points. In fact the angle is not directly a function of time but of sample number. However, as the sampling takes place at a regular rate of 9.1 samples/s, this is irrelevant. A time constant of $\Delta t = 0.1$ s corresponds to a delay of $\Delta n = 0.099$ sample points.

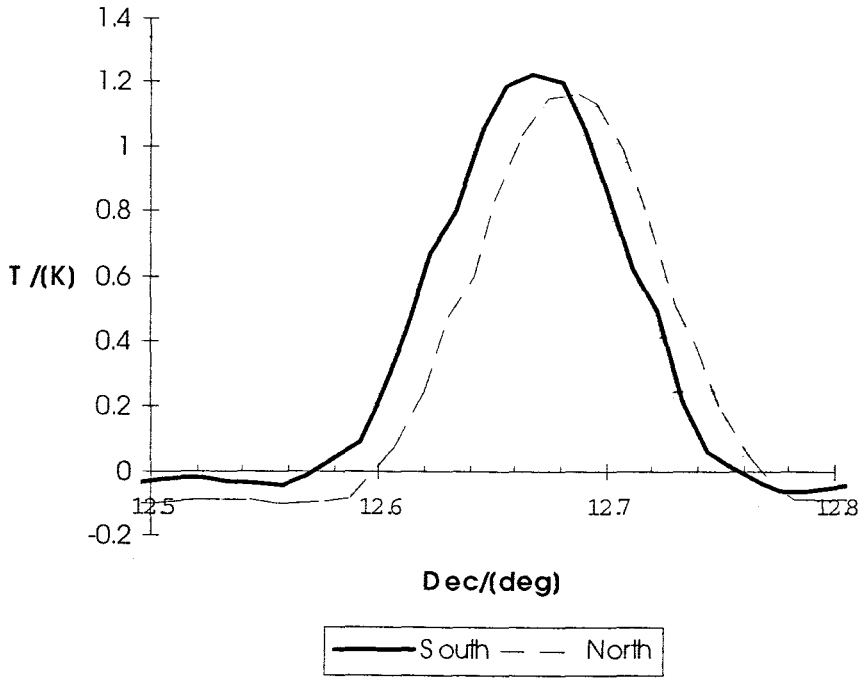


Figure 4-2 Scans through Virgo A. The peak of the northwards scan (dashed line) is shifted to a later time (a higher declination). For the southwards scan (solid line) the peak is shifted south. The difference, in declination, between the peaks is 20 mdeg.

I fitted a local quadratic polynomial to small segments of the scan. The scan angle, ϕ, as a function of sample number, n, is given by

$$\phi(n) = A_0 + A_1 n + A_2 n^2 \qquad (4\text{-}1)$$

I then delayed this function by the time constant (i.e. replaced $\phi(n)$ by $\phi(n - \Delta n)$ where $\Delta n = 0.099$.)

In addition, SCALE determined the coefficients of a linear baseline of each scan by fitting a straight line to three points at each end. The coefficients for each scan were written to a text file (px1.dat or py1.dat). SCALE also uses the noise diode calibration factor stored in the file header to convert from radiometer output voltage to antenna temperature in Kelvin.

SCALE must be run for both latitude and longitude maps. It is possible to select a subset of the map for further processing. At the end of this routine a set of hysteresis corrected scans is written out, one file for each scan.

4.2.3. Determination of intersection points

The basket-weaving method relies on the two sets of scans agreeing at their intersection points. Ideally the scan tracks should form a square grid of orthogonal straight lines. However, in reality the antenna does not track this pattern perfectly. For example, when scanning in latitude at "constant" longitude, the longitude value will wobble slightly around its nominal value. Tracking is worse when the antenna is scanning in Galactic latitude and longitude as both antenna axes must move simultaneously. More seriously the antenna takes some time to settle down at the start of each scan and the "constant" value can deviate by up to three scan spacings from its nominal value. Thus the scans do not intersect on a regular grid, so it is necessary to calculate the arbitrary coordinates of the intersection points, as it is at these points that the two maps must agree.

Figure 4-3 shows the scan tracks, in the longitude-latitude plane, of a set of longitude and latitude scans. Two latitude scans in opposite directions are labeled. For the northwards scan the antenna starts at -2.5° latitude and scans up to 2.5°. The antenna takes about 0.4° from the start of the scan to settle down to tracking an almost constant longitude value.

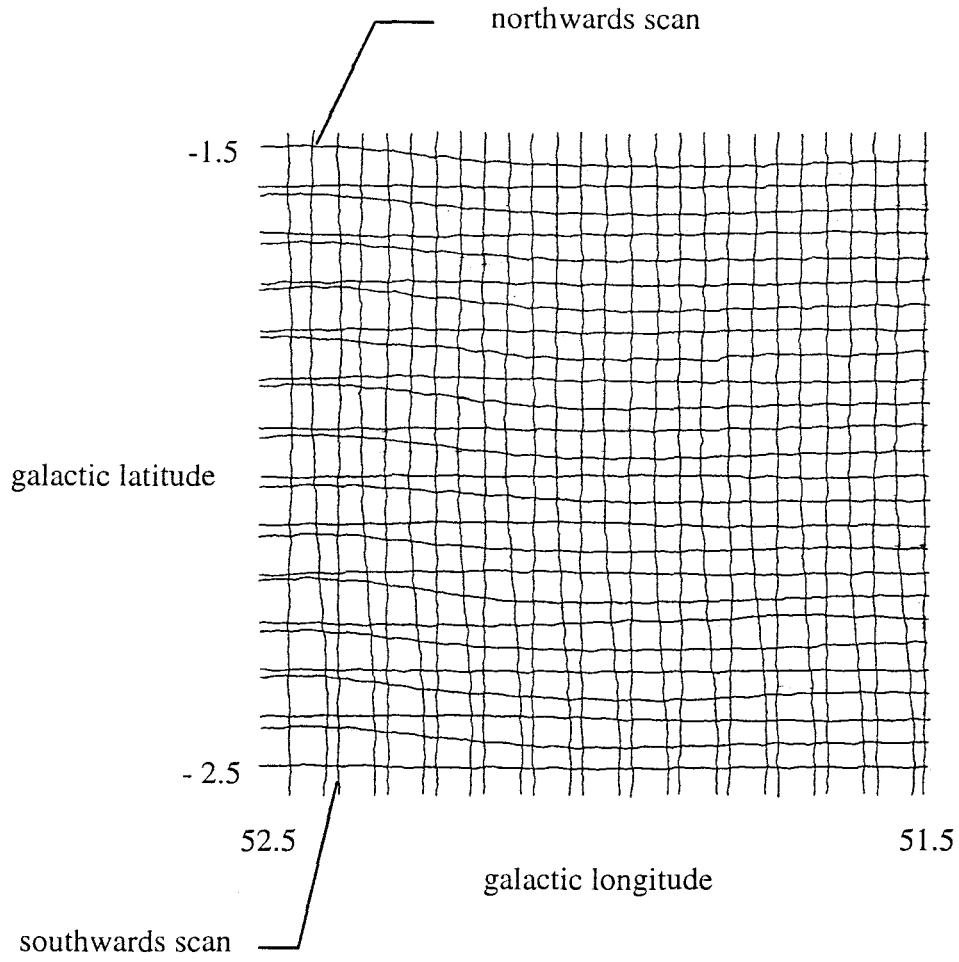


Figure 4-3 The scan tracks for a set of longitude and latitude scans are shown. The map covers the region $47.5° \leq l \leq 52.5°$, $b \leq \pm 2.5°$ but only the bottom left corner is shown.

These intersection points were determined by CROSS (pg. B-5) using the following algorithm.

read all longitude scans into a large array

For I = 1 to M (for each latitude scan)

read scan into array

nominal longitude is $l_i = (I-1) \times \text{binwidth} + \text{Longmin}$

For J = 1 to N (for each longitude scan)

nominal latitude is $b_j = (j-1) \times \text{binwidth} + \text{Latmin}$

find ALong, average longitude for latitude points where $|b-b_j| \leq 6 \times \text{binwidth}$

find ALat, average latitude for longitude points where $|l-l_i| \leq 6 \times \text{binwidth}$

fit LS straight line to all points on latitude scan where |b-ALat | ≤ binwidth

fit LS straight line to all points on longitude scan where |l-ALong | ≤ binwidth

find the intersection point of these two lines

end

end

To illustrate the procedure I will use as an example the determination of the intersection point of scans I and J shown in figure 4-4. Scan I was made by scanning in latitude from -2.5° to 2.5° at a nominal longitude of 47.5° while scan J was made by scanning in longitude from 47.5° to 52.5° at a nominal latitude of -2.5°.

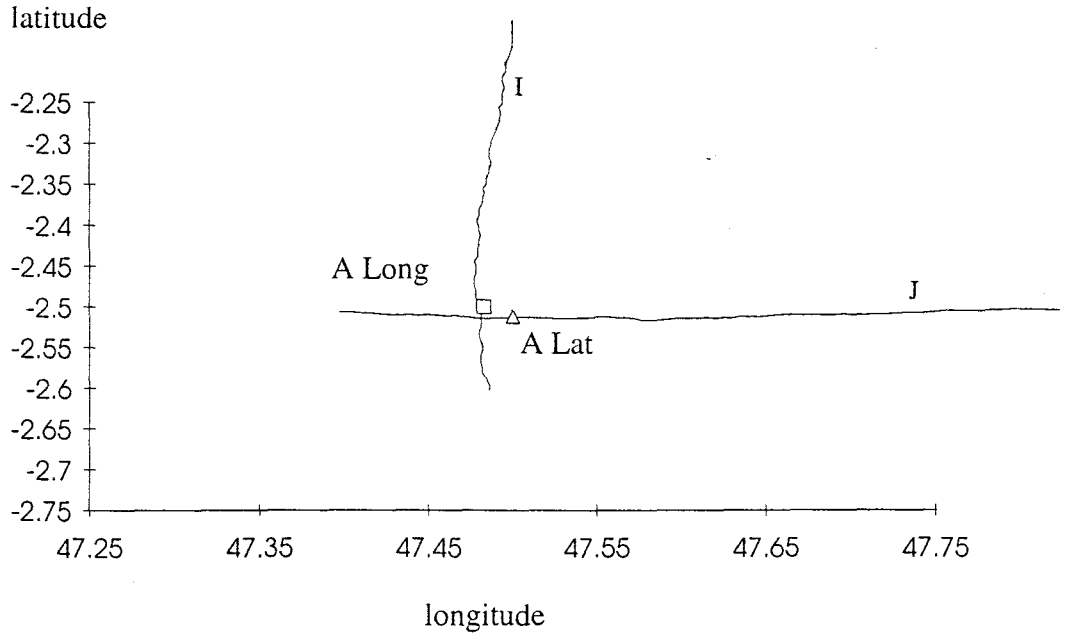


Figure 4-4 Intersecting latitude and longitude scans

The average longitude, ALong, of scan I is found to be 47.48. Then the average latitude of points on scan J that are within 6 binwidths of 47.48 is found. This is ALat = -2.51. Points lying within one binwidth of ALong and ALat are selected from the longitude and latitude scans. A least squares straight line is fitted to each of these sets of points and the coordinates of the intersection of the two lines found.

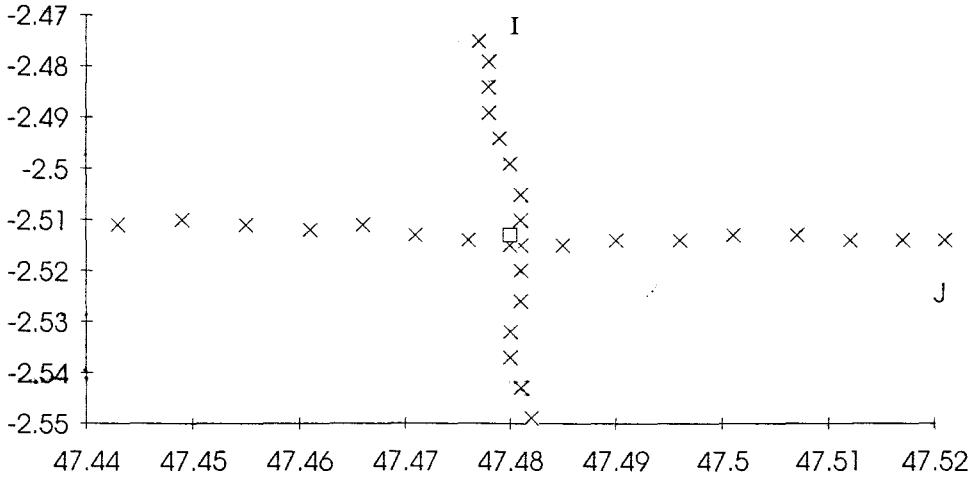


Figure 4-5 Points from scan I within one binwidth of ALat and from scan J within one binwidth of ALong are shown. Least squares straight lines are fitted and the intersection point determined (shown as an open square).

Figure 4-6 confirms that CROSS was able to accurately determine the coordinates of the intersection points.

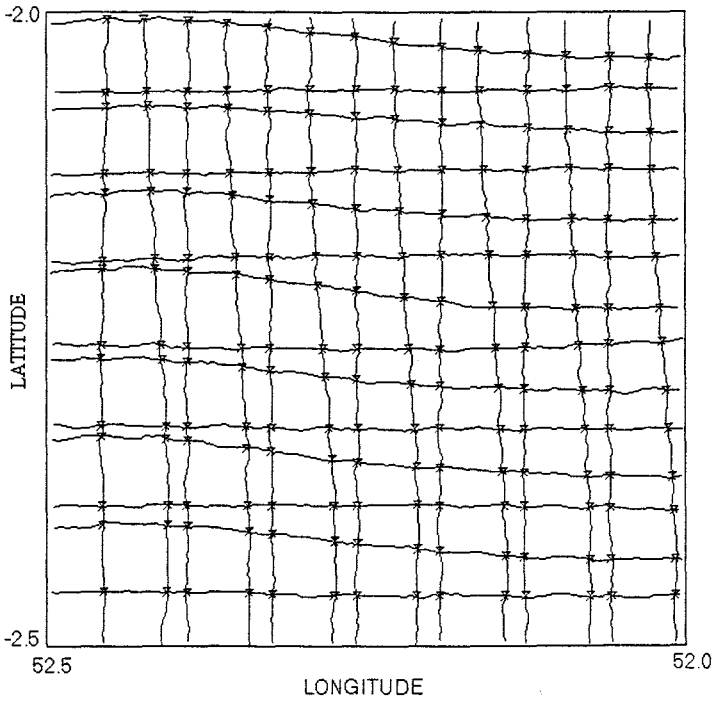


Figure 4-6 Scan tracks are shown for a small section of the map. The intersection points determined by CROSS are superimposed as crosses.

4.2.4. Initial binning

At this stage the intersection points of the latitude and longitude scans have been found. Now I need a representative antenna temperature value from the latitude and longitude maps for each intersection point. This is done by BIN1 (pg. B-7) which interpolates each scan using square bins, of width equal to the scan spacing, centred on the intersection points. A linear baseline that was determined by SCALE is subtracted from each scan at this stage.

BIN1 must be run separately for the latitude and longitude scans. It reduces each set of scans to a matrix of values which are written away to the files binx.dat and biny.dat. Each bin is written away in the form (I, J, longitude, latitude, Ta), one bin per line.

4.2.5. Basket weaving

The next stage is the most important - to find the coefficients of the background polynomials which describe the scanning effects using the basket weave technique. These scanning effects can be treated as being a function of the scanning coordinate only.

If the underlying "real" structure of the area is described by a function F, then F would be common to both maps. Equations for the maps can be written as

$$\begin{aligned} XMap(i,j) &= F(i,j) + Px(i,j) + Nx(i,j) \\ YMap(i,j) &= F(i,j) + Py(i,j) + Ny(i,j) \end{aligned} \qquad (4\text{-}2)$$

Where F(i,j) is the underlying "real" structure at matrix point (i,j)

Px(i,j) is the background of scan J from XMap at (i,j)

Py(i,j) is the background of scan I from YMap at (i,j)

Nx(i,j) is the noise contribution to XMap at (i,j)

Ny(i,j) is the noise contribution to YMap at (i,j)

Each map consists of the real structure, plus a background that can be assumed to be a function of the scanning coordinate only, and noise. The noise Nx and Ny is random and zero-mean. The aim of the basket-weaving procedure is to remove the backgrounds P_x and P_y, leaving only the real structure (and noise).

To find the coefficients of the polynomials I initially find a difference map by subtracting the x-map and y-map from each other. The difference map is given by (using equation 4-2)

$$\begin{aligned} \mathrm{Diff}(i,j) &= \mathrm{XMap}(i,j) - \mathrm{YMap}(i,j) \\ &= \mathrm{Px}(i,j) - \mathrm{Py}(i,j) + \mathrm{Noise} \end{aligned} \tag{4-3}$$

The difference map now is the sum of Px and -Py, which are functions of x and y only respectively, and a noise component.

I use the IMSL routines rlfoth and rldopm to fit least squares polynomials Px'(i,j) and Py'(i,j) to the difference map in the x and y directions. These polynomials are multiplied by a damping term, S, and then subtracted from the original maps to form a new difference map. S is similar to the damping term in a servo equation (Haslam, 1974). I have found a value of 0.7 for S to be most effective.

The new difference map is given by

$$\mathrm{Diff}(i,j) = [\mathrm{XMap}(i,j) - \mathrm{Px}'(i,j)] - [\mathrm{YMap}(i,j) - \mathrm{Py}'(i,j)] \tag{4-4}$$

This process is repeated until the average value of points on the difference map is less than one tenth of the theoretical rms noise. For the 3.5 cm system the theoretical rms noise is $\approx$ 1 mK and the limit is set to 1E-4.

The coefficients of the polynomials are then stored away in files for subtraction in the final binning process.

The algorithm for WEAVE is given below.

repeat

Diff(I,J) = XMap(I,J) - YMap(I,J)

For each longitude scan J

fit polynomial to row J of difference map

subtract polynomial × S from longitude scan

end

For each latitude scan I

fit polynomial to column I of difference map

subtract polynomial × S from latitude scan

end

find average of all points on difference map

until (average is less than limit)

This process can be best illustrated by considering the background removal of one scan. Figure 4-7 shows a latitude scan together with corresponding values from intersecting longitude scans. The longitude points do not lie on the latitude scan because of the different backgrounds. The aim of the weaving process is to "weave" the scan and points together by iteratively subtracting backgrounds from the latitude and longitude scans.

A quick glance at figure 4-7 suggests that the latitude scan needs to be raised between latitudes of -2.5° and -1° and lowered between latitudes of 1° and 2.5°.

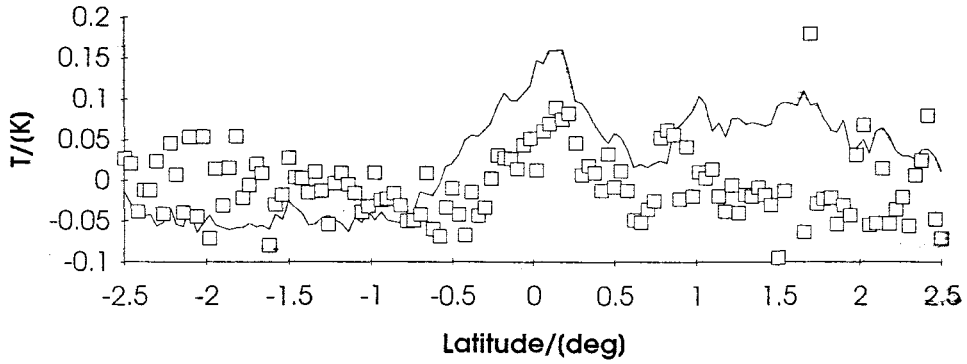


Figure 4-7 The solid line shows a latitude scan. The corresponding values from intersecting longitude scans are shown by open squares.

The background polynomial determined by weave and the final scan are shown in figure 4-8. The latitude and longitude scans should now agree at the intersection points to within the noise.

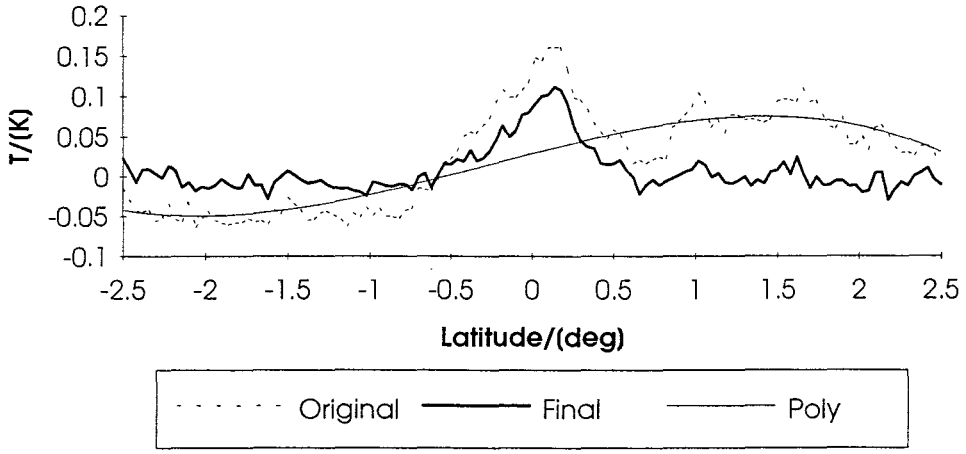


Figure 4-8 The original scan is shown by the dotted line, the final scan by the solid line. The polynomial is not fitted to the scan itself but to the residuals between the orthogonal scans.

WEAVE prompts the user for the maximum degree polynomial to be fitted and for a limit value for the difference map.

The polynomials are subtracted from the scans of each map. The two maps are then averaged to produce a rough map binned at the intersection points, which is written to the file trial.dat.

4.2.6. Exclude scans

Some scans contain interference or other artifacts which cannot be removed by basket weaving. These scans can be seen as striations in the trial map. The program PLOTSCAN plots out background corrected scans to further help in identifying "bad" scans. Figure 4-9 shows the output from PLOTSCAN for a set of scans. Scans 79, 81 and 84 are examples of "bad" scans. These scans can be excluded from the final binning but are used in the basket weaving process.

4.2.7. Final binning

Once the background polynomials have been determined and subtracted, the data from the two maps must be combined and interpolated onto a regular grid. This is done by the program FINALBIN (pg. B-13).

Observing the sky using a scanning antenna is equivalent to convolving the brightness distribution of the sky with the normalised beam pattern of the antenna. The resultant map can be written as

$$Map(i, j) = sky(i, j) * P_n(i, j) \qquad (4\text{-}5)$$

(Kraus, 1966 section 3-4).

Convolution in the spatial domain corresponds to multiplication in the Fourier domain. The normalised beam pattern is proportional to the Fourier transform of auto-correlation of the aperture function A(i,j) (Bracewell, 1986 pg. 281). Hence in the frequency domain equation 4-4 becomes

$$MAP(u, v) = SKY(u, v) \times ACF\ A(i, j) \qquad (4\text{-}6)$$

where MAP and SKY are Fourier transforms of Map and Sky.

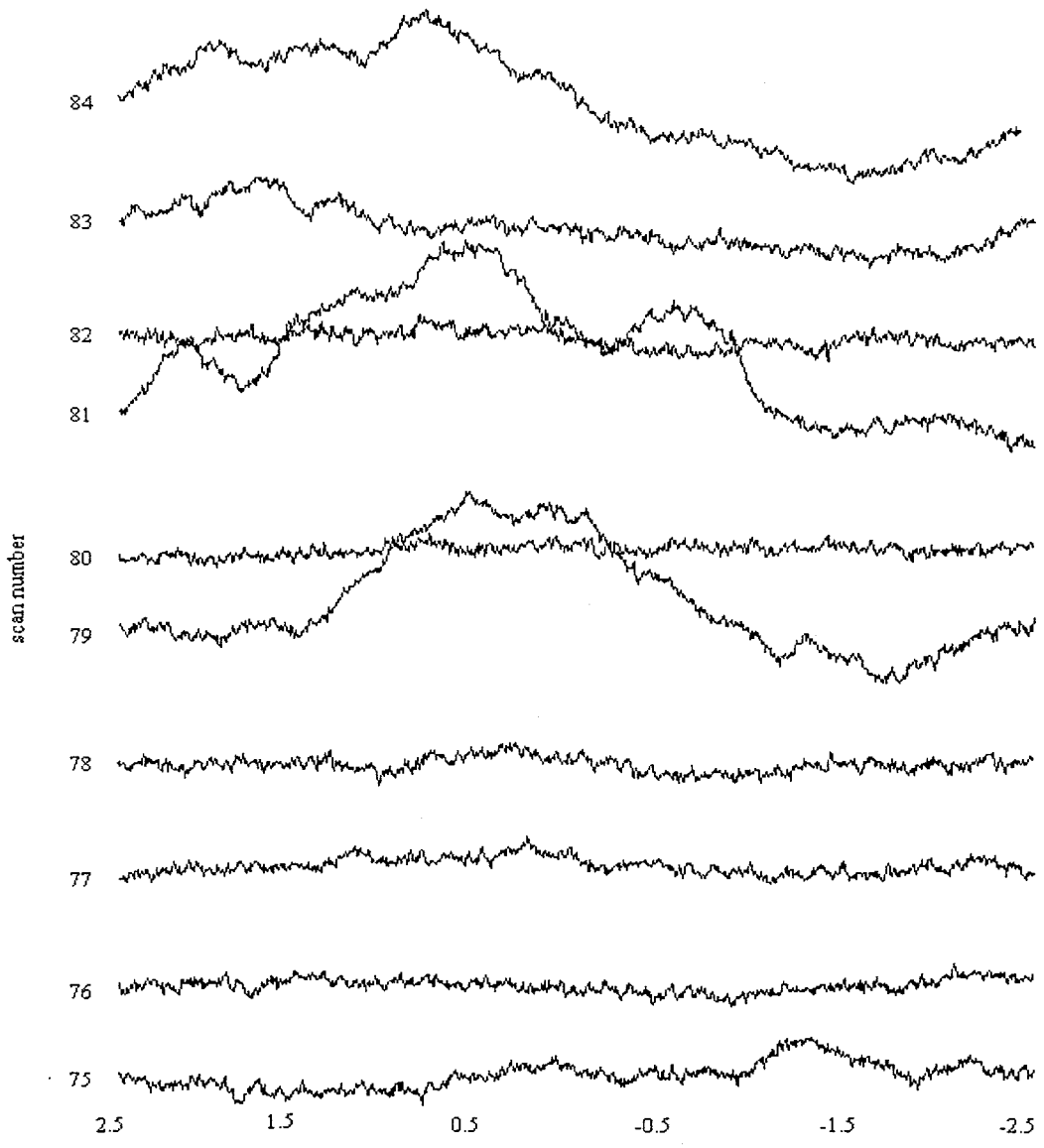


Figure 4-9 Output from PlotScan. Scans 79, 81 and 84 would be excluded

The HartRAO antenna has a diameter of 26 m. When operating at 3.5 cm the aperture is a circular function of diameter 740 λ in the spatial frequency domain. The auto-correlation function of this aperture function has a maximum radius of 740 λ. The highest frequency in the map is 740 cycles/radian or 13 cycles/degree. In the final binning procedure I want to remove all frequencies greater than this highest spatial frequency. This involves multiplying, in the Fourier domain, by a function which is unity over the area of ACF (A). In the spatial domain this is equivalent to convolving the data with the circularly-symmetric function whose

radial profile is shown in figure 4-4. The function is tapered to prevent ringing arising from the convolution.

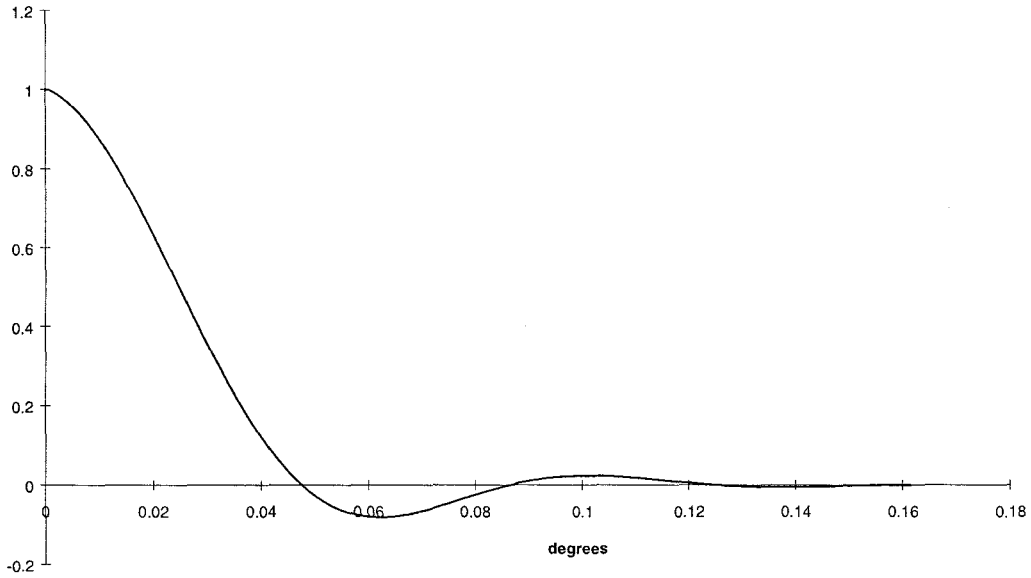


Figure 4-10 Radial profile of the interpolation kernel $\frac{J_1(2\pi\times13\times r)}{\pi\times13\times r}\times\left(1-\frac{2\pi\times13\times r}{13.32}\right)$ where r is in degrees.

FINALBIN subtracts the background profiles from each scan and then convolves the data with the interpolation kernel function shown above. At the start of the program the user is given the opportunity exclude bad scans or to read in previously excluded scans. Excluded scans are stored in the files excludx and excludy.

The final map is written to the file obs.dat.

4.2.8. FITS file conversion

The data can be converted into FITS format (Flexible Image Transport System, Wells et al, 1981) format using the program TOFITS (pg. B-17). This file can then be transported and displayed. After the final binning process there may be empty bins in the map as a result of excluded scans. These empty bins are replaced by an approximate value which is the average of the surrounding points.

Plate 1 (overleaf): shows the effect of the basket weaving procedure on a map (region B). The top and middle images show the raw latitude and longitude maps displayed individually . The maps have been binned at the crossing points. Striations in the scanning direction can be seen. The bottom map is the final processed map, which shows considerable improvement.

4.2.9. Other utility programs

FITS maps can be displayed as false colour images using the popular utility saoimage. This is useful for checking images and identifying bad scans. All images and contour plots were produced using the package wip.

I have written some utility programs for diagnostic purposes. Selected scans can be plotted using the program PLOTSCAN. Figure 4-9 is an example of output from PLOTSCAN. This is useful for identifying bad scans.

Individual scans, at various stages of processing, can be extracted from the data or final maps using GETSCAN. This was useful in tracking the effect of the processing on individual scans.

TRACK plots scan tracks on a latitude-longitude grid. PLOTX plots the intersection points determined by CROSS.

4.3. Conclusion

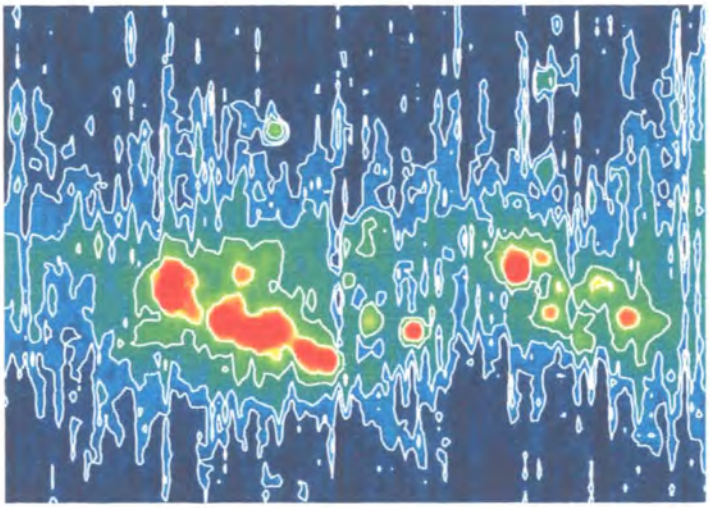
I had now implemented a basket weaving procedure which removed background polynomials from the raw scans and produced a map in FITS format.

The success of the procedure can be seen in Plate 1. Basket weaving has removed all significant scanning effects.

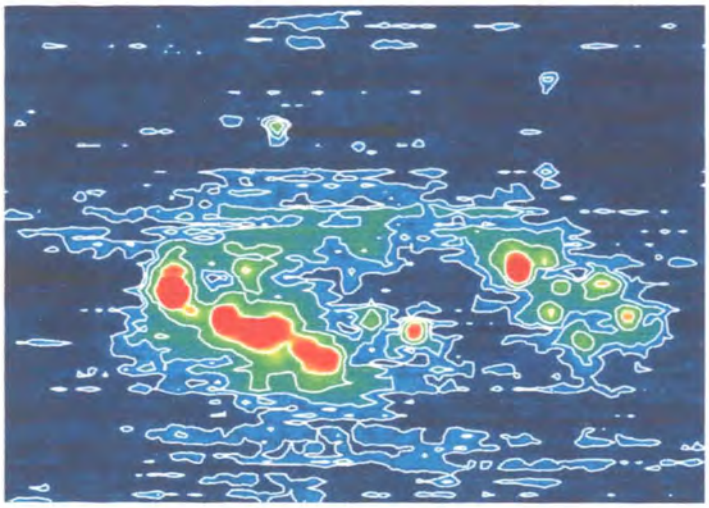
Plate 1

Removal of Scanning Effects

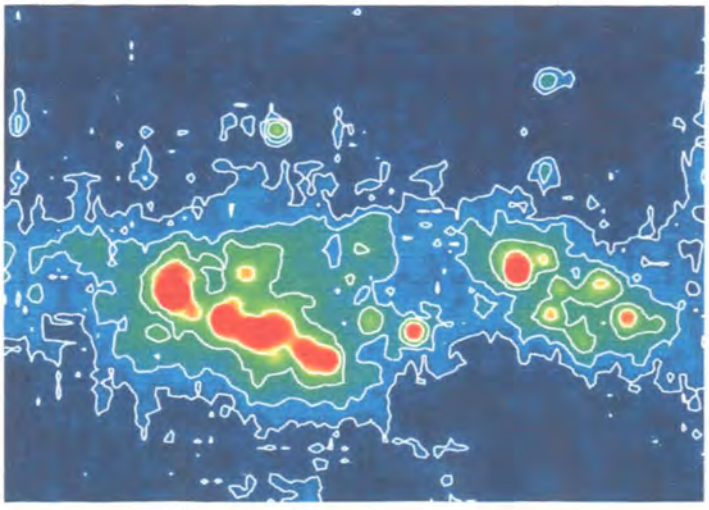
Latitude scans



Longitude Scans



Final Map



It is customary to convert maps from antenna to brightness temperatures to allow for comparison with other maps. This calibration is discussed in the next chapter.

Chapter 5.

Calibration

At this stage the map is in units of antenna temperature which are telescope dependent. To allow comparison between radio maps it is customary to convert from antenna temperature to full beam brightness temperature. This conversion incorporates corrections for ohmic losses, errors in the calibration of the noise diode and losses due to the stray antenna response. The flux of a source can then be found directly by numerical integration.

In this chapter I describe the conversion of antenna temperatures to full beam temperatures for the 3.5 cm and 6 cm systems. I used two methods to determine the conversion factor. Firstly, I used point source calibrators of known fluxes and then as a check I used the moon, a nearly blackbody radiator which fills the full beam at 3.6 and 6 cm. I outline the theory behind the two methods, and then discuss their implementation and final results.

The table below contains pertinent parameters of the antenna at 3.5 cm, some of which are derived in this chapter.

Parameter	Symbol	Value	Uncertainty
Centre frequency		8500 MHz	
Bandwidth	B	400 MHz	
System temperature	T_{sys}	65 K	± 1 K
Point source sensitivity	P.S.S	17.47 Jy/K	± 1.7 Jy/K
Physical aperture	A_p	531 m^2	
Effective aperture	A_e	158 m^2	± 16 m^2
Aperture efficiency	$\varepsilon_{ap}\eta_R$	30 %	± 3 %
Noise diode temp	T_N	10 K	± 1 K
Full beam brightness conversion factor	T_{FB}/T_A	2.2	± 0.1

Table 5-1 Parameters of the HartRAO antenna

5.1. T_A to T_{FB} conversion

The measured antenna temperature that results from a sky brightness temperature distribution $T_b(\theta,\phi)$ is given by (Kraus, 1966 pg. 67)

$$\begin{aligned} T_a(\theta_n,\phi_n) &= \frac{\eta_R}{\Omega_a}\iint_{source} T_b(\theta_n,\phi_n)P_n(\theta-\theta_n,\phi-\phi_n)d\Omega \quad [K] \\ &= \frac{A_e}{\lambda^2}\iint_{source} T_b(\theta_n,\phi_n)P_n(\theta-\theta_n,\phi-\phi_n)d\Omega \quad [K] \end{aligned} \tag{5-1}$$

where $A_e = \frac{\eta_R\lambda^2}{\Omega_a}$ is the effective aperture, η_R is an loss factor to account for power lost in the feed horn and antenna because of dissipative processes and Ω_A is the beam solid angle.

The relationship between T_a and T_b depends on the relative sizes of the source and beam solid angles. For **point sources**, where $\Omega s << \Omega m$ (where Ωm is the main beam solid angle), $P_n(\theta,\phi)$ can be considered to be unity over the angular extent of the source. Equation 5-1 then simplifies to

$$T_a = \frac{\eta_R}{\Omega_a}\iint_{source} T_b(\theta,\phi)d\Omega = \frac{\eta_R\lambda^2}{2k\Omega_a}S_\upsilon \quad [K] \tag{5-2}$$

The ratio of the flux to the antenna temperature for point sources (in units of $Jy.K^{-1}$) is called the point source sensitivity (PSS) and is given by

$$PSS = \frac{S_\upsilon}{T_a} = \frac{2k\Omega_a}{\eta_R\lambda^2}\times 10^{26} Jy.K^{-1} = 2kA_e\times 10^{26} \quad [Jy.K^{-1}] \tag{5-3}$$

In order to find the intrinsic brightness temperature for extended sources it is necessary to invert equation 5-1 which is not practical. Instead the full beam solid angle, Ω_{fb}, a loosely defined quantity is used to define a compromise temperature scale. The full beam contains most of the structure of the beam and Ω_{fb} is found by integrating out to the first diffraction sidelobe of the beam. Further integration would not contribute significantly to the beam solid angle.

Integrating equation 5-1 over the full beam solid angle gives

$$S' = \frac{2k}{\lambda^2}\iint_{\Omega_{fb}} T_a(\theta,\phi)d\Omega = \frac{\eta_R}{\Omega_a}\iint_{\Omega_{fb}} B(\theta,\phi) = \iint_{\Omega_{fb}} P_n(\theta,\phi)d\Omega$$
$$= \frac{\eta_R\Omega_{fb}}{\Omega_a}S_\upsilon = \frac{\eta_R}{\varepsilon_{fb}}S_\upsilon \quad [W.m^{-2}.Hz^{-1}] \tag{5-4}$$

where ε_{fb} is the full beam efficiency.

The flux of the source, the integral over the source of the brightness distribution, is then given by

$$S_\upsilon = S'\frac{\Omega_a}{\eta_R\Omega_{fb}} = S'\frac{1}{\eta_R\Omega_{fb}} \quad [W.m^{-2}.Hz^{-1}] \tag{5-5}$$

It is useful to define a full beam brightness temperature as we defined a brightness temperature

$$T_{fb} = \frac{T_a}{\eta_R\varepsilon_{fb}} \quad [K] \tag{5-6}$$

T_{fb} is a compromise approximation to T_b, which is exact for sources that fill the full beam. The flux of a source can then be found directly by numerical integration and is then given by

$$S_\upsilon = \frac{2k}{\lambda^2}\iint_{\Omega_{fb}} T_{fb}(\theta,\phi)d\Omega \quad [W.m^{-2}.Hz^{-1}] \tag{5-7}$$

In the next two sections I discuss the two methods I used to determine $1/\eta_R\varepsilon_{fb}$.

5.1.1. Method 1: Using a point source calibrator

I chose a strong point source with a well-determined flux and made a beam map using this source. The flux of the source is given by equation 5-7 which can be re-written as

$$S_\upsilon = \frac{1}{\eta_R\varepsilon_{fb}}\left[\frac{2k}{\lambda^2}\iint_{\Omega_{fb}} T_{fb}(\theta,\phi)d\Omega\right] = \frac{1}{\eta_R\varepsilon_{fb}}S' \quad [W.m^{-2}.Hz^{-1}] \tag{5-8}$$

The conversion factor, $1/\eta_R\varepsilon_{fb}$, can thus be determined by finding the ratio of S_v, the flux of the source to S', the underestimated flux obtained by integrating the antenna temperature over the full beam solid angle.

5.1.2. Method 2: Using the moon

The moon has an angular diameter of 0.5°. At 3.5 cm the moon fills the full beam while at 6 cm this can be assumed as an approximation. The moon is a blackbody and the temperature can be assumed to be roughly constant across the surface of the moon.

Using equation 5-2 and assuming the moon's temperature to constant across the disk gives the antenna temperature as

$$\begin{aligned} T_a &= \frac{\eta_R}{\Omega_a} T_M \iint_{moon} P_n(\theta,\phi)d\Omega \\ &= \frac{\eta_R}{\Omega_a} T_M \Omega_{fb} = T_M \eta_R \varepsilon_{fb} \quad [K] \end{aligned} \tag{5-9}$$

$1/\eta_R\varepsilon_{fb}$ is therefore given by the ratio of the average temperature of the moon to the measured antenna temperature.

5.2. Results

5.2.1. Method 1

Strong point sources (Virgo A at 6 cm and Cygnus A at 3.5 cm) were mapped to obtain beam patterns using the program BEAM (see section 3.4.1). These maps were made using an orthographic coordinate projection centred on the source position. I processed each map using the basket weaving technique and imported the processed data as a grid of evenly spaced antenna temperatures into a spreadsheet for further manipulation.

From equation 5-8 $1/\eta_R\varepsilon_{fb}$ is given by the ratio of the flux, S, to S' where

$$S' = \frac{2k}{\lambda^2}\iint_{\Omega_{fb}} T_a(\theta,\phi)d\Omega \quad [W.m^{-2}.Hz^{-1}] \qquad (5\text{-}10)$$

This integral can be found numerically by summing the antenna temperatures over the full beam and multiplying this by the solid angle subtended by each pixel. S' is then given by

$$S' = \frac{2k}{\lambda^2}(\Delta x)(\Delta y)\sum T_a \quad [W.m^{-2}.Hz^{-1}] \qquad (5\text{-}11)$$

where Δx and Δy are the pixel widths in radians.

At 8500 MHz I made a 1°×1° map centred on Cygnus A using a scan spacing of 0.040°. I found S' to be 74.1 Jy. Using the parameters given by Ott et al. (1994) the flux at 8500 MHz is 165.3 Jy. This gives a value for $1/\eta_R\varepsilon_{fb}$ of 2.23.

At 5000 MHz I made a 1.3°×1.3° map centred on Virgo A with a scan spacing of 0.050°. Virgo A has a flux of 74.2 Jy at 5000 MHz (Ott et al., 1994). I measured S' = 45.2 Jy giving a value for $1/\eta_R\varepsilon_{fb}$ of 1.64.

5.2.2. Method 2

The moon was mapped using a program written by Jonas. Keihm & Langseth (1975) give the disk average brightness temperature of the moon as 215 ± 10 K.

The peak temperature of the moon at 8500 MHz is 97 K so $1/\eta_R\varepsilon_{fb}$ = 2.2.

At 5000 MHz the peak temperature is 133 K giving a value for $1/\eta_R\varepsilon_{fb}$ of 1.6.

These factors agree with the factors determined by method 1.

5.2.3. Beam maps

As Cygnus A and Virgo A are point sources the maps provide beam maps of the antenna beam patterns at the two frequencies. Figures 5-1 and 5-2 show the normalised antenna pattern at 8500 MHz and 5000 MHz.

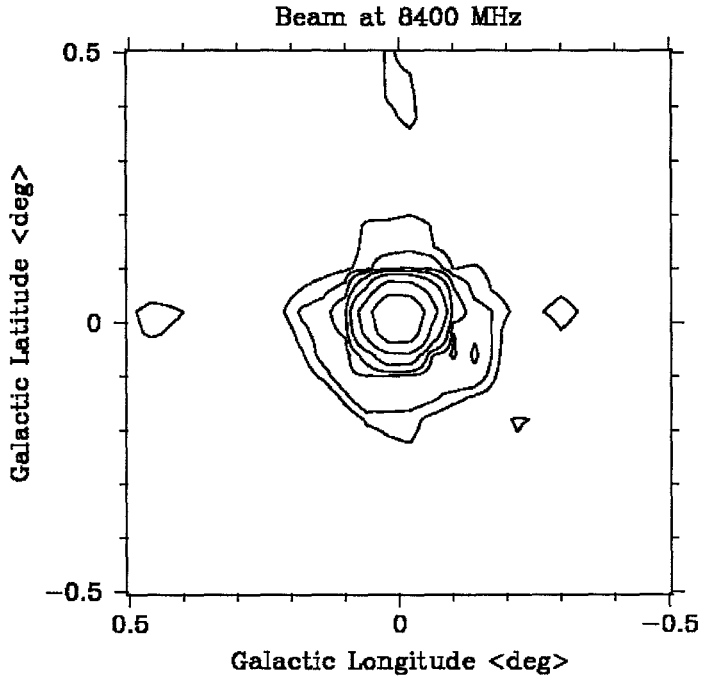


Figure 5-1 Beam map at 8500 MHz. The contours are drawn at 3, 6, 9, 12, 15, 18, 21 dB below the peak.

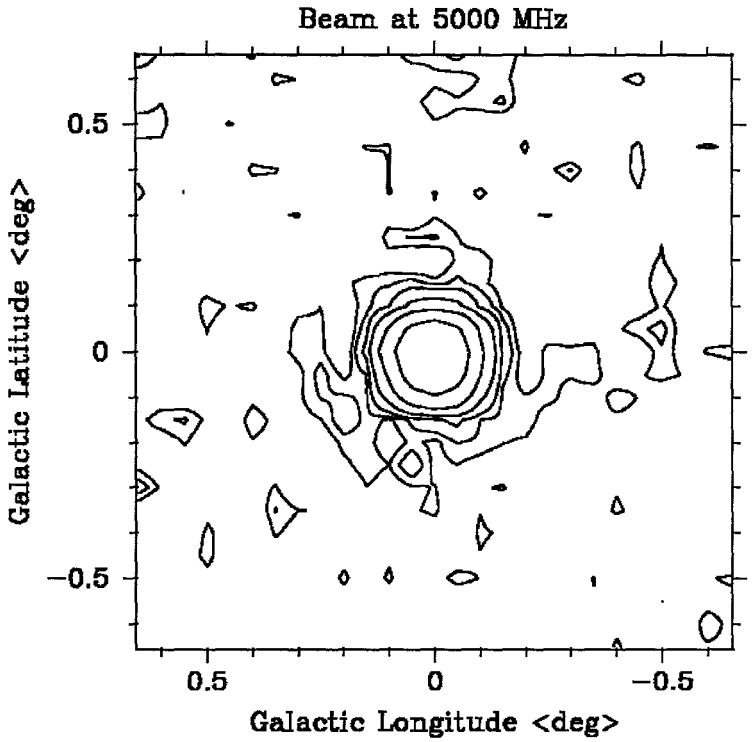


Figure 5-2 Beam map at 5000 MHz. The contours are drawn at 3, 6, 9, 12, 15, 18 dB below the peak.

5.3. Conclusion

I have found the factors to convert from T_a to T_{fb} using two independent methods. Both methods give the same results. At 8500 MHz T_{fb}/T_a is 2.2 and at 5000 MHz T_{fb}/T_a is 1.6.

The effective aperture, A_e, of the antenna when operating at 3.5 cm can be found from equation 5-3 using the PSS (= 17.47 Jy/K). A_e is 138m^2 which gives a value of 30% for the aperture efficiency.

Chapter 6.

The radio spectral index of MSH 15-52

6.1. Introduction

MSH 15-52 (or G320.4-1.2) is an extended composite supernova remnant associated with the optical nebula RCW 89 and containing the x-ray and radio pulsar PSR 1509-58. Pulsar-driven (or Crab-like) supernovae, such as MSH 15-52, are sources of relativistic electrons which are observed in x-rays. These electrons can inverse Compton scatter background photons to higher energies (TeV γ-rays). Du Plessis and De Jager from the Space Research Unit at Potchefstroom University were investigating mechanisms for inverse Compton scattering in supernova MSH 15-52 and wished to establish the spectral index of the remnant. To determine the radio spectral index I observed MSH 15-52 at 5000 MHz and 8500 MHz using the HartRAO telescope. By combining flux measurements at these frequencies with data from the existing Rhodes/HartRAO 2326 MHz survey I determined the radio spectral index. In addition to the scientific value of the observations, processing the data provided a test for the observing and analysis programs I had developed.

In this chapter I begin with a review of the literature discussing the structure of the remnant. I then discuss the observations made with the HartRAO telescope and the subsequent determination of the radio spectral index. Finally I discuss the conclusions drawn by du Plessis et al (1995).

6.2. MSH 15-52

Radio observations of MSH 15-52 (e.g. Caswell et al, 1981) show an extended circular shell of diameter 30′ with two bright arcs lying on the rim in the NW and SE. The brighter NW arc contains the radio source Kes 23 and corresponds to the optical Hα nebulae RCW 89. Initially these two arcs were considered as separate supernovae by Milne (1970). However later radio observations, such as the 1415 MHz map of Caswell, Milne and Wellington (1981), suggest that the two structures are part of the same supernova remnant. Recently Milne, Caswell and Haynes (1993) observed the remnant at 4.8 GHz and 8.4 GHz and found a 40′ area of faint non-thermal emission to the SE of the remnant. They suggest this could be an extension of MSH 15-52 or an overlapping remnant (G320.6-1.6). The distance to the remnant, based on HI absorption measurements, is 4.2 kPc (Caswell et al, 1975).

X-ray observations of MSH 15-52 (Seward et al, 1983) show two bright sources - a northern source, consisting of two bright spots, which corresponds to RCW 89 and a southern pulsating x-ray source lying between the two radio arcs. The pulsar, PSR B 1509-58, has a period of 0.15 s and has been observed at X-ray (Seward and Harnden, 1982) and radio frequencies (Manchester et al, 1982). It has a characteristic age of ≈1600 years making it the second youngest pulsar known (second to the Crab pulsar). Thorsett (1992) proposed that the progenitor was the "guest star" recorded by Chinese astronomers in AD 185. This is disputed by Chin and Huang (1994) who identify the event of AD 185 with a comet. Schaeffer (1995) claims that the guest star observation was the concatenation of a nova and the comet P/Swift-Tuttle.

The shape of MSH 15-52, with the brighter NW region lying closer to the Galactic plane, is typical of Galactic supernova remnants (Seward et al, 1983). It can be explained by assuming that the shell was formed in an area where the interstellar medium increases steeply towards the Galactic plane. The northern part of the shell expanded slowly into a more dense region to form the bright radio, optical and x-ray NW region while the southern part expanded rapidly into a less dense region producing the more diffuse weaker SE arc.
From the diameter of the remnant, Clark and Caswell (1977) estimate the age of MSH 15-52 to be ≈10 000 years. This is inconsistent with the young age of the pulsar and the bright

RCW 89 area. Seward et al (1983) show that an energetic supernova and a low density of the ISM in the area would allow for a younger supernova remnant, which would be more compatible with the pulsar characteristic age. The pulsar and its associated synchrotron plerion lie in a region devoid of radio features (Strom, 1994). This suggests that the plerion and supernova remnant may be physically unrelated and merely coincidentally lie on the same line of sight (van den Bergh and Kamper, 1984; Strom, 1994). Strom (1994) presents recent distance measurements which support this hypothesis. ROSAT observations indicate larger HI absorption towards the pulsar than towards RCW 89. The pulsar distance would then be larger ($\approx$ 5.9 kPc) than the 4.2 kPc determined from H1 measurements for RCW 89. A larger distance for the pulsar means that the pulsar cannot be associated with the guest star of AD 185 as it would have been too faint.

The radio spectral index of MSH 15-52 was previously not well determined. Most existing spectral index values were determined using fluxes at only two frequencies (Du Plessis et al, 1995), often measured with different telescopes. There are problems inherent in combining these measurements if the maps are not absolutely calibrated and have different, uncertain, backgrounds. It was decided to make further independent radio observations of MSH 15-52 at HartRAO. These would perhaps shed some light on the morphology of the remnant. Du Plessis et al wished to determine whether the radio spectral index could be extrapolated to the x-ray spectral index.

6.3. Observations

I used the program Beam (section 3.2) to map MSH 15-52 at 5000 MHz and 8500 MHz. The maps were made using an orthographic coordinate projection centred on (RA = 227.5° and DEC = -59°). At 8500 MHz I mapped an area of 1.5° by 1.5° using a scan spacing of 0.040°, which marginally satisfies the Nyquist criteria, while at 5000 MHz I mapped a slightly larger area of 2° by 2° using a scan spacing of 0.050°. I extracted the relevant section from the Rhodes 2326 MHz survey and transformed it to the same source centred coordinate projection as the observations..

6.3.1. Results

Plate 2 (overleaf): False colour image of MSH 15-52 at 8500 MHz. The contour levels are 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5 K in full beam brightness temperatures.

Plate 3 (overleaf): False colour image of MSH 15-52 at 5000 MHz. The contour levels are 0.1, 0.2, 0.4, 0.8 K in full beam brightness temperatures.

The most prominent features on the two maps are the two bright arcs of MSH 15-52. These arcs are separated by 20`. The NW arc has an extension to the NE, while the southern arc consists of two bright regions. This is consistent with the observations of Milne et al (1993) at 4.8 GHz and 8.4 GHz.

The faint emission found by Milne et al (1993) can be seen in the south east of both maps (at l = 0.5°, b = -0.5°). They observed fainter emission to the NE of the southern arc. This can also be seen in 8500 MHz map.

The rms noise level in the 6 cm map is higher than that of the 3.5 cm map by a factor of 4-5. It appears that the narrow band (20 MHz) receiver was inadvertently selected instead of the wide band receiver usually used at 6 cm.
Prominent features on the maps are the two bright arcs. In the 3.5 cm map the faint region found by Caswell et al (1993) can be seen in the south east.

6.4. Spectral index determination

The spectral index of a radio source is an important parameter which is useful in distinguishing between different emission mechanisms.

Plate 2

MSH 15−52

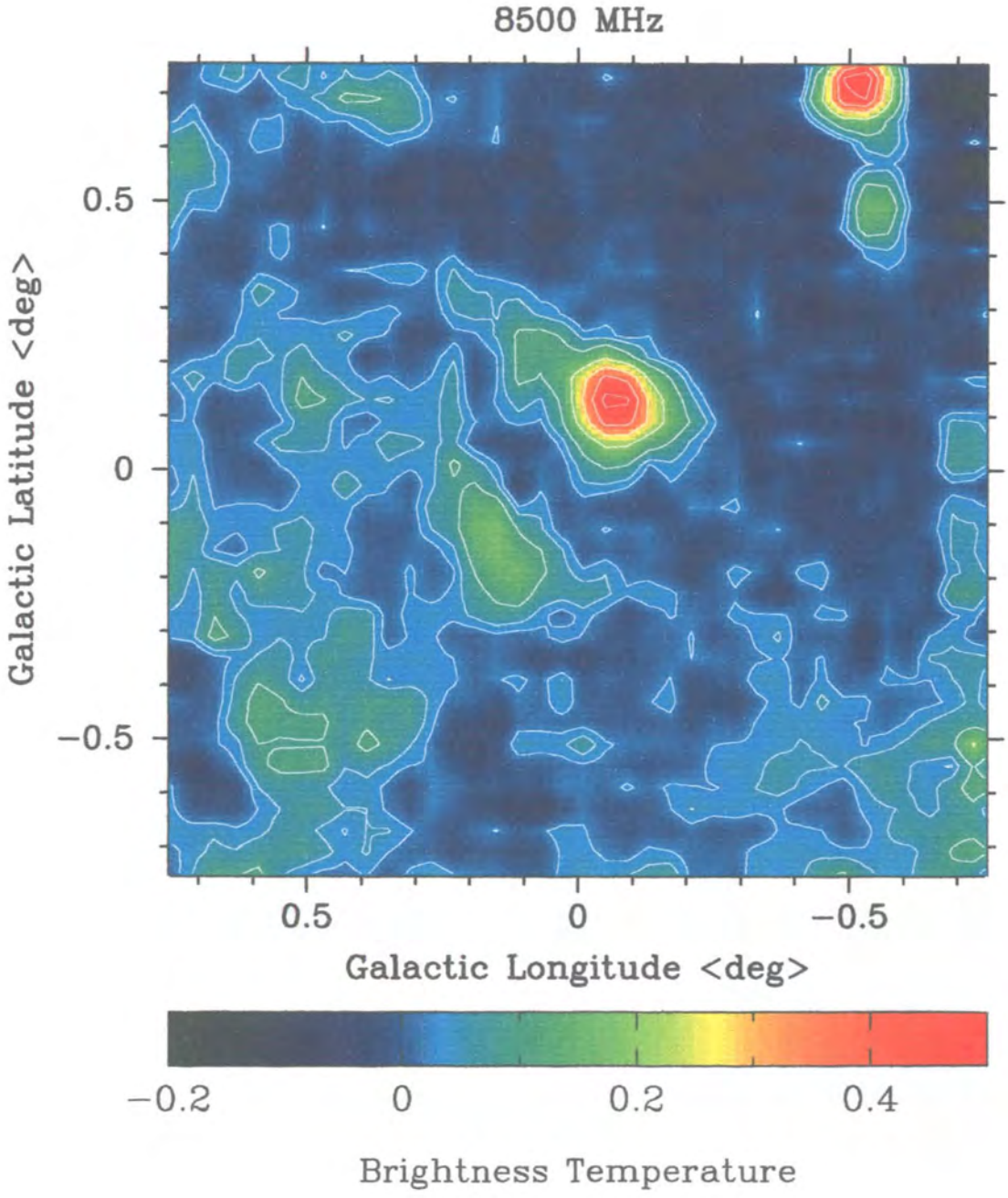
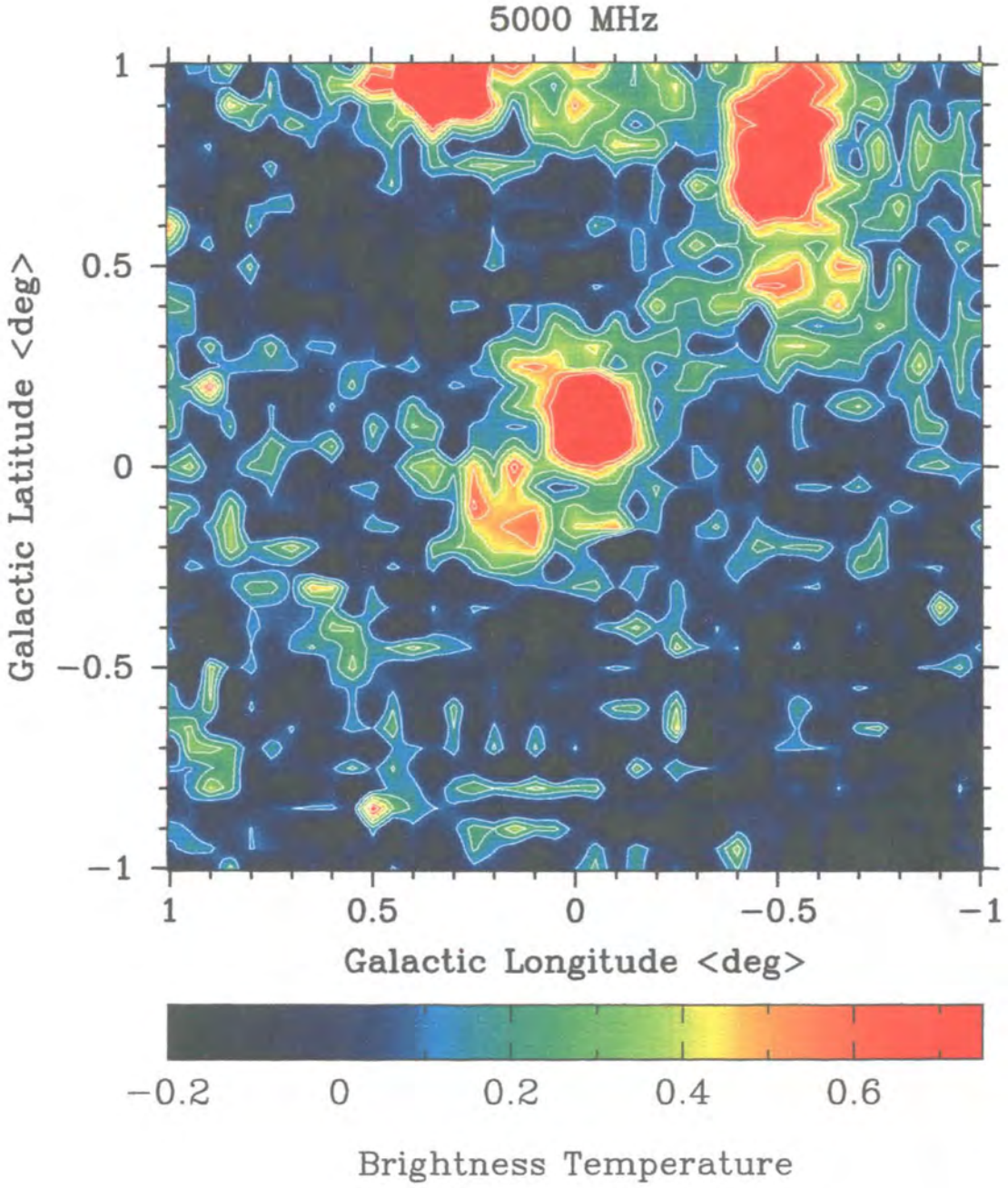


Plate 3

MSH 15−52



The flux spectral index, α, is defined by $S_\upsilon = \upsilon^{-\alpha}$. The temperature spectral index, β, is defined by $T_b = \upsilon^{-\beta}$. This is related to the flux spectral index, α, by $\beta = \alpha + 2$ since:

$$S(\upsilon) = \frac{2k\upsilon^2}{c^2}\iint T_b(\upsilon)d\Omega \quad [W.m^{-2}.Hz^{-1}] \qquad (6\text{-}1)$$

(Kraus, 1966 pg. 87)

I used a combination of two methods to find the spectral index for MSH 15-52. First I used the method of TT-plots (section 6.4.1) to determine a temperature spectral index and to determine the relative backgrounds of the three maps. I then found the integrated flux of the source at each frequency and so determined the spectral index (section 6.4.2).

6.4.1. TT-plots

The temperature spectral index of a radio source may be determined by comparing the observed brightness temperatures at two frequencies. The spectral index of the source is related to the ratio of the brightness temperatures at the two frequencies by

$$\beta = -\frac{\log\left(T_b(\upsilon_1)/T_b(\upsilon_2)\right)}{\log\left(\upsilon_1/\upsilon_2\right)} \qquad (6\text{-}2)$$

The spectral index can be calculated by finding a pixel-by-pixel ratio of the temperatures over the area under consideration. Another method of determining the temperature spectral index is that of TT-plots (Turtle et al, 1962). For each point in the region the temperature at the first frequency (T_1) is plotted against the temperature at the second frequency (T_2). A straight line, described by equation 6-3 is then fitted to these points using a least squares fitting method.

$$T_1 = kT_2 + c \quad [K] \qquad (6\text{-}3)$$

This method avoids problems arising from different backgrounds in the maps.

The temperature spectral index can then be calculated from the slope of this line using

$$\beta = \frac{\log k}{\log\left(\upsilon_1 / \upsilon_2\right)} \qquad (6\text{-}4)$$

The y-intercept, c, gives the difference between the temperature backgrounds of the two maps.

As the resolution of the 5000 MHz and 8500 MHz maps are higher than that of the 2300 MHz map I smoothed the high frequency maps to the resolution of the 2300 MHz map (0.33°). This is done by convolving the maps with a Gaussian of half-power beam width given by

$$\theta = \sqrt{\theta_F^{\,2} - \theta_I^{\,2}} \qquad (6\text{-}5)$$

where θ_F is the desired half-power beam width and θ_I is the initial resolution of the map.

The resolutions of the three maps and half-power beam width of the required Gaussian smoothing beam are tabulated below.

Frequency	Resolution	Gaussian width
2300 MHz	0.33°	N/A
5000 MHz	0.15°	0.29°
8500 MHz	0.1°	0.31°

I performed this convolution using a coordinate transform program (ctfm.f) written by Jonas. In addition to smoothing the maps I re-gridded the 5000 and 8500 MHz maps onto a 0.1° grid and selected a 1° by 1° area centred on the source. I extracted the relevant area from the 2300 MHz map. As there was a pointing error in the 2300 MHz map it was necessary to shift the map to agree with the other 2 maps.

I then did TT-plots between each pair of frequencies using a program written by Jonas (tvst.f). For the TT-plots I used only the central 0.5° by 0.5° of each map.

6.4.1.1. Results from TT-plots

Figures 6-1 to 6-3 show TT-plots for pairs of frequencies. For each plot two spectral index values were found (assuming errors in only the x or y coordinate respectively) and the average determined.

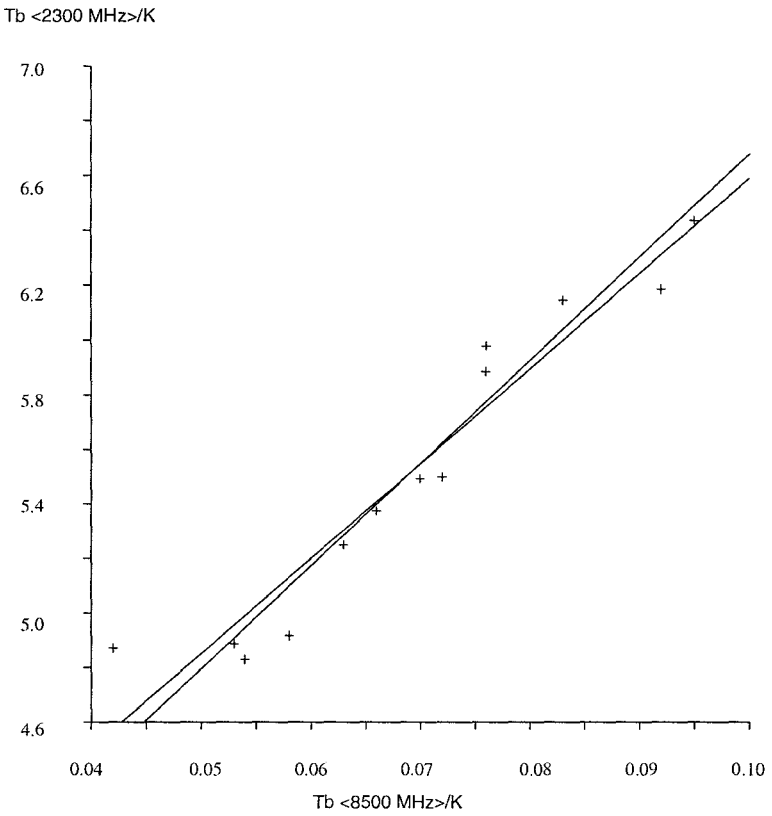


Figure 6-1 T-T plot of 2300 MHz vs. 8500 MHz. The average spectral index is $\beta = -2.76$

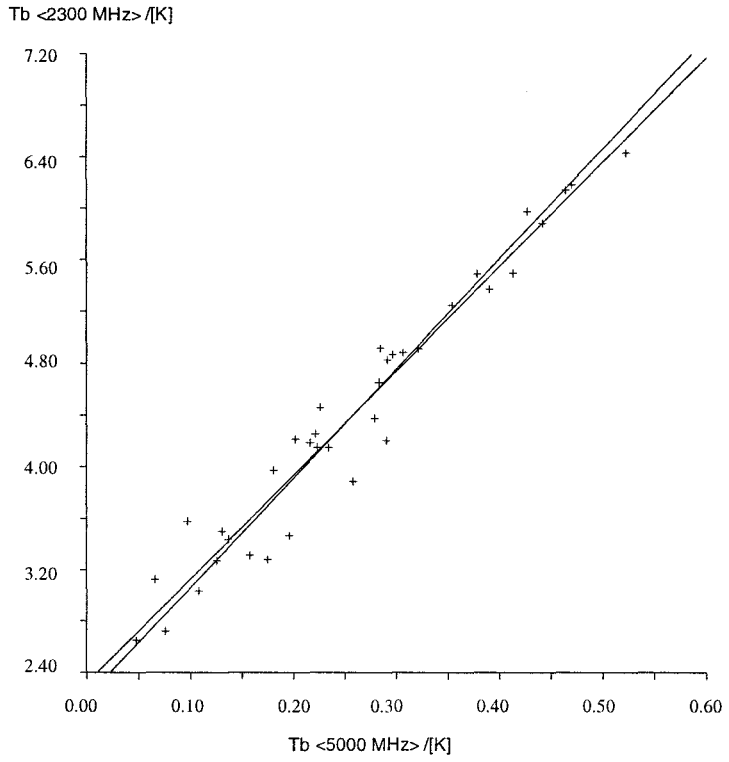


Figure 6-2 TT-plot of 2300 MHz vs. 5000 MHz. The average spectral index is $\beta = -2.73$

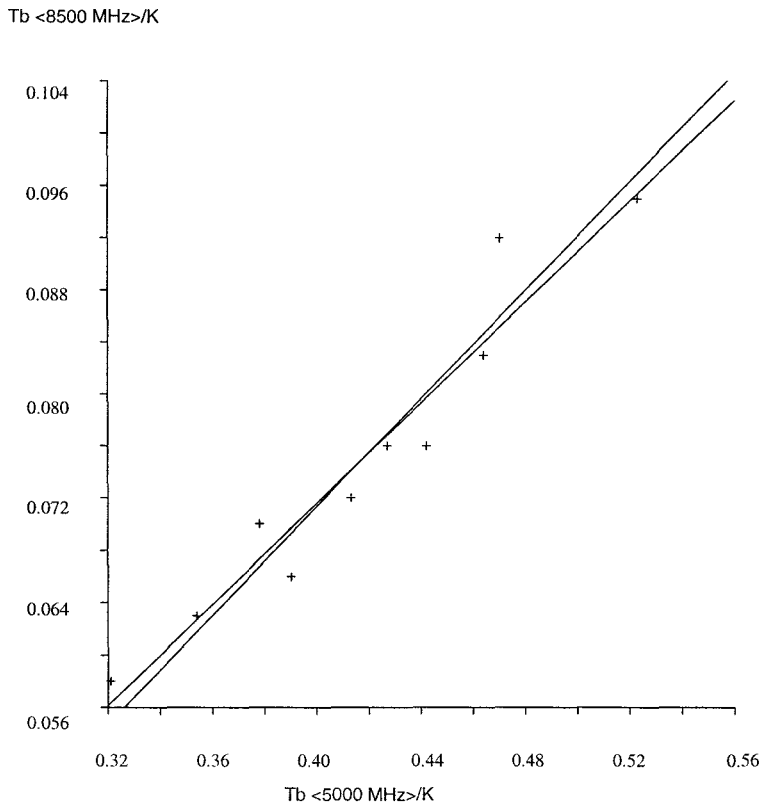


Figure 6-3 T-T plot of 8500 MHz vs. 5000 MHz. The average spectral index is $\beta = -3.03$.

Combining the three TT-plots gives an average temperature spectral index for MSH 15-52 of $\beta = 2.84$ (corresponding to $\alpha = 0.84$).

I used the y-intercepts of the plots to find the temperature backgrounds of the 8500 MHz and 2300 MHz maps relative to 5000 MHz data. The 2300 MHz and 8500 MHz data had relative backgrounds of 2.27 K and -0.007 K respectively.

6.4.2. Using integrated flux to determine spectral index

The second method for finding the spectral index was directly from the three integrated fluxes.

I extracted brightness temperature values of the source from the fits map into a text file (using the program FROMFITS) and imported these values into a spreadsheet. After a background was subtracted I summed the brightness temperatures over the source and determined the flux of the source.

Initially I set the background of the 5000 MHz map to zero. This value determined the backgrounds of the other maps and hence the fluxes and the spectral index. In order to choose the best background for the 5000 MHz map, I calculated flux values at the 3 frequencies assuming different 5000 MHz backgrounds. From these fluxes I determined a spectral index and uncertainty using least squares regression technique and the R^2 value to find the best fit.

This is summarised in table 6-1

5000 MHz background [K]	2300 MHz Flux [Jy]	5000 MHz Flux [Jy]	8500 MHz Flux [Jy]	Spectral Index	Uncert. in SI	R^2
-0.01	48.5	27.7	22.1	-0.610	0.082	0.9821
-0.005	48.4	27.0	20.2	-0.676	0.055	0.9935
0	48.3	26.4	18.2	-0.748	0.022	0.9982
0.005	48.1	25.7	16.2	-0.829	0.015	0.9993
0.01	48.0	25.0	14.3	-0.920	0.060	0.9915

Table 6-1 Effect of 5000 MHz background on spectral index

Choosing the 5000 MHz background as 0.005 K gives the straight line with best fit and the lowest uncertainty in the spectral index. Using this background value, the flux of MSH 15-52 is 16.2 Jy at 8500 MHz, 25.7 Jy at 5000 MHz and 48.1 Jy at 2300 MHz.

I plotted log(Flux) against log(Frequency) for these values (see figure 6-7). Using a least squares method I found the slope and hence the spectral index:

$$\alpha = 0.831 \pm 0.017$$

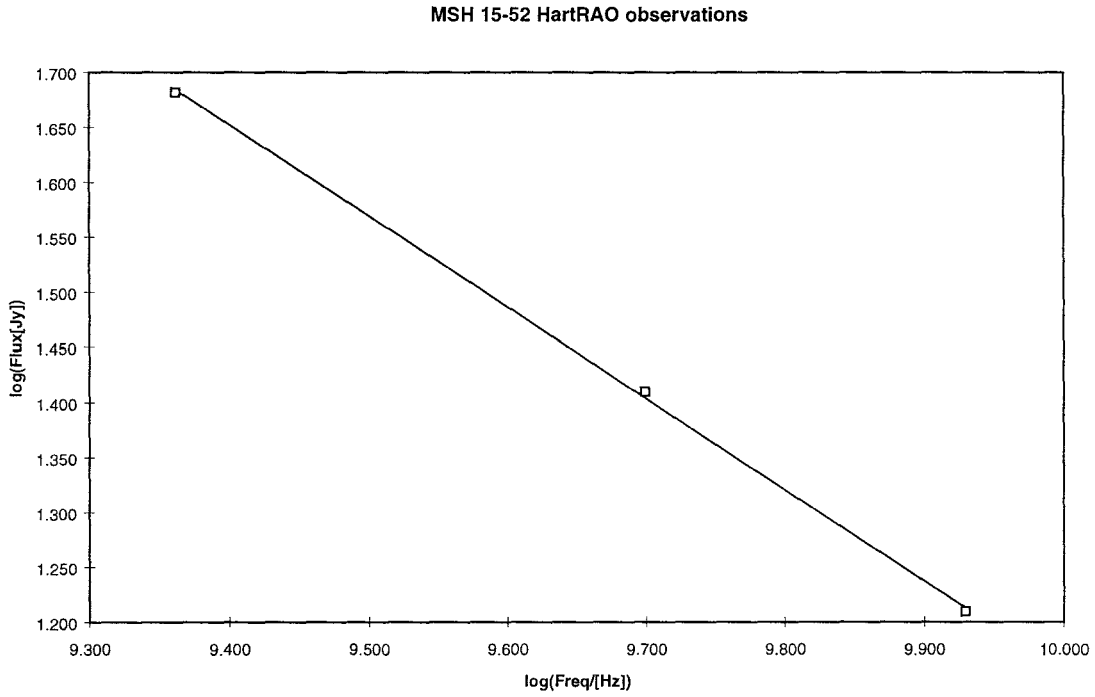


Figure 6-4 Spectral index plot for measurements made at HartRAO

6.5. Results

A number of other flux measurements have been made of MSH 15-52 at radio frequencies. These are summarised in table 6-2.

I combined these values with the measurements from HartRAO to find a spectral index of 0.52 ± 0.052. The values are shown in figure 6-5.

Freq. (MHz)	Flux Jy	Ref.
85.7	252	1
408	94	2
1000	65	4
1000	58.4	2
1410	46	5
2650	23	6
2700	41	5
4800	37±7	7
5000	40	3
8400	24	7

1. Mills, Slee & Hill (1960)
2. Shaver & Goss (1970)
3. Clark & Caswell (1976)
4. Milne (1970)
5. Milne (1972)
6. Day, Thomas & Goss (1969)
7. Milne, Caswell & Haynes (1993)

Table 6-2 Flux values of MSH 15-52 at different frequencies

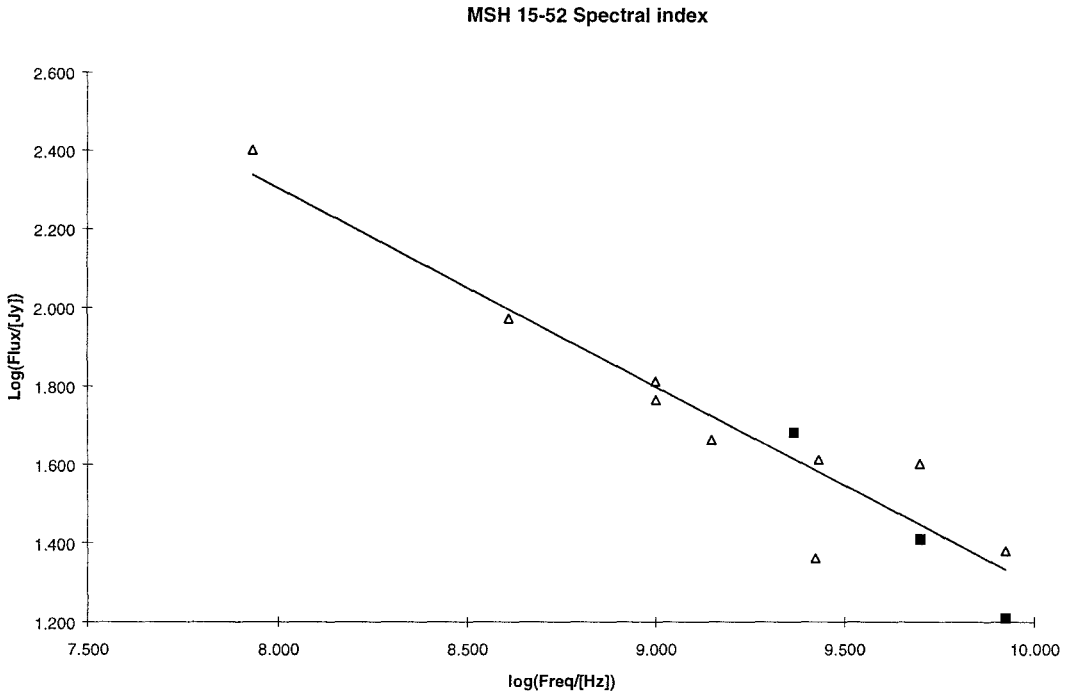


Figure 6-5 Flux Vs Freq. for MSH 15-52 including all measurements in table 6-2. Measurements made at HartRAO are shown as filled squares.

6.6. Discussion

The radio spectral index determined using the measurements made at HartRAO did not extrapolate to the plerionic x-ray spectrum (du Plessis et al, 1995). They concluded that either the plerion was unrelated to the remnant or that the radio emission was from a shell around the plerion and hence would not have the same spectrum.

The spectral index determined using the measurements made at HartRAO is steeper than the spectral index determined from the other measurements and the fluxes are lower. Inconsistencies between observations using different telescopes are not uncommon (Du Plessis et al, 1995).

However the three fluxes measured at HartRAO are internally consistent. Milne et al (1993) made measurements of MSH 15-52 at 4.8 MHz and 8.4 MHz. The spectral index derived from the fluxes at these two frequencies is 0.82 which is within experimental error of the result found at HartRAO. Milne et al (1993) combined their measurements with other flux measurements to give a spectral index of 0.45 (compare with the value 0.52 shown in figure 6-5). It appears that the spectral index of MSH 15-52 is steeper at higher frequencies suggesting a possible spectral break.

In the next chapter I look at observations of two areas of the Galactic plane.

Chapter 7.

Galactic plane mapping at 3.5 cm

7.1. Introduction

To test my observation and data analysis techniques and to investigate the feasibility of a 8500 MHz Galactic plane survey, I mapped two test regions of the Galactic plane.

The Galactic plane emission is a combination of thermal emission from HII regions, non-thermal synchrotron emission from supernova remnants (SNRs) and diffuse background synchrotron radiation. Separating these components has been a problem since the earliest radio surveys. Furst et al (1987) and Haslam & Osborne (1987) have independently developed a technique for isolating non-thermal emission by comparing the radio emission to the far-infrared. I used this technique to identify potential non-thermal sources in the two regions.

In this chapter I present the maps of the regions I observed and discuss the identification and separation of the non-thermal from the thermal sources. Finally I discuss the rms noise level of the maps.

7.2. Observations

I used the 3.5 cm total power receiver of the HartRAO telescope to observe two areas of the Galactic plane. The first region, which will be referred to as region A, extended from $l = 47.5°$

to 52.5° and covered a latitude range of ±2.5°, was observed over two nights in September 1992 using a scan spacing of 0.040°.

The second area was more carefully selected as containing a number of sources. As little structure was observed beyond 1.5° Galactic latitude in the first map I decided to restrict the latitude range of the map to ±1.5°. This area, called region B, was originally intended to stretch from 330° to 335°. However I had to discard the first night's observations as a bug in the observing program allowed the observer to choose an observation name of any length. This led to a non-unique naming system for the scan files. Due to observation time constraints I reduced the area of the map to l = 330.4° to 334.6°. The scan spacing was 0.050°.

The two observations were then processed using the data analysis method outlined in chapter 4 and finally FITS maps were produced in units of brightness temperature. The resolution of the maps, defined by the half-power beamwidth, is 0.1°.

7.3. Comparison of far infrared and radio data

In 1983 the IRAS satellite observed the infrared sky at four wavelength bands, 12, 25, 60 and 100 μm. HII regions are relatively strong emitters in the 60 μm band while supernova remnants are poor IR emitters. A useful quantity to define is the ratio of the infrared brightness at 60 μm to the radio brightness at centimetre wavelengths, termed the infrared radio ratio (IRR). For HII regions the IRR is ≥ 500 while for supernova remnants the ratio is < 20 (Broadbent et al , 1989, Dwek & Arendt, 1992). Arendt (1989) suggests that the SNR sample chosen by Furst et al (1987) was limited and biased towards younger SNRs and that some older SNRs may have higher IRR ranging from 20 to 500.

Comparison of the 60 μm IRAS data with the Effelsburg 2.7 GHz (11 cm) survey shows remarkable spatial correlation for regions of thermal emission (Broadbent et al, 1989) while most known supernova remnants are not visible at FIR frequencies. The detailed correlation

between FIR and radio emission for thermal regions suggests that the 60 μm data, which is near the peak wavelength of infrared emission from HII regions, can be used to predict radio thermal emission. Furst et al (1987), after subtracting large scale structure and HI emission from the IRAS data, found an IRR for thermal regions of 1000. The thermal radio emission could then be predicted by dividing the IRAS 60 μm data by 1000. Subtracting this predicted model from the radio map accentuates the non-thermal emission features which are potential supernova remnant candidates.

7.3.1. Method

I selected 60 μm FIR data for the relevant regions from the IRAS Sky Survey Atlas available on CD-ROM (Wheelock et al, 1994, Beichman et al, 1988). For meaningful comparison, large scale structure must be subtracted from both the radio and infrared maps, and both maps must be at the same resolution and in the same units. The basket weaving process subtracts large scale structure from the radio map and I used IRAS maps from which a gaussian diffuse Galactic background model had been subtracted. The beamwidth of the IRAS beam is not well defined as the beam is not gaussian but rectangular in profile. The resolution of the IRAS data is 4`-6` while that of the radio data is 6`. I smoothed both the IRAS and radio to a resolution of 7` using gaussian convolution filters of 6` and 3.6` respectively.

To calculate the IRR I converted both maps into units of Jy/beam. The units of the IRAS maps on the CD-ROM are MJy/steradian. To convert to Jy/beam involves dividing by 10^6 and multiplying by the beam area in steradians. The solid angle of a gaussian beam of half power beamwidth θ_{HP} (radians) is

$$\Omega = 1.133 \times (\theta_{HP})^2 \qquad (7\text{-}1)$$

For a beamwidth of 7` the beam solid angle is 4.75×10^{-6} steradians so the IRAS data must be multiplied by 4.8 to convert it into Jy/beam.

The original 3.5 cm map is in units of full beam brightness temperature. To convert this to Jy/beam we need to know the relationship between the flux received by the full beam and T_{FB}. The flux of the beam is given by

$$S = \frac{10^{26} \times 2kT_{FB}\Omega_{FB}}{\lambda^2} \text{ [Jy]} \qquad (7\text{-}2)$$

For a point source with a peak temperature of T_{FB} the flux to full beam brightness temperature ratio is :-

$$\frac{S}{T_{FB}} = \frac{10^{26}2k}{\lambda^2} 1.133(\theta_{HP})^2 = 10.1 \text{ [Jy/K]} \qquad (7\text{-}3)$$

where a gaussian profile for the full beam has been assumed. Hence to convert the full beam brightness temperature radio maps to units of Jy/beam they must be multiplied by 10.1.

I used a program, radiofir, written by Jonas to compute the nominal ratio of IRAS to radio brightness. The FIR data divided by the nominal ratio is used as a model for the radio thermal emission and the non-thermal emission is given by

$$\text{nonthermal} = MAP_{8500MHz} - \frac{1}{\text{ratio}} MAP_{FIR} \quad \text{[Jy/beam]} \qquad (7\text{-}4)$$

7.4. Region A: L = 47.5° to 52.5°

The 3.5 cm radio emission in Region A is shown as a false colour image in plate 4. The stronger emission in the plane is saturated to bring out the fainter emission. For comparison, figure 7-1 shows a map of the region taken from the Effelsburg 2700 MHz survey (Reich et al, 1984). This map has been smoothed to the resolution of the 8500 MHz data, and converted from units of mK to units of K.

In both FIR and radio maps the most prominent feature is the supernova which is superimposed on the HII region W51. The supernova remnant G49.2-0.7 (Green, 1996) is seen in the radio data but not in the IRAS 60 μm data (Arendt, 1989).

Plate 4 (overleaf): False colour image of Region A at 8500 MHz. Contours are at 0.1, 0.2, 0.4 and 0.8 K T_{FB}.

Plate 5 (overleaf): Region A

Top: Thermal emission modelled on 60 μm FIR data. Contour levels are 0.125, 0.25, 0.5, 1, 2, 4 Jy/beam

Bottom: Potential non-thermal emission after a thermal emission model has been subtracted from the 8500 MHz radio image. Contour levels are 0.2, 0.4 and 0.8 Jy/beam.(1 K T_{FB} = 10.1 Jy/beam)

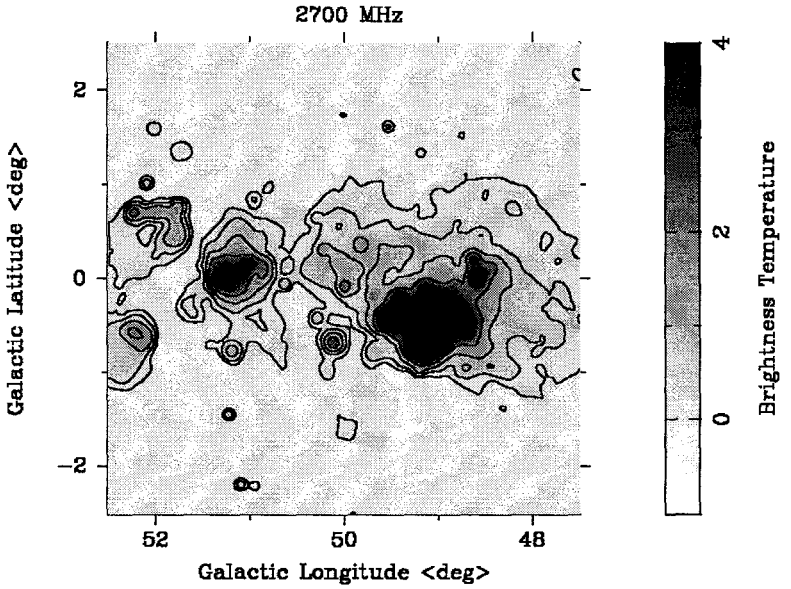


Figure 7-1 Map of region A from the Effelsburg 2.7 MHz survey. Contours are drawn at 0.25, 0.5, 1, 2,4 K

The nominal ratio of the IRAS to the radio brightness is approximately 850. I subtracted $\frac{1}{850}$ of the FIR data from the radio image to get the non-thermal emission. Plate 5 shows the non-thermal and thermal emission in units of Jy/beam (1 K = 10.1 Jy/beam).

In the non-thermal map, G49.2-0.7 can be easily identified. There is also an area of potential non-thermal emission at l = 51.2° b= 0.0° which is not associated with a supernova remnant.

Plate 4

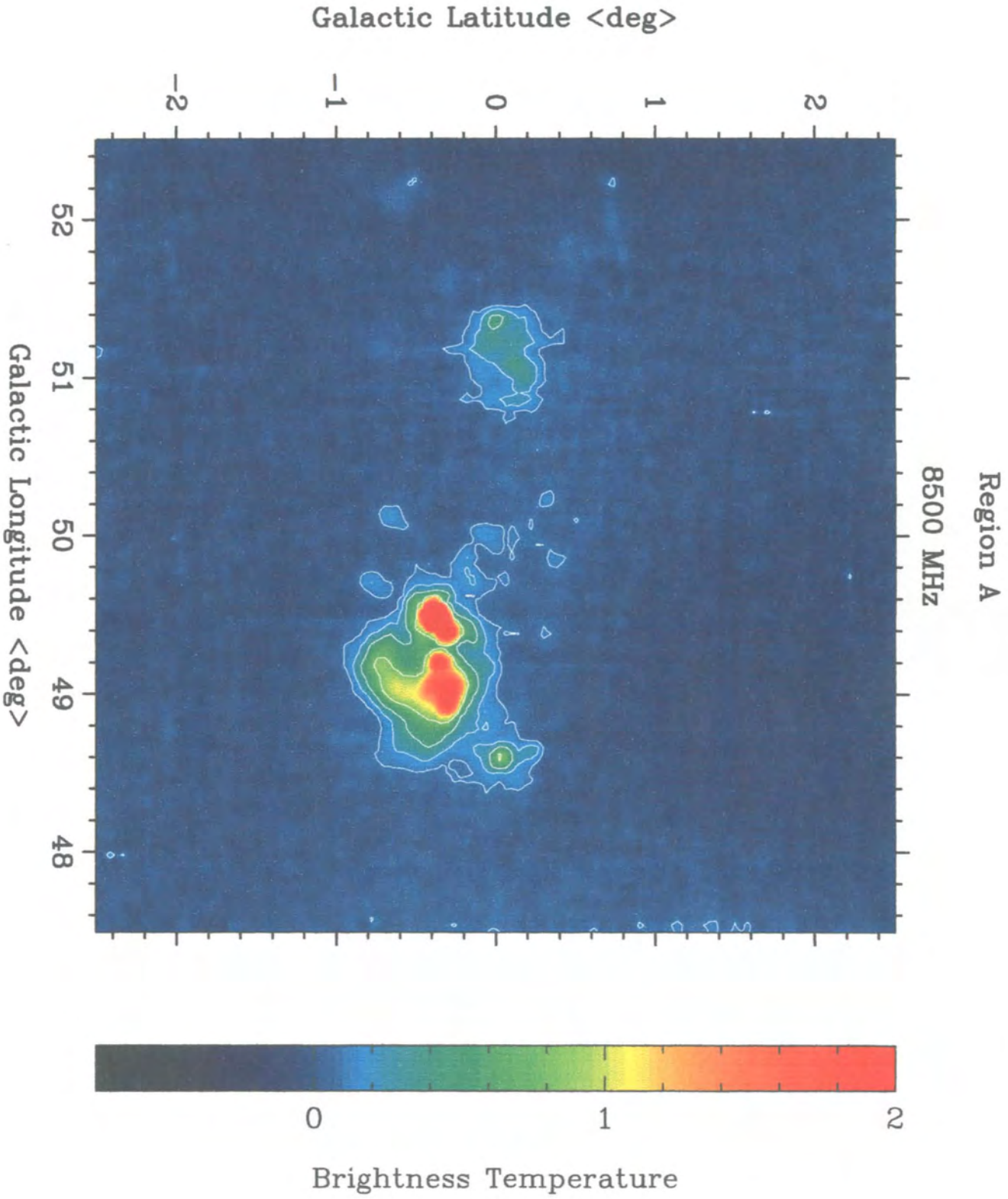

Galactic Latitude <deg>
−2
−1
0
1
2
Galactic Longitude <deg>
52
51
50
49
48
Region A
8500 MHz
0
1
2
Brightness Temperature

Plate 5

Region A

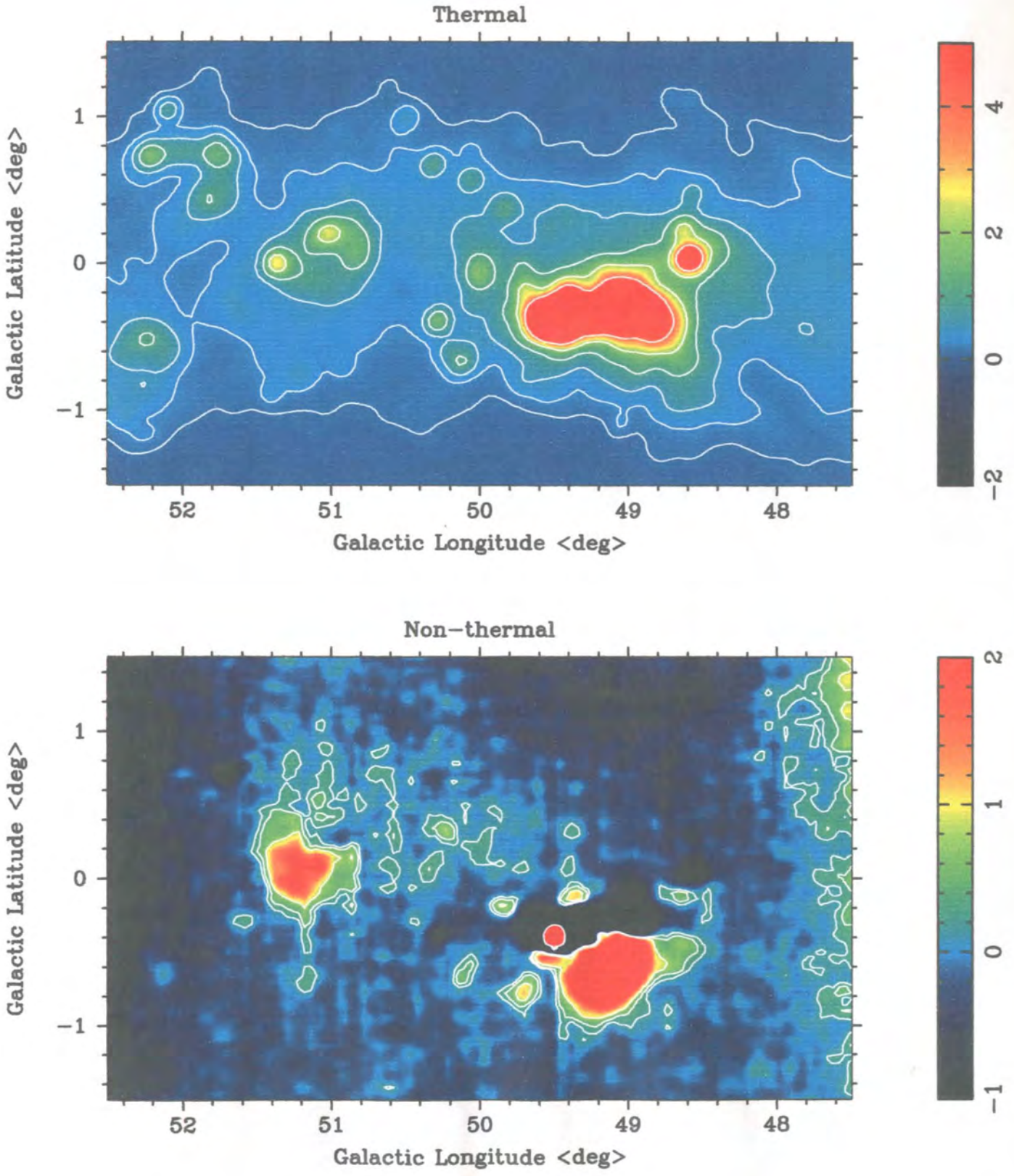


Plate 6 (overleaf): False colour image of Region B at 8500 MHz. Contours are at 0.1, 0.2, 0.4 and 0.8 K T_{FB}.

Plate 7 (overleaf): Region B

Top: Thermal emission modelled on 60 μm FIR data. Contour levels are 1, 2, 4, 8 and 16 Jy/beam

Bottom: Potential non-thermal emission after a thermal emission model has been subtracted from the 8500 MHz radio image. Contour levels are 0.2, 0.4 and 0.8 Jy/beam.(1 K T_{FB} = 10.1 Jy/beam)

7.5. Region B: L = 330.4° to 334.6°

Plate 6 is a false colour image of Region B. For comparison, figure 7-2 shows the same region from the Parkes 5000 MHz survey (Haynes et al, 1978), smoothed to a resolution of 6` and converted to brightness temperatures in K.

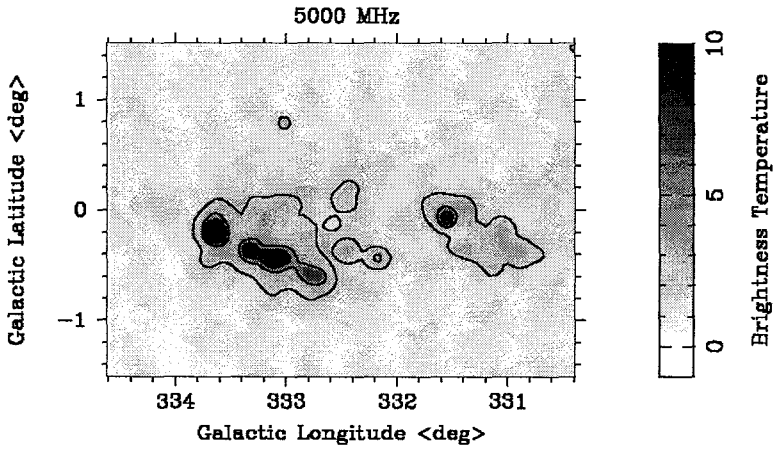


Figure 7-2 Map of region B from the Parkes survey at 5000 MHz. Contours at 2.5, 5, 7.5 and 10 K.

The region contains a number of HII regions along the plane (see the Parkes' survey of southern HII regions, Caswell & Haynes, 1987) which are visible at both 5000 MHz and 8500 MHz.

Region B contains three supernova remnants listed in Green's catalogue (Green, 1996). After removing a model of the thermal component to get the non-thermal component a number of

Plate 6

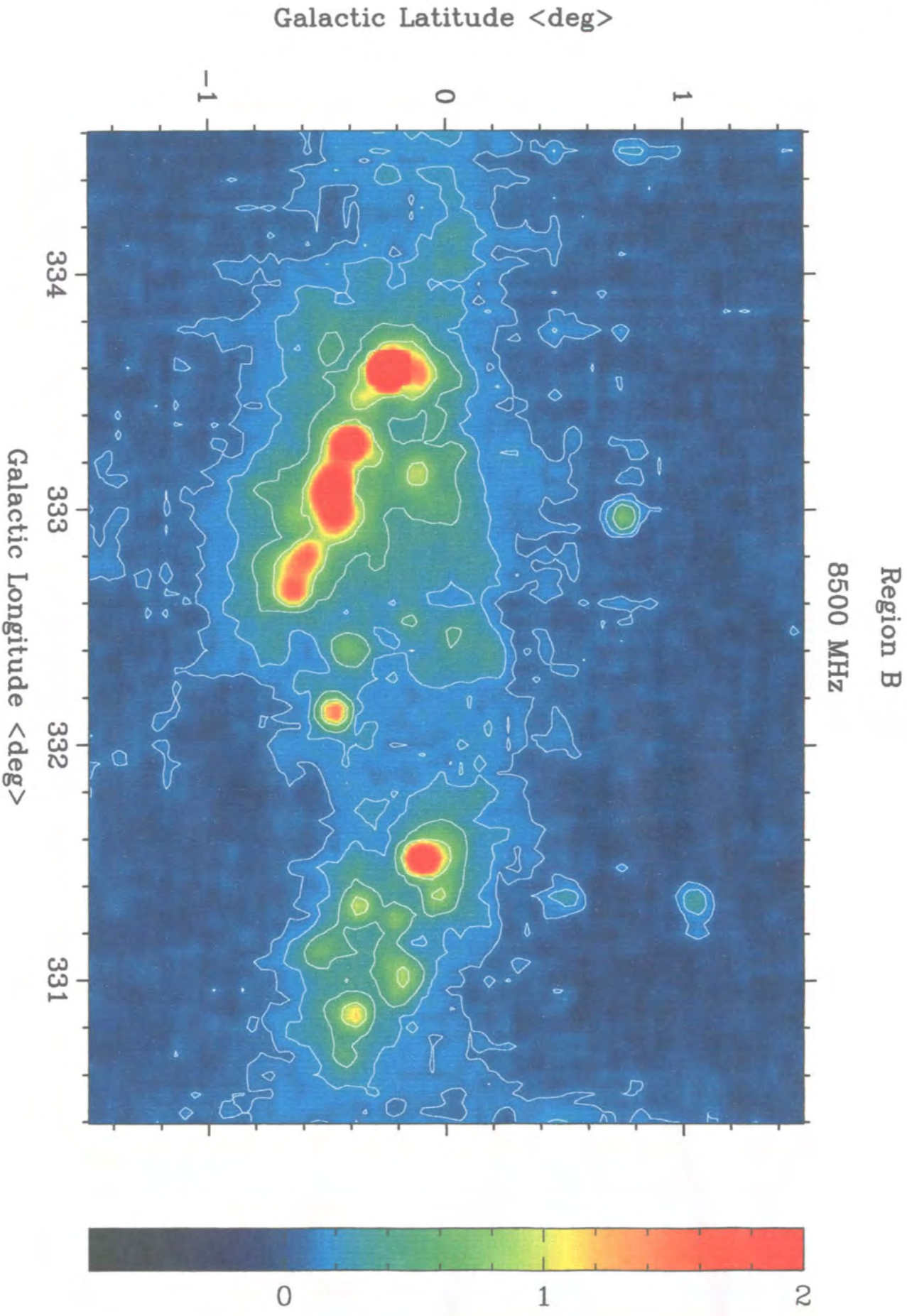
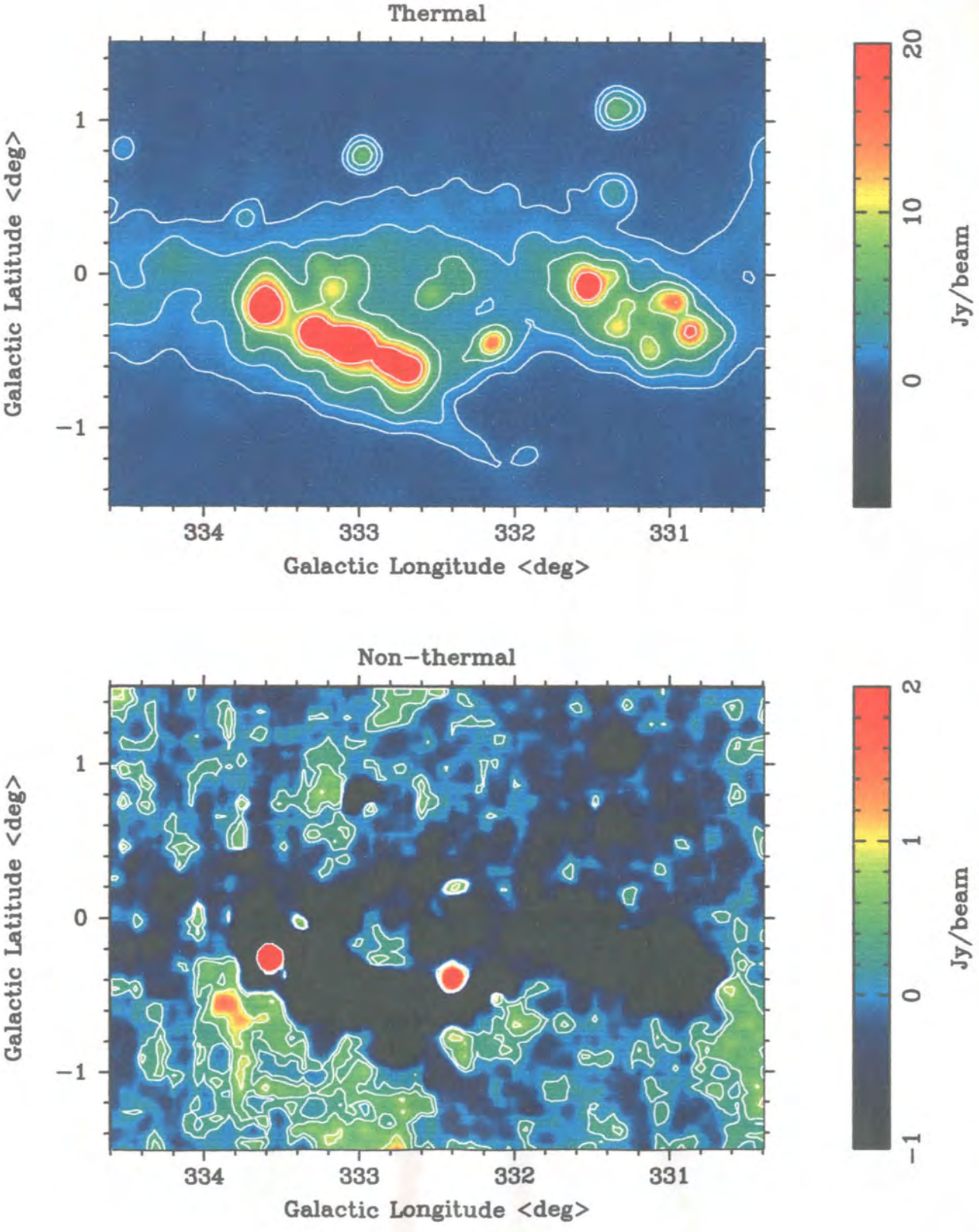

Galactic Latitude <deg>
−1
0
1
334
333
332
331
Galactic Longitude <deg>
Region B
8500 MHz
0
1
2

Plate 7

Region B



potential non-thermal sources are revealed. The thermal and non-thermal components are shown in plate 7.

The strongest known supernova remnant in the region is G332.4-0.4 (RCW 103). A comparison of the radio and infrared map plates shows G332.4-0.4 is present in the radio map but not in the infrared. It can be clearly seen in the non-thermal map.

The extended (12` × 13`) supernova remnant G332.4+0.1 or MSH 16-51 (identified by Green, 1996) can be seen in the non-thermal map. It contains a bright IR source (Kes 32) on the rim of the radio shell. Although the eastern edge of the source coincides with the edge of the radio shell it is smaller in size and emits strongly at 12 and 25 μm and is thought to be unrelated to the supernova (Saken et al, 1992).

A faint object in the non-thermal map (at l = 332.0, b = 0.2) is possibly the supernova remnant G332.0+0.2 in the Green catalogue.

There are a number of other areas of potential non-thermal emission in the map. Duncan et al (1997) identify an extended (18° diameter) supernova remnant G325+0 in the 2400 MHz Parkes survey. The diffuse emission around l = 334.0° appears to be part of the eastern arc of the supernova remnant.

7.6. Noise

To measure the noise in the maps I selected source-free areas as input to a program written by Jonas to determine the rms noise level. For each region chosen the program subtracted a second order polynomial background surface and determined the rms noise from the residual data. Table 7-1 presents the results of this analysis. The second and third columns give the coordinates of the centre of the box, the next two columns give the size of the box and the last column gives the measured rms noise in mK.

Region	Long	Lat	Long size	Lat size	RMS noise
A	50.0	2.0	3.5	0.8	24.9
A	52.0	2.0	0.8	0.8	18.7
A	51.9	-1.0	1.0	0.6	17.7
A	48.8	-1.5	1.3	0.6	18.8
B	331.0	-1.1	1.1	0.4	25.1
B	333.2	1.3	1.0	0.4	27.4
B	330.9	1.2	0.6	0.6	23.0
B	334.2	-1.2	0.7	0.4	26.7

Table 7-1 RMS noise values

The average noise is ≈ 23 mK which is much higher than the value expected from the radiometer equation. The theoretical noise level for the total power system is given by

$$\Delta T = \frac{T_{SYS}}{\sqrt{2B\tau}} \qquad (7\text{-}5)$$

where T_{sys} is the system noise temperature, B the bandwidth and τ the integration time (Kraus,1966, pg. 244)

At the time the maps were made the 3.5 cm system had a bandwidth of 400 MHz. and a system temperature of 65 K. Scanning at a rate of 0.05° per second with a bin spacing of 0.04° gives an integration time per pixel of 1.25 s for one observation. Effectively each pixel was scanned twice giving an integration time of 2.5s. This gives a theoretical noise level of 2 mK using equation 7-5.

The total power noise consists of two components - the broadband receiver noise given by equation 7-5 and 1/f noise resulting from fluctuations in the receiver gain. The total power noise is dominated by 1/f noise which degrades the rms noise by a factor of 3-4 (Nicolson, private communication). In Dicke mode the system performs at the expected theoretical level. This explains the discrepancy between measured and predicted rms noise values. The raw

scan data had 1/f noise consistent with that of the final map thus the processing does not add noise to the data.

The 1/f noise could be significantly reduced by using a switched radiometer, such as the noise adding system used for the Rhodes/HartRAO 2326 MHz survey observations. At the time the observations were made this was not possible and so the maps were made in total power mode.

7.7. Conclusion

The primary aim of this section was to test my observing and analysis programs. The maps confirm that the basket weaving method is effective at removing scanning effects and providing useful data. A 3.5 cm Galactic plane survey appears to be feasible from these preliminary observations. The noise in the maps could be substantially reduced by using a switched radiometer (section 7-6) and by making a series of overlapping maps.

The following chapter discusses plans for further observations and evaluates the system.

Chapter 8.

Conclusion

In this thesis I have achieved my aim of developing, implementing and testing an observing and data reduction system for making radio continuum maps at HartRAO. The observing program makes use of the new antenna control system. The basket weaving technique is very effective in removing significant scanning effects from the maps.

I have tested my mapping techniques by making the following observations:

1. I observed the supernova remnant MSH 15-52 at 8500 MHz and 5000 MHz. Using these observations, together with data from the Rhodes/HartRAO 2326 MHz survey, I found the spectral index of the remnant to be $\alpha = 0.83 \pm 0.02$.
2. I observed two regions of the Galactic plane at 8500 MHz. These maps have a resolution of 6` and a measured rms noise of 23 mK.
3. I used IRAS 60 μm FIR data to separate the thermal and non-thermal components of the emission for the Galactic plane maps.

These radio continuum observing and data analysis programs will be incorporated into the standard set of observing procedures at HartRAO. The mapping procedure will be used for scientific and maintenance purposes.

A survey of the southern Galactic plane at 3.5 cm appears to be feasible. The test observations were made in total power mode and the rms noise was degraded by low frequency gain fluctuations. This noise could be reduced by using a switched radiometer (see Chapter 7-7). The noise could be further reduced by making a series of overlapping maps along the plane.

Once the 3.5 cm system has been upgraded, in the second half of 1998 (Nicolson, private communication) it will be possible to make dual polarisation and polarimetry observations, which will increase the scientific value of the resulting maps.

References

Arendt RG: 1989, Astrophys.J.Suppl, 70, 181

Beichman CA, Neugebauer G, Habing HJ, Clegg PE, Chester TJ: 1988, IRAS Explanatory Supplement

Bracewell RN: 1986, 'The Fourier Transform and its Applications', 2nd edition, revised, McGraw-Hill International

Broadbent A, Haslam CGT, Osborne JL: 1989, Mon.Not.R.Astro.Soc, 237, 381

Cane HV, Whitham PS: 1977, Mon.Not.R.Astro.Soc, 179, 21

Caswell JL, Murray DJ, Roger RS, Cole DJ, Cooke DJ, Astron.Astrophys, 45, 239

Caswell JL, Milne DK and Wellington KJ: 1981, Mon.Not.R.Astro.Soc, 195, 89

Caswell JL, Haynes RF: 1987, Astron.Astrophys, 171, 261

Chin YN, Huang YL: 1994, Nature, 371, 398

Clark DH, Caswell JL: 1977, Mon.Not.R.Astro.Soc, 174, 267

Day GA, Caswell JL, Cooke DJ: 1972, Aust.J.Phys.Astrophys.Suppl, 25,1

Duncan AR, Stewart RT, Haynes RF, Jones KL: 1995, Mon.Not.R.Astro.Soc, 277, 36

Duncan AR, Stewart RT, Haynes RF, Jones KL: 1997, Mon.Not.R.Astro.Soc, 287, 722

du Plessis I: 1994, MSc thesis, Potchefstroom Univ.

du Plessis I, de Jager OC, Büchner S, Nel HI, North AR, Raubenheimer BC, van der Walt DJ: 1995, Astrophys.J, 53, 746

Dwek E, Arendt RG: 1992, Annu.Rev.AA, 30:11

Emerson DT, Gräve R: 1988, Astron.Astrophys, 190, 353

Furst E, Reich W, Sofue Y: 1987, Astron.Astrophys.Suppl. 71, 63

Green AJ: 1974, Astron.Astrophys.Suppl. 18, 267

Green DA: 1996,'A Catalogue of Galactic Supernova Remnants (1996 August version)', Mullard Radio Observatory, Cambridge, UK (available on the World-Wide-Web at "http://www.mrao.cam.ac.uk/surveys/snrs/"

Greiveldinger C, Caucino S, Massaglia S, Oegelman H, Trussoni E: 1995, Astrophys.J, 454, 855

Haslam CGT, Wilson WE, Graham DA, Hunt GC: 1974,Astron.Astrophys.Suppl, 13, 359

Haslam CGT, Osborne JL: 1987, Nature, 327, 211

Haynes RF, Caswell JL, Simons LWJ: 1978, Aust.J.Phys.Astrophys.Suppl, 45, 1

Hill ER, Slee OB, Mills BY: 1958, Aust.J.Phys, 11, 530

Hill ER: 1968, Aust.J.Phys, 21, 735

Jonas JL: 1982, MSc thesis, Rhodes University

Jonas JL: 1991,"HartRAO antenna control system", HartRAO internal report

Jonas JL, Baart EE: 1995, Astrophysics and Space Science, 230, 351

Jonas JL, Baart EE, Nicolson G: 1998, Mon.Not.R.Astro.Soc [in press]

Jones BB, Findlay EA: 1974, Aust.J.Phys, 27, 687

Keihm SJ, Langseth MG: 1975, ICARUS 24, 211

Kraus JD: 1966, 'Radio Astronomy', McGraw-Hill

Komesaroff MM: 1966, Aust.J.Phys, 19, 75

Manchester RN, Tuohy IR, D'Amico N: 1982, Astrophys.J, 262, L31

Mathewson DS, Healey JR, Rome JM: 1962, Aust.J.Phys, 15, 354

Milne DK: 1970, Aust.J.Phys, 23, 425

Milne DK, Caswell JL and Haynes RF: 1993, Mon.Not.R.Astro.Soc 264, 853

Mountfort P: 1989, PhD thesis, Rhodes University

Ott M, Witzel A, Quirrenbach A, Krichbaum TP, Standke KJ, Scainski CJ, Hummel CA: 1994, Astron.Astrophys, 284, 331

Reich W, Furst E, Haslam CGT, Steffen P, Reif K: 1984, Astron.Astrophys.Suppl, 58, 197

Saken JM, Fesen RA, Shull JM, 1992: Astrophys.J.Suppl, 81, 715

Schaeffer BE: 1995, Astron.J, 110, 1793

Seward FD, Harnden FR: 1982: Astrophys.J, 256, L45

Seward FD, Harnden FR et al: 1983, Astrophys.J, 267, 698

Shain CA, Komerasoff MM, Higgins DS: 1961, Aust.J.Phys, 14, 508

Sieber W, Haslam CGT, Salter CJ: 1979, Astron.Astrophys, 74,361

Strom RG: 1994: Mon.Not.R.Astro.Soc, 268, L5

Thorsett SE: 1992, Nature, 356, 690

Trussoni E, Massaglia S, Caucino S, Brinkmann W, Aschenbach B: 1996, Astron.Astrophys, 306,581

Turtle AJ, Pugh JF, Kenderdine S, Pauliny-Toth IIK: 1962, Mon.Not.R.Astro.Soc, 124, 297

van den Bergh S and Kamper KW: 1984, Astrophys.J, 280, L51

Wells DC, Greisen EQ, Harten RH: 1981, Astron.Astrophys.Suppl, 44, 363

Wheelock SL et al: 1994, IRAS Sky Survey Atlas Explanatory Supplement, JPL Publication 94

Appendix A

Observation Programs

program Map;

```
uses

   Graph, IO_Utils,GlobVar,MapUtils,
   Crt, dos,Utils,Screen;

var

    NSam,I                  : integer;
    LongRate,LatRate        : real;

    PROCEDURE GetParams(NData : integer);
  (*-----------------*)
  (* initialise variables : size, spacing, LatMin, LongMin, ScanLength,
ScanTime, ScanRate,ReadDVM, FeedSys, ScanIn, StartNo,EndNo,
CoordType,NScan,NData *)
   var
      EndLat,EndLong : real;
      Size           : real (* area over which to scan *);

   BEGIN

      LatMin    :=  -2.5;
      LongMin   := 47.5;
      EndLat    := 2.5;
      EndLong   := 52.5;
      ScanRate  := 0.1;
      Spacing   := 0.20;
      ScanIn    := 'LAT';
      StartNo   := 1;
      CoordType := 'GALA';
      FeedSys   := 3;

      IF ScanIn = 'Lat' THEN
         EndNo := ROUND((EndLong -LongMin)/Spacing)+1
      ELSE
         EndNo := ROUND((EndLat-LatMin)/Spacing)+1;

      SetParams(EndLat,EndLong); (* set params using menu *)
      SysName   := SysNames[FeedSys];
      NoiseTemp := NoiseTemps[FeedSys];

      IF ScanIn = 'LONG' THEN  (* scanning in longitude - at const lat *)
      BEGIN
         Size := EndLat - LatMin;
         ScanLength := EndLong - LongMin;
      END;

      IF ScanIn = 'LAT' THEN  (* scanning in latitude - at const long *)
      BEGIN
         ScanLength := EndLat - LatMin;
         Size       := EndLong - LongMIn;
      END;

      NScan    := Round(Size/Spacing)+1;
      NData    := Trunc(ScanLength / (2*Spacing))*2+1;
      ScanTime := Round((ScanLength+Extra) /ScanRate);
      IF ReadDVM = 'NO' THEN NoiseDVM := 1.0;

   END (*GetParams *);


PROCEDURE DoScan (var ISam: integer);
  (*---------------*)
   var
      IStat,Code                    : integer;
      Long,Lat,AntStatus,Stat,Com   : string;
      Error                         : Boolean;

   begin
      ISam := 0;
      IStat := ReadStatus;
      WHILE (IStat < Halted) AND (Com <> 'GoToScan') DO
      (* sample while antenna is scanning *)
      BEGIN
         ISam := ISam +1;
         ReadString(Antenna,AntStatus,Error);
         IF Error = FALSE THEN
         BEGIN
            Stat:=AntField(StatField,AntStatus);
            Val(Stat,IStat,code);
            Long:=AntField(LongField,AntStatus);
            Val(Long,AntLong[ISam],code);
            Lat := AntField(LatField,AntStatus);
            Val(Lat,AntLat[ISam],code);
            Com := AntField(ComField,AntStatus);
         END
         ELSE
            HandleError(AntStatus);


         IF ReadDVM = 'YES' THEN ReadReal(DVM1,Scan[ISam]);
      END;

   end; (* DoScan *)



   PROCEDURE DoScans ;
  (*---------------*)

   VAR
      IScan, NData          : integer;
      Ch                    : char;
      NextLat,NextLong      : real;

   BEGIN

      SetUp;
```

```
FOR IScan:= StartNo TO EndNo DO
BEGIN
   IF ScanIn = 'LAT' THEN
   BEGIN
      IF Odd(IScan) THEN      (* up scan *)
      BEGIN
         Lat     := LatMin-Extra;
         LatRate := ScanRate;
         NextLat := LatMin + ScanLength + Extra;
      END
      ELSE                    (* down scan *)
      BEGIN
         Lat     := LatMin+ScanLength+Extra;
         LatRate := -ScanRate;
         NextLat := LatMin - Extra;
      END;
      Long     := LongMin + (IScan-1)*Spacing;
      NextLong := Long + Spacing;
      LongRate := 0.0;
   END;

   IF ScanIn = 'LONG' THEN
   BEGIN
      IF Odd(IScan) THEN      (* up scan *)
      BEGIN
         Long     := LongMin-Extra;
         LongRate := ScanRate;
         NextLong := LongMin + ScanLength +Extra;
      END
      ELSE                    (* down scan *)
      BEGIN
         Long     := LongMin+ScanLength+Extra;
         LongRate := -ScanRate;
         NextLong := LongMin - Extra;
      END;
      Lat      := LatMin + (IScan-1)*Spacing;
      NextLat  := Lat + Spacing;
      LatRate  := 0.0;
   END;

   IF IScan = StartNo THEN
   BEGIN
      WriteString(Antenna,'Begin GoToScan');
      WriteString(Antenna,'Clear');
      WriteString(Antenna,'Duration '+IntStr(400));
      WriteString(Antenna,'A0 '+RealStr(Long,8,3));
      WriteString(Antenna,'A1 0.0');
      WriteString(Antenna,'B0'+RealStr(Lat,8,3));
      WriteString(Antenna,'B1 0.0');
      WriteString(Antenna,'End');
   END;

   IF IScan <> StartNo THEN
     IF(NSam > 0) THEN        (* output previous scan *)
     BEGIN
        OutScan(NSam,IScan);
        WriteText(TextView,'Sample No = '+Intstr(NSam),5);
        NewColour;
     END;
   Calibrate(FeedSys,DVM1);
   AntOnSource;               (* wait until antenna is on source *)

   WriteString(Antenna,'Begin Scan_'+IntStr(IScan));
   WriteString(Antenna,'Clear');
   WriteString(Antenna,'Duration '+IntStr(ScanTime+5));
   WriteString(Antenna,'A0 '+RealStr(Long,8,3));
   WriteString(Antenna,'B0 '+RealStr(Lat,8,3));
   WriteString(Antenna,'A1 '+RealStr(LongRate,8,3));
   WriteString(Antenna,'B1 '+RealStr(LatRate,8,3));
   WriteString(Antenna,'End');

   IF IScan <> EndNo THEN
   (* append next command to FIFO *)
   BEGIN
      WriteString(Antenna,'Begin GoToScan');
      WriteString(Antenna,'Append');
      WriteString(Antenna,'Duration '+IntStr(400));
      WriteString(Antenna,'A0 '+RealStr(NextLong,8,3));
      WriteString(Antenna,'A1 0.0');
      WriteString(Antenna,'B0'+RealStr(NextLat,8,3));
      WriteString(Antenna,'B1 0.0');
      WriteString(Antenna,'End');
   END;

   WriteText (TextView,'Doing scan '+IntStr(IScan)+' of
'+IntStr(NScan)+
                    '  Lat = '+RealStr(Lat,8,3)+' Long =
'+RealStr(Long,8,3),5);
   DoScan (NSam);
   IF ReadDVM = 'NO'  THEN ReadScan(NSam,ISCan);

END;
OutScan(NSam,EndNo+1);

WriteText(TextView,'Finished Scanning - Press any key to
continue',5);
REPEAT
   (* wait *)
UNTIL KeyPressed;

END;
var

Finished : Boolean;
begin
   TextBackGround(Black);
   TextColor(White);
   ClrScr;

   Write('Sub Dir : ');
   ReadLn(DirString);
```

```
   Finished := FALSE;
   WHILE Not Finished DO
   BEGIN
      Write('Observation Name ( < 4 characters) : ');
      ReadLn(ObsName);
      IF (Length(ObsName)<= 4) THEN Finished := True
      ELSE
         Write('Observation Name must have four or less characters');
   END;
   ClrScr;

   GetParams(Ndata);

   SetUpScreen;

   DrawGrid;
   LabelAxis;
   OpenViewPort(TextView);

   DoScans;

   CloseGraph;

end.
```

Program Beam;

```
uses

   Graph, IO_Utils, BeamUtil,GlobVar,
   Crt, dos,Utils,Screen ;

var

    NSam,I                          : integer;
    LongRate,LatRate : real;


   PROCEDURE DoScan (var ISam: integer);
  (*---------------*)
   var
      y                          : real;
      IStat,MaxISam,Code         : integer;
      Long,Lat,AntStatus,Stat,Com : string;
      Error                      : Boolean;

   begin
      ISam := 0;
      IStat := ReadStatus;
      WHILE (IStat < Halted) AND (Com <> 'GoToScan') DO
      BEGIN
         ISam := ISam +1;
         ReadString(Antenna,AntStatus,Error);
         IF Error = FALSE THEN
         BEGIN
            Stat := AntField(StatField,AntStatus);
            Val(Stat,IStat,code);
            Long := AntField(UsrLongField,AntStatus);
            Val(Long,AntLong[ISam],code);
            IF AntLong[ISam] > 300 THEN
               AntLong[ISam] := AntLong[ISam] - 360.0;
            Lat := AntField(UsrLatField,AntStatus);
            Val(Lat,AntLat[ISam],code);
            Com := AntField(ComField,AntStatus);
         END
         ELSE
            HandleError(AntStatus);


          IF ReadDVM = 'YES' THEN ReadReal(DVM1,Scan[ISam]);
      END;

   end; (* DoScan *)

   PROCEDURE GetParams(NData : integer);
  (*-----------------*)
   var
      EndLat,EndLong : real;

BEGIN

      SourceRA    := 299.44;
      SourceDec   := 40.6;
      Size        := 1;
      ScanRate    := 0.05;
      FeedSys     := 3;
      Spacing     := 0.04;
      ScanIn      := 'LAT';
      StartNo     := 1;
      EndNo       := 21;

      SetParams;

      SysName     := SysNames[FeedSys];
      NoiseTemp   := NoiseTemps[FeedSys];

      LatMin      := -Size/2;
      LongMin     := -Size/2;
      NScan       := Round(Size/Spacing)+1;
      NData       := Trunc(Size/(2*Spacing))*2+1;
      ScanTime    := Round((Size+Extra)/ScanRate)+5;
      ScanLength  := Size;

   END (*GetParams *);


   PROCEDURE DoScans ;
  (*---------------*)

   VAR
      IScan, NData          : integer;
      Ch : char;
      NextLong,NextLat : real;


   BEGIN

      GetParams(Ndata);

      SetUpScreen;
      DrawGrid;
      LabelAxis;

      OpenViewPort(TextView);

      IF ReadDVM = 'YES' THEN WriteString(DVM1,'*S1T3');

      BeamSetUp;

      FOR IScan:= StartNo TO EndNo DO
      BEGIN
         NewColour;
         IF ScanIn = 'LAT' THEN

         BEGIN
```

```
      IF Odd(ISCan) THEN
      BEGIN
         Lat     := LatMin-Extra;
         NextLat := LatMin + ScanLength + Extra;
         LatRate := ScanRate;
      END
      ELSE
      BEGIN
         Lat     := LatMin + ScanLength + Extra;
         NextLat := LatMin - Extra;
         LatRate := -ScanRate;
      END;
      Long       := -Size/2 + (IScan-1)*Spacing;
      NextLong   := Long + Spacing;
      LongRate   := 0.0;
   END;

   IF ScanIn = 'LONG' THEN
   BEGIN
      IF Odd(ISCan) THEN
      BEGIN
         Long     := LongMin-Extra;  (* start 1 degree before start
*)
         NextLong := LongMin + SCanLength + Extra;
         LongRate := ScanRate;
      END
      ELSE
      BEGIN
         Long :=  LongMin + ScanLength + Extra;
         NextLong := LongMin - Extra;
         LongRate := -ScanRate;
      END;
      Lat      := -Size/2 + (IScan-1)*Spacing;
      NextLat := Lat + Spacing;
      LatRate  := 0.0;
   END;

   IF IScan = 1 THEN
   BEGIN
      WriteString(Antenna,'Begin GoToScan');
      WriteString(Antenna,'Duration '+IntStr(400));
      WriteString(Antenna,'A0 '+RealStr(Long,8,3));
      WriteString(Antenna,'A1 0.0');
      WriteString(Antenna,'B0'+RealStr(Lat,8,3));
      WriteString(Antenna,'B1 0.0');
      WriteString(Antenna,'End');
   END;
   IF IScan <> 1 THEN
     IF(NSam > 0) THEN BEGIN
       WriteText(TextView,'Sample No = '+Intstr(NSam),5);
       OutScan(NSam,IScan);
     END;

   AntOnSource;
   Calibrate(FeedSys,DVM1);
         WriteString(Antenna,'Begin Scan_'+IntStr(IScan));
         WriteString(Antenna,'Clear');
         WriteString(Antenna,'Duration '+IntStr(ScanTime+5));
         WriteString(Antenna,'A0 '+RealStr(Long,8,3));
         WriteString(Antenna,'B0 '+RealStr(Lat,8,3));
         WriteString(Antenna,'A1 '+RealStr(LongRate,8,3));
         WriteString(Antenna,'B1 '+RealStr(LatRate,8,3));
         WriteString(Antenna,'End');

         WriteText (TextView,'Doing scan '+IntStr(IScan)+' of
'+IntStr(NScan),5);
         WriteText (TextView,'StartLat  =  '+RealStr(Lat,8,3)+'  StartLong
=  '
                    +RealStr(Long,8,3),5);

         IF IScan <> NScan THEN
         (* put next command into FIFO *)
         BEGIN
            WriteString(Antenna,'Begin GoToScan');
            WriteString(Antenna,'Append');
            WriteString(Antenna,'Duration '+IntStr(400));
            WriteString(Antenna,'A0 '+RealStr(NextLong,8,3));
            WriteString(Antenna,'A1 0.0');
            WriteString(Antenna,'B0'+RealStr(NextLat,8,3));
            WriteString(Antenna,'B1 0.0');
            WriteString(Antenna,'End');
         END;
         DoScan (NSam);
         IF ReadDVM = 'NO'  THEN ReadScan(NSam,ISCan);
      END;

      OutScan(NSam,NScan+1);
      WriteText(TextView,'Finished Scanning - Press any key to
continue',5);
      REPEAT
         (* wait *)
      UNTIL KeyPressed;

   END;

begin
   TextColor(White);
   TextBackground(Black);
   ClrScr;
   Write('Directory : ');
   ReadLn(DirString);
   Write('Observation Name : ');
   ReadLn(ObsName);
   ClrScr;

   DoScans;

   CloseGraph;

end.
```

program Source;

```
uses

   Graph, IO_Utils,GlobVar,
   Crt, dos,Utils,Screen ;

const

   MaxLine = 8;
   MaxCol = 23;
   XCol = 15;
   Line1 = 'Source RA      ';
   Line2 = 'Source DEC     ';
   Line3 = 'Scan Length    ';
   Line4 = 'Scan Rate      ';
   Line5 = 'No of Scans    ';
   Line6 = 'Feed System    ';
   Line7 = 'Read from DVM? ';
   Line8 = 'CoordSys       ';

   Lines:array[1..MaxLine] of string
                 = (Line1,Line2,Line3,Line4,Line5,Line6,Line7,Line8);


var
    Left,Top               : integer;
    NextLong, NextLat      : real;
    NSam, Num, Line,I,ScanNo : integer;
    LongRate,LatRate       : real;
    SourceLong,SourceLat   : real;
    Finished               : Boolean;
    Again                  : char;


   PROCEDURE WriteSourceParams;

   var
      FileName : string;
      f : text;

   BEGIN

      FileName := DirString+'\'+BaseFile+'.prm';
      Assign(f,FileName);
      ReWrite(f);
      WriteLn(f,'Source RA      ',SourceLong);
      WriteLn(f,'Source DEC     ',SourceLat);
      WriteLn(f,'Scan Length    ',ScanLength);
      WriteLn(f,'Scan Rate      ',ScanRate);
      WriteLn(f,'Feed System    ',FeedSys);
      WriteLn(f,'Read from DVM? ',ReadDVM);
      WriteLn(f,'Num            ',Num);
      WriteLn(f,'Coord System   ',CoordType);
      Close(f);

   END;

   PROCEDURE SetParams;
(*-------------------*)


   var
      Ch : char;

    PROCEDURE WriteParam(I: integer;Attr:byte);
   (*--------------------*)

   BEGIN
      TextAttr := Attr;
      GoToXY(1,I);
      ClrEol;
      Write(Lines[I]);
      GotoXY(XCol,I);
      CASE I OF
         1 : Write(SourceLong:10:2);
         2 : Write(SourceLat:10:2);
         3 : Write(ScanLength:10:2);
         4 : Write(ScanRate:10:2);
         5 : Write(Num : 10);
         6 : Write(FeedSys : 10);
         7 : Write(ReadDVM:9);
         8 : Write(CoordType:9);
      END;
   END;

   PROCEDURE EditParam(I : integer);
(*-------------------*)
   BEGIN
      TextAttr:= HighLight;
      ClrEol;
      GoToXY(XCol,Line);
      CASE I OF
         1 : Read(SourceLong);
         2 : Read(SourceLat);
         3 : Read(ScanLength);
         4 : Read(ScanRate);
         5 : REad(Num);
         6 : Read(FeedSys);
         7 : IF ReadDVM = 'YES'
                   THEN ReadDVM := 'NO'
                   ELSE ReadDVM := 'YES';
         8 : IF CoordType = 'B1950'
                   THEN CoordType := 'GALAC'
                   ELSE CoordType := 'B1950';

      END;
      WriteParam(Line,HighLight);
   END;
```

```
BEGIN
   InitParamMenu(Left,Top,MaxCol,MaxLine);
   FOR I := 1 TO MaxLine DO
      WriteParam(I,Normal);
   Line := 1;
   WriteParam(Line,HighLight);
   GoToXY(XCol,Line);
   Finished := False;
   REPEAT
      REPEAT
         Ch := ReadKey
      UNTIL Ch = #0;
      Ch := ReadKey;
      CASE Ch OF
         #60 : EditParam(Line);      (* F2 *)
         #80 : BEGIN                 (* down arrow *)
                  WriteParam(Line,Normal);
                  INC(Line);
                  IF Line > MaxLine THEN Line := 1;
                  WriteParam(Line,HighLight);
                  GoToXY(XCol,Line);
               END;
         #72 : BEGIN                 (* up arrow *)
                  WriteParam(Line,Normal);
                  Line:= Line - 1;
                  IF Line < 1 THEN Line := MaxLine;
                  WriteParam(Line,HighLight);
                  GoToXY(XCol,Line);
               END;
         #68 : Finished := True;     (* f10 *)
      END;
   UNTIL Finished
END;

PROCEDURE DoScan (var ISam: integer);
(*---------------*)
var
   y                        : real;
   IStat,MaxISam,Code       : integer;
   Long,Lat,AntStatus,Stat,Com : string;
   Error                    : Boolean;

begin
   ISam := 0;
   IStat := ReadStatus;
   Com := ' ';
   WHILE (IStat < Halted) AND (Com <> 'GoToScan') DO
   BEGIN
      ISam := ISam +1;
      ReadString(Antenna,AntStatus,Error);
      IF Error = FALSE THEN
      BEGIN
         Stat := AntField(StatField,AntStatus);
         Val(Stat,IStat,code);
         Long := AntField(LongField,AntStatus);
         Val(Long,AntLong[ISam],code);
         Lat := AntField(LatField,AntStatus);
         Val(Lat,AntLat[ISam],code);
         Com := AntField(ComField,AntStatus);
      END
      ELSE
         HandleError(AntStatus);


       IF ReadDVM = 'YES' THEN ReadReal(DVM1,Scan[ISam]);
   END;

end; (* DoScan *)

PROCEDURE GetParams;
(*-----------------*)
var
   EndLat,EndLong : real;

BEGIN
   FeedSys    := 3;
   SourceLat  := 12.667;
   SourceLong := 187.080;
   ScanLength := 5.0;
   ScanRate   := 0.1;
   Num        := 2; { no of scans in each direction }
   CoordType  := 'B1950';

   SetParams;

   NoiseTemp  := NoiseTemps[FeedSys];
   SysName    := SysNames[FeedSys];

   IF ReadDVM = 'YES' THEN WriteString(DVM1,'*S1T3');

   ScanTime := Round((ScanLength+Extra)/ScanRate);
   NScan    := Num * 2;
   WriteSourceParams;

END (*GetParams *);


PROCEDURE DoScans (dir : integer; IScan : integer);
(*---------------*)

VAR
   NData         : integer;
   Ch : char;

BEGIN
   NewColour;
   IF ScanIn = 'LAT' THEN
      BEGIN
         IF Dir = 1 THEN
         BEGIN
            Lat     := SourceLat - ScanLength/2 - Extra;
```

```
            NextLat := SourceLat + ScanLength/2 + Extra
         END
         ELSE
         BEGIN
            Lat     := SourceLat + ScanLength/2 + Extra;
            NextLat := SourceLat -ScanLength/2 -Extra
         END;
         LatRate := Dir * ScanRate;
      END;

   IF ScanIn = 'LONG' THEN
      BEGIN
         IF Dir = 1 THEN
         BEGIN
            Long     := SourceLong - ScanLength/2 - Extra;  (* start 1
degree before start *)
            NextLong := SourceLong + SCanLength/2 + Extra
         END
         ELSE
         BEGIN
            Long     := SourceLong + ScanLength/2 + Extra;
            NextLong := SourceLong - ScanLength/2 - Extra;
       END;
       LongRate := Dir * ScanRate;
      END;

   WriteText(TextView,'Driving to start of scan  '+IntStr(IScan),5);
   IF IScan <> 1 THEN
      IF(NSam > 0) THEN BEGIN
          IF IScan = 2 * Num + 1 THEN ScanIn := 'LAT';
          WriteText(TextView,Intstr(NSam),5);
          OutScan(NSam,IScan);
          WriteText(TextView,'Temp =  '+RealStr(Temp,10,7),5);
          IF IScan = 2 * Num + 1 THEN ScanIn := 'LONG';
      END;

   AntOnSource;

   WriteString(Antenna,'Begin Scan_'+IntStr(IScan));
   WriteString(Antenna,'Clear');
   WriteString(Antenna,'Duration '+IntStr(ScanTime+5));
   WriteString(Antenna,'A0 '+RealStr(Long,8,3));
   WriteString(Antenna,'B0 '+RealStr(Lat,8,3));
   WriteString(Antenna,'A1 '+RealStr(LongRate,8,3));
   WriteString(Antenna,'B1 '+RealStr(LatRate,8,3));
   WriteString(Antenna,'End');


   WriteText (TextView,'Scanning in '+ScanIn + '  dir =
'+IntStr(dir),5);
   WriteText (TextView,'StartLat  =  '+RealStr(Lat,8,3)+'  StartLong =
'
               +RealStr(Long,8,3),5);

   WriteString(Antenna,'Begin GoToScan');
   WriteString(Antenna,'Append');
   WriteString(Antenna,'Duration '+IntStr(400));
   WriteString(Antenna,'A0 '+RealStr(NextLong,8,3));
   WriteString(Antenna,'A1 0.0');
   WriteString(Antenna,'B0'+RealStr(NextLat,8,3));
   WriteString(Antenna,'B1 0.0');
   WriteString(Antenna,'End');


   DoScan (NSam);
   IF ReadDVM = 'NO'  THEN ReadScan(NSam,ISCan);


END;

procedure GoToSource;
(*-------------*)
(* sets up the commands necessary to drive to source *)

var

   SysOnPC,AntStatus : String;
   Error             : Boolean;

BEGIN
   IF ScanIN = 'LAT' THEN
   BEGIN
      Lat  := SourceLat - ScanLength/2 -Extra;
      Long := SourceLong;
   END
   ELSE
   BEGIN
      Long  := SourceLong - ScanLength/2 -Extra;
      Lat   := SourceLat;
   END;
   WriteString (Antenna,'Begin FindSrce');
   WriteString (Antenna,'Clear');
   WriteString (Antenna,'Duration 900');
   WriteString (Antenna,'Coordinat_System '+ CoordType);
   WriteString (Antenna,'Pointing On');
   WriteString (Antenna,'Feed_System '+IntStr(FeedSys));
   WriteString (Antenna,'A0 '+RealStr(Long,8,3));
   WriteString (Antenna,'A1 0.0');
   WriteString (Antenna,'B0 '+RealStr(Lat,8,3));
   WriteString (Antenna,'B1 0.0');
   WriteString (Antenna,'PSI 0.0');
   WriteString (Antenna,'PHI 0.0');
   WriteString (Antenna,'THETA 0.0');
   WriteString (Antenna,'PAR 0.0');

   WriteString (Antenna,'End');


   IF ScanIn = 'LAT' THEN
   BEGIN
      (* Check PC selected field against internal name *)
      ReadString(Antenna,AntStatus,Error);
```

```
         SysOnPc := AntField(FeedSysField,AntStatus);

         WriteText(TextView,'PC Steer Feed is  '+SysOnPC,5);
         IF SysOnPC <> SysName THEN
            WriteText(TextView,'PC Steer Feed System list is inconsistant
',5);

      END;

       AntOnSource;

       WriteText(TextView,'Antenna at start',5);

   END; (* GoToSource *)

   PROCEDURE DoLatScans (Num : integer);

   BEGIN
      ScanIn := 'LAT';
      GoToSource;

      Calibrate(DVM1);

      FOR I := 1 TO Num DO
      BEGIN
         NextLong := SourceLong;
         LatMin   := SourceLat - ScanLength/2;
(*         LatMax   := SourceLat + ScanLength/2;*)
         Long     := SourceLong;
         LongRate := 0.0;

         INC(ScanNo);
         DoScans ( 1,ScanNo);
         INC(ScanNo);
         DoScans (-1,ScanNo);
      END;
   END;

   PROCEDURE DoLongScans (Num : integer);

   BEGIN
      ScanIn := 'LONG';
      GoToSource;

      LongMin := SourceLong - SCanLength/2;
(*      LongMax := SourceLong + ScanLength/2;*)
      Lat      := SourceLat;
      LatRate  := 0.0;
      NextLat  := SourceLat;
      Calibrate(DVM1);

      FOR I := 1 TO Num DO
      BEGIN
         INC(ScanNo);
         DoScans ( 1,ScanNo);
         INC(ScanNo);
         DoScans (-1,SCanNo);

      END;
         OutScan(NSam,ScanNo+1);
      END;

begin
   Spacing   := 0.0;
   ClrScr;
   Write('Sub Dir : ');
   ReadLn(DirString);

   ClrScr;

   Finished := FALSE;
   ScanNo := 0;
   WHILE Finished = FALSE DO
   BEGIN
      Write('File name : ');
      ReadLn(BaseFile);
      GetParams;

      SetUpScreen;

      DoLatScans(Num);
      DoLongScans(Num);

      WriteText (TextView,'Repeat ? (Y/N)',5);
      ReadLn (Again);
      IF (Again = 'Y') OR (Again = 'y')
         THEN Finished := FALSE
      ELSE
         Finished := TRUE;
   END; (* DO *)
   CloseGraph;
end.
```

Unit BeamUtil;

```
interface

uses
   GlobVar, IO_Utils,Graph,Crt,Dos,Screen,Utils;

var
   SourceRA, SourceDec : real;

PROCEDURE SetParams;
PROCEDURE BeamSetUp;

implementation

const
   MaxLine = 10;
   MaxCol  = 28;
   XCol    = 15;
   Line1   = 'Source RA      ';
   Line2   = 'Source DEC     ';
   Line3   = 'Size           ';
   Line4   = 'Start Scan No  ';
   Line5   = 'End Scan No    ';
   Line6   = 'Scan Rate      ';
   Line7   = 'Scan Spacing   ';
   Line8   = 'Feed System    ';
   Line9   = 'Scan Direction ';
   Line10  = 'Read from DVM? ';
   Lines:array[1..MaxLine] of string
                  = (Line1,Line2,Line3,Line4,Line5,Line6,Line7,Line8,
                     Line9,Line10);
var

   I,Left,Top : integer;
   Finished : Boolean;

   PROCEDURE WriteBeamParams;
   (* write the parameters to a file *)

   var
      FileName : string;
      f        : text;

   BEGIN

      FileName := DirString+'\'+ObsName+'.prm';
      Assign(f,FileName);
      ReWrite(f);
      WriteLn(f,'Source RA      ',SourceRA:10:3);
      WriteLn(f,'Source DEC     ',SourceDEC:10:3);
      WriteLn(f,'Scan Length    ',Size:10:3);
      WriteLn(f,'Scan Rate      ',ScanRate:10:3);
      WriteLn(f,'Feed System    ',FeedSys);
      WriteLn(f,'Read from DVM? ',ReadDVM);
      WriteLn(f,'Scan Direction ',ScanIn);
      WriteLn(f,'Scan Spacing   ',Spacing:10:3);
      Close(f);

   END;


   PROCEDURE SetParams;
(*--------------------*)

   var
      Ch : char;
      Line : integer;

    PROCEDURE WriteParam(I: integer;Attr:byte);
   (*--------------------*)

   BEGIN
      TextAttr := Attr;
      GoToXY(1,I);
      ClrEol;
      Write(Lines[I]);
      GoToXY(XCol,I);
      CASE I OF
         1 : Write(SourceRA:15:3);
         2 : Write(SourceDec:15:3);
         3 : Write(Size:15:3);
         4 : Write(StartNo:14);
         5 : Write(EndNo:14);
         6 : Write(ScanRate:15:3);
         7 : Write(Spacing:15:3);
         8 : Write(Sys[FeedSys]:13);
         9 : Write(ScanIn:14);
         10: Write(ReadDVM:14);
      END;
   END;

   PROCEDURE EditParam(I : integer);
(*--------------------*)
   BEGIN
      TextAttr := HighLight;
      ClrEol;
      GoToXY(XCol,Line);
      CASE I OF
         1 : Read(SourceRA);
         2 : Read(SourceDec);
         3 : Read(Size);
         4 : Read(StartNo);
         5 : Read(EndNo);
         6 : Read(ScanRate);
         7 : Read(Spacing);
         8 : BEGIN
                FeedSys := FeedSys + 1;
                IF FeedSys = MaxFeedSys
```

```
                THEN FeedSys := 1;
            END;
        9 : IF ScanIn = 'LONG'
                THEN ScanIn := 'LAT'
                ELSE ScanIn := 'LONG';
        10: IF ReadDVM = 'YES'
                THEN ReadDVM := 'NO'
                ELSE ReadDVM := 'YES';
    END;
    IF ((I<>4) AND (I<>5)) THEN
       EndNo := Round(size/Spacing)+1;
    WriteParam(Line,HighLight);
END;


BEGIN
    TextBackGround(Black);
    ClrSCr;
    InitParamMenu(Left,Top,MaxCol,MaxLine,'  B E A M   M A P  ');
    FOR I := 2 TO MaxLine DO
       WriteParam(I,Normal);
    Line := 1;
    WriteParam(Line,HighLight);
    GoToXY(XCol,Line);
    Finished := False;
    REPEAT
       REPEAT
          Ch := ReadKey
       UNTIL Ch = #0;
       Ch := ReadKey;
       CASE Ch OF
          #60 : EditParam(Line);      (* F2 *)
          #80 : BEGIN                 (* down arrow *)
                  WriteParam(Line,Normal);
                  INC(Line);
                  IF Line > MaxLine THEN Line := 1;
                  WriteParam(Line,HighLight);
                  GoToXY(XCol,Line);
                END;
          #72 : BEGIN                 (* up arrow *)
                  WriteParam(Line,Normal);
                  Line:= Line - 1;
                  IF Line < 1 THEN Line := MaxLine;
                  WriteParam(Line,HighLight);
                  GoToXY(XCol,Line);
                END;
          #68 : Finished := True;   (* f10 *)

       END;
    UNTIL Finished;
    WriteBeamParams;
END;


 procedure BeamSetUp;
```

```
  (*--------------------*)
(* sets up the commands necessary to drive to source position *)

   var

      SysOnPC,AntStatus : String;
      Error             : Boolean;
      Phi               : Real;

   BEGIN
      CoordType := B1950;
      WriteString (Antenna,'Begin FindStrt');
      WriteString (Antenna,'Clear');
      WriteString (Antenna,'Duration 900');
      WriteString (Antenna,'Coordinat_System B1950');
      WriteString (Antenna,'Pointing On');
      WriteString (Antenna,'Feed_System '+IntStr(FeedSys));
      WriteString (Antenna,'A0 0.0');
      WriteString (Antenna,'A1 0.0');
      WriteString (Antenna,'B0 0.0');
      WriteString (Antenna,'B1 0.0');
      WriteString (Antenna,'PSI '+RealStr(SourceRA,8,3));
      Phi := - SourceDec;
      WriteString (Antenna,'PHI '+RealStr(Phi,8,3));
      WriteString (Antenna,'THETA 0.0');
      WriteString (Antenna,'PAR 0.0');

      WriteString (Antenna,'End');


      WriteText (TextView,'Driving to start at Long =
'+REalStr(SourceRA,8,3)
                          +' Lat = '+RealStr(SourceDec,8,3),5);

      Calibrate(FeedSys,DVM1);
      (* Check PC selected field against internal name *)
      ReadString(Antenna,AntStatus,Error);
      IF Error THEN HandleError(AntStatus);
      SysOnPc := AntField(FeedSysField,AntStatus);

      WriteText(TextView,'PC Steer Feed is  '+SysOnPC,5);
      IF SysOnPC <> SysName THEN
          WriteText(TextView,'PC Steer Feed System list is inconsistant
',5);

       AntOnSource;

       WriteText(TextView,'Antenna at start',5);

   END; (* SetUp *)

begin


end.
```

unit GlobVar;

```
interface

const

   MaxData = 1500;              (* Max no of data points *)
   Extra   = 1.0;               (* extra degrees at start of scan *)

type

   Scans    = Array[1..MaxData] of real;

var

   YScale, XScale, Temp : Real;
   TitleStr             : string;
   ObsName,DirString    : String;
   ScaleScan            : Boolean;
   Scan,AntLat,AntLong,Coord : Scans;
   PlotColour                : integer;
   Clear,Grid,Labeled : Boolean;
   ReadDVM                   : String;
   CoordType                 : String;
   ScanIn                    : String;
   ScanLength,Size,Spacing   : real;
   ScanTime,NScan, Ndata,StartNo,EndNo          : integer;
   LatMin,LongMin,ScanRate   : real;
   Long,Lat,NoiseDVM : real;
   FeedSys                   : Integer;
   SysName                   : String;
   NoiseTemp                 : real;

(*-------------------------------------------------------------------------
-*)
implementation


begin
   Clear      := FALSE;
   Labeled    := FALSE;
   ScaleScan  := TRUE;
   Grid       := FALSE;
   TitleStr   := 'SCAN  ';
   PlotColour := 9; (* LightBlue *)
   Temp       := 0.0;
   ReadDVM    := 'NO';

end.
```

unit IO_Utils;

```
interface
uses

   Graph,Crt, dos,GlobVar;

type
   ValidFeedSystem = set of 0..9;

const

   OnSource      =    0;
   Driving       =    1;                    (* status codes *)
   Halted        =    2;

   StatField     =    1;                    (* fields in antenna *)
   ComField      =    2;                    (*  reply message *)
   TimeField     =    4;
   MJDField      =    5;
   FeedSysField  =    6;
   HAOffField    =    7;
   DecOffField   =    8;
   LongField     =   17;
   LatField      =   18;
   UsrLongField  =   21;
   UsrLatField   =   22;

   MaxFeedSys    = 9;
   Num           = 90;
   SysNames: array [1..MaxFeedSys] of string
         = ('2.5cm single','3.6cm dicke ','3.6cm single','4.5cm single',
            '6cm dicke   ','6cm single  ','13cm single ','18cm        ',
            '18cm untilt ');
   Sys :     array [1..MaxFeedSys] of string
         = ('2.5 S    ','3.6 D    ','3.6 S    ','4.5 S    ',
            '6 D      ','6 S      ','13 S     ','18       ',
            '18 U     ');
   NoiseTemps : array [1..MaxFeedSys] of real
         = (158.0,10.0,10.0,82.0,1.54,16.0,3.96,33.0,33.0);
   NoiseDiode    =    1;
   ValidFeedSys : ValidFeedSystem = [3,6];
   pc14base = $1b0;

var

   Antenna, DVM1,DVM2 : integer;

   PROCEDURE AntOnSource;
{  continually reads antenna status until antenna is on source }

   PROCEDURE Calibrate(FeedSys,DVM_No : integer);
{ perform noise diode calibration reading from DVM_No }

   FUNCTION  IntStr(i:longint) :string;
{  returns i ( longint ) as a string }
   FUNCTION  RealStr(x:real;w,d:integer) :string;
{  returns x ( a real w:d) as a string }

   FUNCTION  ReadStatus :integer;
{ returns the antenna status code }

   PROCEDURE WriteString (device : integer; Comm:string);
{ writes the string Comm to device }
   PROCEDURE ReadString(device : integer;var s:string;var Error :
Boolean);
{ reads a string from a device setting error flag }
   PROCEDURE ReadReal (device : integer; var x: real);
{ reads a real from a device }

   FUNCTION  AntField(Field : integer; s:string) :string;
{ returns the contents of a Field of the reply message }

   PROCEDURE SelectDVM(i: integer);
{ selects DVM no i }


(*-------------------------------------------------------------------------
-*)
implementation

uses
   TPDECL , Utils;

var

   un,bn : nbuf;
   bd    : integer;
   I,ErrorCount : integer;
   NoiseDiodeNo : array[1..MaxFeedSys] of integer;

type

    commbuffer = array[1..80] of char;



  PROCEDURE ErrorCheck(var s:string;var Error:Boolean);
 (*--------------------*)

   BEGIN
      IF (IBSTA AND ERR) <> 0 THEN
      BEGIN
         Inc(ErrorCount);
         Error := True;
         s := 'Error :   '+IntStr(IBERR) +
                  '       ErrorCount :    ' + IntStr(ErrorCount);
      END;

   END (*ErrorCheck *);
```

```
   PROCEDURE SelectDVM (i: integer);
(* -------------------*)
   BEGIN

   END;

   PROCEDURE NoiseDiodeOn  (i: integer);
(*-----------------------*)

   BEGIN
      Port[pc14base+1] := i;
      Port[pc14base+0] := 5;
      Port[pc14base+0] := 0;
      delay (1000);
   END; (* NoiseDiodeOn *)

   PROCEDURE NoiseDiodeOff;
(*-----------------------*)
   BEGIN

    Port[pc14base+1] := 0;
    Port[pc14base+0] := 5;
    Port[pc14base+0] := 0;
    delay (1000);
   END;



   PROCEDURE ReadString(device : integer;var s:string;var Error :
Boolean);
 (*--------------------*)
 (* reads a string from device into the string s *)

   var
      Buffer : array[1..255] of char;
      I      : integer;


   BEGIN
      Error := FALSE;
      IBRD(device,Buffer,255);
      ErrorCheck(s,Error);
      IF Error = FALSE THEN
      BEGIN
         s := '';
         FOR I := 1 TO IBCNT DO   (* IBCNT hold the length of the string
*)*
             IF ((Buffer[I] <> #10) AND (Buffer[I]<> #13)) THEN
                s := s + Buffer[I];
      END;
      (* if error then s contains error message *)

   END;  (* ReadString *)
```

```
   FUNCTION ReadStatus:integer;
 (*--------------------*)
(* returns the status of the antenna *)

   var
      i,code : integer;
      s,stat : string;
      Error  : Boolean;

   BEGIN
      ReadString(Antenna,s,Error);
      IF Error = FALSE THEN
      BEGIN
         stat := COPY(s,1,1);
         VAL(stat,i,code);
         ReadStatus := i;
      END;

   END;

   PROCEDURE AntOnSource;
 (*--------------------*)

   var
      Status : integer;

   BEGIN
      REPEAT
         Status := ReadStatus;
      UNTIL Status <> Driving;
      IF (Status <> OnSource) AND (Status <> Halted)
         THEN StatusError(Status);
   END;

   PROCEDURE ReadReal (device : integer; var x: real);
(*--------------------*)
   var
      s     : string;
      Error : Boolean;
      code  : integer;

   BEGIN
      ReadString(device,s,error);
      IF Error = FALSE THEN Val(s,x,code);
      IF code <> 0 THEN Error := TRUE;

   END;



    PROCEDURE WriteString (device : integer; Comm:string);
   (*--------------------*)

    var
```

```
      Buff  : CommBuffer;
      Error : Boolean;
      S     : string;

   begin

      FOR I := 1 TO LENGTH(Comm) DO
         Buff[I] := Comm[I];

         IBWRT(device,Buff,Length(Comm));
         ErrorCheck(s,Error);
         Comm := '';

   end; (* WriteString *)

  FUNCTION IntStr(i:longint) :string;
 (*--------------*)

  var
     s : string[11];

  BEGIN

     Str(i, s);
     IntStr := s
  END;


  FUNCTION AntField(Field : integer; s:string) : string;
(*---------------------*)

  var
     code : integer;
     x    : string;

  BEGIN
     CASE Field OF

        StatField:      x := COPY(s,1,1);
        ComField:       x := COPY(s,3,8);
        TimeField:      x := COPY(s,19,9);
        MJDField:       x := COPY(s,28,13);
        HAOffField:     x := COPY(s,53,6);
        DecOffField:    x := COPY(s,59,6);
        LatField:       x := COPY(s,127,8);
        LongField :     x := Copy(s,119,8);
        FeedSysField :  x := Copy(s,41,12);
        UsrLongField:   x := Copy(s,151,8);
        UsrLatField:    x := Copy(s,159,8);

     END;
     AntField := x;

  END;

  FUNCTION RealStr(x:real;w,d:integer) :string;
  (*---------------*)
   var
      s : string[11];

   BEGIN

      Str(x:w:d, s);
      RealStr := s
   END;

   PROCEDURE Calibrate(FeedSys,DVM_No : integer);

   var
      NoiseOn, NoiseOff,N : Real;

   BEGIN
      IF ReadDVM = 'YES' THEN
      BEGIN
         NoiseOn  := 0.0;
         NoiseOff := 0.0;
         NoiseDiodeOff;
         FOR I := 1 TO Num DO
         BEGIN
            ReadReal(DVM_No,N);
            NoiseOff := NoiseOff + N;
         END;
         NoiseDiodeOn(NoiseDiodeNo[FeedSys]);
         FOR I := 1 TO 2*Num DO
         BEGIN
            ReadReal(DVM_No,N);
            NoiseOn := NoiseOn + N;
         END;
         NoiseDiodeOff;
         FOR I := 1 TO Num DO
         BEGIN
            ReadReal(DVM_No,N);
            NoiseOff := NoiseOff + N;
         END;
         NoiseDVM := (NoiseOn - NoiseOff)/(2*Num);
(*         WriteText(TextView,'Noise Voltage = ' +
RealStr(NoiseDVM,15,10),5);*)
      END;
   END;


(*-------------------------------------------------------------------------
-*)
begin

   un      := 'ant    ';
   Antenna := IBFIND (un);
   bn      := 'GPIB0  ';
   bd      := IBFIND (bn);
   ibdma (bd,1);
   un     := 'DVM1   ';
   DVM1   :=  IBFIND (un);
```

```
(* un      := 'DVM2    ';
   DVM2    := IBFIND (un);*)

   NoiseDiodeNo[3] := 1;
   NoiseDiodeNo[6] := 16;

  {initialization of 8255 card}
   Port[pc14base+3] := $80;
   Port[pc14base+0] := 0;

end.
```

Unit MapUtils;

```
interface

uses
   Graph,Screen,GlobVar,Crt,Utils;


   PROCEDURE SetParams(var EndLat,EndLong : real);
   PROCEDURE SetUp;

(*-------------------------------------------------------------------------
-*)
implementation

uses
   IO_Utils;

const
   MaxLine   = 12;
   MaxCol    = 23;
   XCol      = 15;
   Line1 =  'StartLat       ';
   Line2 =  'StartLong      ';
   Line3 =  'EndLat         ';
   Line4 =  'EndLong        ';
   Line5 =  'ScanRate       ';
   Line6 =  'ScanSpacing    ';
   Line7 =  'Start Scan No  ';
   Line8 =  'End Scan No    ';
   Line9 =  'Feed Sys       ';
   Line10 = 'ScanIn         ';
   Line11 = 'Read from DVM? ';
   Line12 = 'Coord Sys      ';

   Lines:array[1..MaxLine] of string
           = (Line1,Line2,Line3,Line4,Line5,Line6,Line7,Line8,Line9,
              Line10,Line11,Line12);

var
   I,Left,Top : integer;
   Finished : Boolean;
   Line : integer;

    procedure SetUp;
  (*-------------*)
   (* sets up the commands necessary to drive to Long and Lat *)

   var

      SysOnPC,AntStatus : String;
      Error             : Boolean;

   BEGIN
      IF ReadDVM = 'YES' THEN WriteString(DVM1,'*S1T3');
      WriteString (Antenna,'Begin FindStrt');
      WriteString (Antenna,'Clear');
      WriteString (Antenna,'Duration 900');
      WriteString (Antenna,'Coordinat_System '+CoordType);
      WriteString (Antenna,'Pointing On');
      WriteString (Antenna,'Feed_System '+IntStr(FeedSys));
      WriteString (Antenna,'A0 '+RealStr(LongMin,8,3));
      WriteString (Antenna,'A1 0.0');
      WriteString (Antenna,'B0 '+RealStr(LatMin,8,3));
      WriteString (Antenna,'B1 0.0');
      WriteString (Antenna,'PSI 0.0');
      WriteString (Antenna,'PHI 0.0');
      WriteString (Antenna,'THETA 0.0');
      WriteString (Antenna,'PAR 0.0');

      WriteString (Antenna,'End');

      WriteText (TextView,'Driving to start at Long =
'+RealStr(LongMin,8,3)
                          +'  Lat = '+RealStr(LatMin,8,3),5);

      (* Check PC selected field against internal name *)
      ReadString(Antenna,AntStatus,Error);
      IF Error THEN HandleError(AntStatus);
      SysOnPc := AntField(FeedSysField,AntStatus);
      WriteText(TextView,'PC Steer Feed is  '+SysOnPC,5);
      IF SysOnPC <> SysName THEN
          WriteText(TextView,'PC Steer Feed System list is inconsistant
',5);

      AntOnSource;
      WriteText(TextView,'Antenna at start',5);

   END; (* SetUp *)

   PROCEDURE WriteParamFile (var EndLat, EndLong : real);

   var
        f : text;
        FileName : String;

   BEGIN
        FileName := DirString+'\'+ObsName+'.prm';
        Assign(f,FileName);
        ReWrite(f);
        WriteLn(f,'LatMin',LatMin:10:2);
        WriteLn(f,'LongMin',LongMin:10:2);
        WriteLn(f,'EndLat',EndLat:10:2);
        WriteLn(f,'EndLong',EndLong:10:2);
        WriteLn(f,'Scan Rate',ScanRate:10:2);
        WriteLn(f,'Spacing',Spacing:10:2);
        WriteLn(f,'StartNo',StartNo:9);
        WriteLn(f,'EndNo',EndNo:9);
        WriteLn(f,'Feed System',FeedSys:9);
```

```
      WriteLn(f,'ScanIn',ScanIn:9);
      WriteLn(f,'ReadDVM',ReadDVM:9);
      WriteLn(f,'CoordType',CoordType:9);
      Close(f);
  END;

  FUNCTION CheckError(EndLat,EndLong: real) : Boolean;
  var
    error : boolean;
    L,T,X1,X2,Y1,Y2 : integer;
    Str : string;
  BEGIN
     Error := FALSE;
     Error := NOT ((LatMin < EndLat) AND (LongMin < EndLong) AND
        (StartNo <= EndNo) AND (FeedSys IN ValidFeedSys));
     CheckError := Error;
     IF Error THEN
     BEGIN
        X1 := LO(WindMin);
        X2 := LO(WindMax);
        Y1 := HI(WindMin);
        Y2 := HI(WindMax);

        IF (LatMin > EndLat) THEN
           Str := 'START LAT < END LAT'
        else
           if (LongMin > EndLong) THEN
              Str := 'START LONG < START LAT'
              ELSE
                 IF (StartNo > EndNo) THEN
                    Str := 'START SCAN > END SCAN'
                    ELSE
                       Str := 'INVALID FEED SYS';
        ErrorWindow(Str);

        Window(1,1,80,25);
      END;
  END;

  PROCEDURE SetParams(var EndLat,EndLong : real);
(*-------------------*)


  var
     Ch : char;

   PROCEDURE WriteParam(I: integer;Attr:byte);
  (*--------------------*)

  BEGIN

     TextAttr := Attr;
     GoToXY(1,I);
     ClrEol;
     Write(Lines[I]);
     CASE I OF
        1 : Write(LatMin:9:2);
        2 : Write(LongMin:9:2);
        3 : Write(EndLat:9:2);
        4 : Write(EndLong:9:2);
        5 : Write(ScanRate:9:2);
        6 : Write(Spacing:9:2);
        7 : Write(StartNo:9);
        8 : Write(EndNo:9);
        9 : Write(Sys[FeedSys]:8);
       10 : Write(ScanIn:8);
       11 : Write(ReadDVM:8);
       12 : Write(CoordType:8);
     END;
  END;

  PROCEDURE UpDateScanNo;

  BEGIN
     IF ScanIn = 'LAT' THEN
        EndNo := Round((EndLong-LongMin)/Spacing) + 1
     ELSE
        EndNo := Round((EndLat-LatMin)/Spacing) + 1;
     WriteParam(8,Normal);

  END;

  PROCEDURE EditParam(I : integer);
(*-------------------*)
  BEGIN
     TextAttr:= HighLight;
     ClrEol;
     GoToXY(XCol,Line);
     CASE I OF
        1 : Read(LatMin);
        2 : Read(LongMin);
        3 : Read(EndLat);
        4 : Read(EndLong);
        5 : Read(ScanRate);
        6 : Read(Spacing);
        7 : Read(StartNo);
        8 : Read(EndNo);
        9 : BEGIN
               FeedSys := FeedSys + 1;
               IF FeedSys = MaxFeedSys
                  THEN FeedSys := 1;
            END;
       10 : IF ScanIn = 'LONG'
                  THEN ScanIn := 'LAT'
                  ELSE ScanIn := 'LONG';
       11 : IF ReadDVM = 'YES'
                  THEN ReadDVM := 'NO'
                  ELSE ReadDVM := 'YES';
       12 : IF CoordType = 'GALA'
                  THEN CoordType := 'B1950'
                  ELSE IF CoordType = 'B1950'
                           THEN CoordType := 'J2000'
```

```
                        ELSE CoordType := 'GALA' ;
      END;                                                              BEGIN
      IF ((I<>7) AND (I<>8)) THEN
         UpDateScanNo;                                                  END.
      WriteParam(Line,HighLight);
   END;


   BEGIN
      TextBackGround(Black);
      ClrScr;
      InitParamMenu(Left,Top,MaxCol,MaxLine,'  P C  M A P  ');
      FOR I := 1 TO MaxLine DO
         WriteParam(I,Normal);
      Line := 1;
      WriteParam(Line,HighLight);
      GoToXY(XCol,Line);
      Finished := False;
      REPEAT
         REPEAT
            Ch := ReadKey
         UNTIL Ch = #0;
         Ch := ReadKey;
         CASE Ch OF
            #60 : EditParam(Line);      (* F2 *)
            #80 : BEGIN                 (* down arrow *)
                    WriteParam(Line,Normal);
                    INC(Line);
                    IF Line > MaxLine THEN Line := 1;
                    WriteParam(Line,HighLight);
                    GoToXY(XCol,Line);
                  END;
            #72 : BEGIN                 (* up arrow *)
                    WriteParam(Line,Normal);
                    Line:= Line - 1;
                    IF Line < 1 THEN Line := MaxLine;
                    WriteParam(Line,HighLight);
                    GoToXY(XCol,Line);
                  END;
            #68 :  IF NOT CheckError(EndLat,EndLong) THEN
                      Finished := True   (* f10 *)
                   ELSE
                   BEGIN
                     InitParamMenu(Left,Top,MaxCol,MaxLine,'  P C  M A P
');

                     FOR I := 1 TO MaxLine DO
                     WriteParam(I,Normal);
                     Line := 1;
                     WriteParam(Line,HighLight);
                     GoToXY(XCol,Line);
                   END;
         END;
      UNTIL Finished;
      WriteParamFile(EndLat,EndLong);
   END;
```

unit Screen;

```
interface

uses
   Graph,Dos,Crt,GlobVar;

type
   ViewPort  = record
                  View       : ViewPortType;
                  YPos       : integer;
                  MaxY       : integer;
               end;
const

   HighLight = White + Red * 16;
   Normal    = Yellow + Blue * 16;

var
   OldView                        : ViewPortType;
   GraphView, TextView, KeyView : ViewPort;
   BoxY : integer;
   PROCEDURE ErrorWindow(Str : string);
   PROCEDURE LabelAxis;
   PROCEDURE DrawGrid;
   PROCEDURE SetUpScreen;
   PROCEDURE WriteText (var window: ViewPort;s: string;XPos : integer);
   PROCEDURE OpenViewPort(var Window:ViewPort);
   PROCEDURE PlotScan (NData, IScan :integer);
{ plot scan with NData points }
   PROCEDURE SetView(var XView : ViewPort;Left,Top,Right,
                     Bottom : integer; XClip: Boolean; YPos : integer);
   PROCEDURE NewColour;
{ change plot colour }

   PROCEDURE InitParamMenu(var Left,Top : integer ;MaxCol,MaxLine :
integer;
                           Str:string);
   PROCEDURE KeyMenu;
   PROCEDURE DrawBorder(Left,Top,Width,Height : integer);
(*-------------------------------------------------------------------------
-*)
implementation

uses
   IO_Utils;

var
   I : integer;

   PROCEDURE DrawBorder(Left,Top,Width,Height : integer);

   var
      I,J : integer;

   BEGIN
      Window(Left-2,Top+1-1,Left+Width+2,Top+Height+1);
      ClrScr;
      Write(' ');Write(#214); FOR I := 1 TO (Width+1) DO Write(#196);
Write(#183);Write(' ');
      FOR I := 2 TO (Height+1) DO
      BEGIN
         Write(' ');Write(#186); FOR J := 1 TO (Width+1) DO Write(' ');
Write(#186);Write(' ');
      End;
      Write(' ');Write(#211); FOR I := 1 TO (Width+1) DO Write(#196);
Write(#189);
      GoTOXY(1,1);
      Window(Left,Top+1,Left+Width,Top+Height);
      ClrScr;

   end;

   PROCEDURE TitleBox(YPos : integer; Str : string);
   var
   XPos : integer;
   OldAttr : byte;

   BEGIN
      OldAttr := TextAttr;
      XPos := TRUNC((80 - Length(Str))/2);
      GoToXY(XPos,YPos);
      TextAttr := Yellow + Blue * 16;
      Write(Str);
      TextAttr := OldAttr;
   END;


   PROCEDURE KeyMenu;
 (*-----------------------*)
   var
      L , T : integer;
   BEGIN
      TextAttr := Normal;
      L := 3; (*Trunc((Lo(WindMax)-1-16)/2);*)
      T := TRUNC((Hi(WindMax)-3)/2);
      Window(L,t+1,L+16,T+2);
      ClrScr;
      DrawBorder(L,T,16,2);
      GoToXY(1,1);
      Write('F2  EDIT FIELD');
      GoToXY(1,2);
      Write('F10 SAVE PARAM');
      Window(1,1,80,25);
   END;

   PROCEDURE ErrorWindow( Str : string);

   var Left,Top : integer;
   BEGIN
```

```
        Left := Trunc((80-1-25)/2);
        Top := 25-4;
        Window(Left,Top,Left+25,Top+2);
        TextBackground(Red);
        TextColor(White);
        DrawBorder(Left,Top,25,2);
        GoToXY(1,1);
        Write(Str);
        GoToXY(1,2);
        Write('PRESS ANY KEY TO CONTINUE');
        REPEAT

        UNTIL KeyPressed;
        Window(1,1,80,25);
        TextBackground(Black);

        ClrScr;
   END;
   PROCEDURE InitParamMenu(var Left,Top : integer ;
                           MaxCol,MaxLine : integer; Str : string);
 (*-----------------------*)

   BEGIN
      KeyMenu;
      TextAttr := Normal;
      Left := Trunc((Lo(WindMax)-1-MaxCol)/2);
      Top := Trunc((Hi(WindMax)-1-MaxLine)/2);
      TitleBox(Top - 3,Str);
      Window(Left,Top+1,Left+MaxCol,Top+MaxLine);
      ClrScr;
      DrawBorder(Left,Top,MaxCol,MaxLine);
   END;



   PROCEDURE NewColour;

   BEGIN
      INC(PlotColour);
      IF PlotColour = White THEN
      BEGIN
         PlotColour := LightBlue;
      END;
   END;

    PROCEDURE WriteText (var window: ViewPort;s: string;XPos : integer);
  (*-------------------*)
(* writes the string s in the text window at position XPos,Ypos *)

   VAR
      Height : integer;
      OldView : ViewPortType;

   BEGIN
      GetViewSettings(OldView);
      WITH Window DO
      BEGIN
         WITH View DO
            SetViewPort(X1,Y1,X2,Y2,Clip);

         Height := TextHeight(s)+2;
         IF (YPos + Height) >= BoxY/2 THEN
         BEGIN
            ClearViewPort;
            YPos:= 5;
         END;
         OutTextXY(XPos,YPos,s);
         YPos := YPos + Height;
      END;
      WITH OldView DO SetViewPort(X1,Y1,X2,Y2,Clip);

   END;

   PROCEDURE WriteParamFile (var EndLat, EndLong : real);

   var
        f : text;
        FileName : String;

   BEGIN
        FileName := DirString+'\'+ObsName+'.prm';
        Assign(f,FileName);
        ReWrite(f);
        WriteLn(f,'LatMin',LatMin:10:2);
        WriteLn(f,'LongMin',LongMin:10:2);
        WriteLn(f,'EndLat',EndLat:10:2);
        WriteLn(f,'EndLong',EndLong:10:2);
        WriteLn(f,'Scan Rate',ScanRate:10:2);
        WriteLn(f,'Spacing',Spacing:10:2);
        WriteLn(f,'StartNo',StartNo:9);
        WriteLn(f,'EndNo',EndNo:9);
        WriteLn(f,'Feed System',FeedSys:9);
        WriteLn(f,'ScanIn',ScanIn:9);
        WriteLn(f,'ReadDVM',ReadDVM:9);
        WriteLn(f,'CoordType',CoordType:9);
        Close(f);
   END;

   PROCEDURE SetUpGraphics;
  (*---------------------*)
   var
      GraphDriver : integer;
      GraphMode   : integer;
      ErrorCode   : integer;

   begin
      GraphDriver := Detect;

(*       InitGraph(GraphDriver,GraphMode,'z:\usr\limited\tp\bgi');*)

      InitGraph(GraphDriver,GraphMode,'c:\tp');
```

```
      ErrorCode := GraphResult;
      If ErrorCode <> GrOK then
      begin
         WriteLn('Graphics error;', GraphErrorMsg(ErrorCode));
         WriteLn('Program aborted!');
         Halt(1);
      end;
   end;

   PROCEDURE LabelAxis;
 (*------------------*)
   var
      LabelStr : string;
      XLabel,XScale : real;
      N, col,dX  : integer;

   BEGIN

      N := TRUNC(ScanLength/0.5)+1;

      SetViewPort(0,0,GetMaxX,GetMaxY,Clipoff);
      XScale := (GetMaxX-4)/ScanLength;

      FOR I := 1 TO N DO
      BEGIN
         IF ScanIn = 'LAT'
            THEN XLabel := LatMin + (I-1)*0.5;
         IF ScanIn = 'LONG'
            THEN XLabel := LongMin + (N-I)*0.5;
         LabelStr := RealStr(Xlabel,2,1);
         dX := ROUND((Length(LabelStr)*TextWidth('2'))/2);
         Col := TRUNC(((I-1)*0.5)*XScale);
         IF I = 1 THEN Col := dX;
         IF I = N THEN Col := GetMaxX-dX;
         OutTextXY(Col-dX,BoxY+12,LabelStr);
      END;
      WITH GraphView.View DO SetViewPort(X1,Y1,X2,Y2,Clipoff);
      Labeled := TRUE;
   END;

   PROCEDURE DrawGrid;
 (*------------------*)
   var
      R, XLabel, Scale : real;
      NDiv, I, Col         : integer;

   BEGIN

      NDiv := TRUNC(ScanLength/0.5)+1;
      WITH GraphView.View DO SetViewPort(X1,Y1,X2,Y2,Clipoff);
      FOR I := 1 TO NDiv DO
      BEGIN
         Scale := (GetMaxX-4)/ScanLength;
         Col := TRUNC(((I-1)*0.5)*Scale+2);
         MoveTo(Col,0);
         LineTo(Col,BoxY);
      END;
      Grid := TRUE;
   END;

   PROCEDURE OpenViewPort(var Window : ViewPort);
 (*----------------------*)

   BEGIN
      WITH Window DO
      BEGIN
         WITH View DO
            SetViewPort(X1,Y1,X2,Y2,Clip);
         YPos := 1;
      END;
      ClearViewPort;
   END;

   PROCEDURE SetView(var XView : ViewPort;Left,Top,Right,
                     Bottom : integer; XClip: Boolean; YPos : integer);

   BEGIN
      WITH XView DO
      BEGIN
         With View DO
         BEGIN
            X1 := left;
            Y1 := top;
            X2 := right;
            Y2 := bottom;
            Clip := XClip;
         END;
         YPos := 1;
      END;
   END;



   PROCEDURE SetUpScreen;
 (*----------------------*)

   var
      BoxX       : integer;

   BEGIN

      SetUpGraphics;
      SetGraphMode(GetGraphMode);
      SetBkColor(Blue);
      SetColor(White);
      ClearViewPort;
      BoxX := GetMaxX-4;
      BoxY := 2*trunc((GetMaxY-20)/3)-4;
      SetView(GraphView,2,2,BoxX+2,BoxY+2,ClipOff,1);
      SetView(TextView,2,BoxY+22,445,GetMaxY-2,Clipoff,1);
      SetView(KeyView,450,BoxY+22,BoxX+2,GetMaxY-2,ClipOff,1);
      Rectangle(0,0,BoxX+4,BoxY+4);
```

```
      Rectangle(0,BoxY+20,BoxX+4,GetMaxY);
      MoveTo(448,BoxY+20);
      LineTo(448,GetMaxY);
      SetColor(White);

   END;


   procedure PlotScan (NData, IScan :integer);
  (*----------------*)


     var
      I,Points : integer;
      MaxYData,MinYData : real;
      X,Y,MaxX,MaxY,Count  : integer;
      CoordMax,CoordMin : real;



     FUNCTION Max(N : integer) :real;
  (*------------*)
   var
      Temp : real;
      I : integer;
   BEGIN
      Temp := Scan[Round(N/2)];
      FOR I := 1 TO N DO
            IF (Coord[I] < CoordMax) AND (Coord[I] > CoordMin) THEN
         IF Scan[I] > Temp THEN Temp := Scan[I];
      Max := Temp;
   END; (* Max *)


   FUNCTION Min(N : integer) :real;
 (*------------*)
   var
      Temp : real;
      I : integer;
   BEGIN
      Temp := Scan[Round(N/2)];
      FOR I := 1 TO N DO
         IF (Coord[I] < CoordMax) AND (Coord[I] > CoordMin) THEN
            IF Scan[I] < Temp THEN Temp := Scan[I];
      Min := Temp;
   END; (* Min *)



var
  d : integer;
  coordstr : string;
  LeftCoord,DEg : real;

   begin

      SetColor(PlotColour);

      IF NData > 5 THEN
      BEGIN
      WITH GraphView.View DO
         SetViewPort(X1,Y1,X2,Y2,Clip);
      MaxX := GetMaxX-4;
      MaxY := BoxY - 10;
      XScale := MaxX/ScanLength;


      IF ScanIn = 'LAT' THEN
         BEGIN
            CoordMin := LatMin;
            CoordMax := LatMin + ScanLength;
            d := 1;
            LeftCoord := CoordMin;
         END;
      IF ScanIn = 'LONG' THEN
         BEGIN
            CoordMin := LongMin;
            CoordMax := LongMin + ScanLength;
            d := -1;
            XScale := -XScale;
            LeftCoord := CoordMax;
         END;
         IF ScanIn = 'LAT'  THEN CoordStr := RealStr((Long-Spacing),6,2);
         IF ScanIn = 'LONG' THEN CoordStr := RealStr((Lat-Spacing),6,2);


      MaxYData := Max(NData);
      MinYData := Min(NData);
      IF ReadDVM  = 'YES'  THEN
          Temp := (MaxYData - MinYData) * NoiseTemp/NoiseDVM;
      IF TitleStr = 'TRACK ' THEN
         Deg := (MaxYData - MinYData);
      IF MaxYData = MinYData THEN WriteText(TextView,'Error in
plotting',5)
      ELSE
      BEGIN
         IF TitleStr = 'TRACK ' THEN
            WriteText(KeyView,REalStr(Deg,5,4)+#248+' '+
                                    TitleStr+IntStr(IScan)+' '+
CoordStr,1)
         ELSE
            WriteText(KeyView,RealStr(Temp,5,2)+' K '+
                           TitleStr+IntStr(IScan)+' '+ CoordStr,1);

      IF ScaleScan OR Clear THEN YScale := -MaxY/(MaxYData-MinYData);
      Points := 0;

      For I:= 1 to NData do
      begin
         Count := I;
         IF ((Coord[Count] <= CoordMax) and (Coord[Count] >= CoordMin))
THEN
```

```
        BEGIN
            X := Trunc((Coord[Count] - LeftCoord)*(XScale))+2;
            Y := round((Scan[Count]-MinYData) * YScale + MaxY+5);
        IF (X > 1) and (X < GetMaxX) THEN
        BEGIN
           Inc(Points);
           IF  Points = 1
              THEN
                 BEGIN
                    PutPixel (X,Y,PlotColour);
                    MoveTo (X,Y);
                 END
              ELSE LineTo(X,Y)
           end;
        END;
        END;
    END;
    IF Clear THEN
     BEGIN
        ClearViewPort;
        Grid  := FALSE;
        Clear := FALSE;
        WITH KeyView.View DO SetViewPort(X1,Y1,X2,Y2,Clipoff);
        ClearViewPort;
        WITH TextView.View DO SetViewPort(X1,Y1,X2,Y2,Clipoff);
     END;
     IF Grid = FALSE THEN
     BEGIN
       DrawGrid;
     END;
     IF Labeled = FALSE THEN
     BEGIN
       LabelAxis;
    END;
    END;
    TitleStr := 'SCAN  ';

  end; (* PlotScan *)


(*-------------------------------------------------------------------------
-*)

begin


end.
```

unit Utils;

```
interface

uses

   Graph, Screen,GlobVar,
   Crt, dos;

   PROCEDURE OutScan (N,IScan : integer);
{ output scan no IScan }
   PROCEDURE PutFlHead (var f : text;IScan : integer);
{ write file header to file f }
   PROCEDURE HandleError(s : string);
{ handle GPIB error }
   PROCEDURE StatusError(stat : integer);
{ write error message and terminate }
   PROCEDURE ReadScan (N: integer;IScan : integer);
{ set up dummy scan in simulator mode }

(*-------------------------------------------------------------------------
-*)
implementation

uses
   IO_Utils;

Const
   Num = 90;

var

   I : integer;

   PROCEDURE ReadScan (N: integer;IScan : integer);
 (*------------------*)
    var
       I : integer;
       P : integer;

    BEGIN
       P := IScan*2+N;
       FOR I:= 1 TO N DO

          Scan[I] := SIN(2*3.14*I/N+P)

    END;

   PROCEDURE StatusError(stat : integer);

    BEGIN
       SetBkColor(Red);
       CASE Stat OF
       3 : WriteText(TextView,'COMMAND LIMITS',25);
       4 : WriteText(TextView,'ANTENNA NOT ONLINE',25);
       5 : WriteText(TextView,'SYSTEM ERROR',25);
       6 : WriteText(TextView,'SYSTEM PAUSED',25);
       8 : WriteText(TextView,'STOP SYSTEM',25);
       END;
       Halt(1);
    END;
   PROCEDURE HandleError(s : String);

   BEGIN

   END;




   PROCEDURE PutFlHead (var f : text;IScan : integer);

   var

      MJD,St,HAOff,DEcOff,AntStatus     : string;
      Error : Boolean;

   BEGIN
      ReadString(Antenna,AntStatus,Error);
      MJD := AntField(MJDField,AntStatus);
      ST := AntField(TimeField,AntStatus);
      HAOff := AntField(HAOffField,AntStatus);
      DEcOff := AntField(DECOffField,AntStatus);

      WriteLn(f,'MJD number ',MJD,'  Local Mean Sidereal Time ',ST,'
Coord Type ',CoordType);
      WriteLn(f,'HA Beam Offset ',HAOff,'   Dec Beam Offset ',DECOff,'
Feed Sys No',
                FeedSys,'  ',SysName);
      WriteLn(f,' Scan ',' ','NScan','  ','Noise DVM','     ','Scan in
');
      WriteLn(f,IScan:5,' ',NScan:5,' ',NoiseDVM:10:5,' ',ScanIn:9);
      IF ScanIn = 'LAT' THEN
      BEGIN
         WriteLn(f,'Start Lat ','  ','End Lat  ','    ','Long ','
','Spacing',
                 '  ','ScanRate');
         WriteLn(f,LatMin:10:3 ,' ',(LatMin + ScanLength):8:3,' ',
                 (Long-Spacing):8:3,'    ',Spacing:8:3,'
',ScanRate:8:3);
      END
      ELSE
      BEGIN
         WriteLn(f,'Start Long ','  ','End Long','    ','Lat ','
','Spacing',
                 '  ',ScanRate);
         WriteLn(f,LongMin:10:3 ,' ',(LongMin + ScanLength):8:3,' ',
                 (Lat-spacing):8:3,'     ',Spacing:8:3,'
',ScanRate:8:3);
```

```
    END;

  END;



  PROCEDURE OutScan
(*-----------------*)
            (N,IScan : integer);

  var

     FileName : string;
     j        : Integer;
     f        : text;
     S      : string;

  BEGIN
     IF ScanIn = 'LAT' THEN
        S := 'y'
     ELSE
        s := 'x';


     FileName := DirString+'\'+ObsName+s+Intstr(IScan-1) + '.dat';
     Assign(f,FileName);
     ReWrite(f);
     PutFlHead(f,IScan-1);
     FOR J := 1 TO N DO
        Writeln(f,Scan[J],' ',AntLat[J]:8:3,
                ' ',AntLong[J]:8:3);
     Close(f);
     IF ScanIN = 'LAT' THEN Coord := AntLat
                       ELSE Coord := AntLong;
     IF (IScan MOD 15) = 0 THEN Clear := TRUE;
     PlotScan (N,(IScan-1));
     SetColor(White);

  END; (* OutScan *)


(*-------------------------------------------------------------------------
-*)

begin


end.
```

Appendix B

Data reduction programs

PROGRAM WriteParam

```
C generates a parameter file

      INTEGER M, N
      REAL    LatMin, LongMin, YSize, XSize, BinSize
      CHARACTER XStr*12, YStr*12, ParamFile*20,Obs*5,DirString*12

      WRITE (6,400) 'Name of parameter file          '
      READ (5,200) ParamFile
      OPEN (UNIT = 10, FILE  = ParamFile, STATUS = 'NEW')

      write (6,400) 'Name of observation             '
      read (5,300) Obs
       write(6,*) Obs
       write (10,330) 'Observation name:   ',Obs

       write (6,400) 'No of latitude scans           '
      read (5,*) M
       write (10,55)  'M:                  ',M

      write (6,400) 'No of longitude scans           '
      read (5,*) N
       write (10,55)  'N:                  ',N

       write (6,400) 'Minimum latitude               '
      READ (5,100) LatMin
      write (10,110) 'Bottom:             ',LatMin

      write (6,400) 'Minimum longitude               '
      read (5,100) LongMin
       write (10,110) 'Right:              ',LongMin

      write (6,400) 'Latitude size                   '
       READ (5,100) YSize
      write (10,110) 'Y size:             ',YSize

      write (6,400) 'Longitude size                  '
      read (5,100) XSize
       write (10,110) 'X size:             ',XSize

      write (6,400) 'Bin size                        '
      read (5,100) BinSize
       write (10,110) 'Sample spacing:     ',BinSize

      write (6,400) 'Directory containing X scans    '
       READ (5,200) XStr
      write (10,220) 'X scan directory:   ',XStr

      write (6,400) 'Directory containing Y scans    '
      read (5,200) YStr
       write (10,220) 'Y scan directory:   ',YStr

      write (6,400) 'Directory to contain output files '
      read (5,200) DirString
      write (10,220) 'Output directory:   ',DirString


      CLOSE(10)

50    FORMAT (I4)
55    FORMAT (A20,I4)
100   FORMAT (F10.5)
110   FORMAT (A20,F10.5)
200   FORMAT (A12)
220   FORMAT (A20,A12)
300   FORMAT (A5)
330   FORMAT (A20,A5)
400   FORMAT (A,$)

      END
```

program Scale

```
C     ==============

C corrects each scan for effect of delay constant in radiometer
C finds the coefficients of a linear baseline
C              - stored in py1.dat and px1.dat
C converts from mV to K using noise diode calibration

      Implicit  None
      Integer Cnt,I,L,K,M,N,ScanCnt,
     &        Lat,ScanNo,junk1,junk2,StartNo, EndNo,First,Last,
     &        InUnit,OutUnit,CoeffUnit,LogUnit
      Character IStr*3 ,Line*80,XScan*25,YScan*25,JStr*3,IntStr*3,
     &          OldFile*40, NewFile*40 , RmBlanks*40, DirString*15,
     &          OldScan*25,SubFile*40,Obs*8,Sub*1,LogFile*40,
     &          OldStr*12

      Real BinWidth, ScanCoord(1000), Coord(1000),Z(1000),
     &     LongMax, A1,A2,A3, Sample(1000),
     &     LatMax,XSize, YSize,NoiseDVM, NoiseTemp,SampleRate,
     &     TDelay, Delay,NewCoord(1000),X1,X2,S,C, Top,Bottom, Left,
     &     Right
      Real LatMin,LongMin,CoordMax,CoordMin
      Real  Xp, Yp, Zp
      PARAMETER (SampleRate = 9.0, TDelay = 0.1)
      DATA InUnit/40/,LogUnit/10/,CoeffUnit/20/,OutUnit/50/

      CALL readparam(M, N, LatMin, LongMin, YSize, XSize,
     &               BinWidth, XScan, YScan,DirString,obs)


C process either latitude or longitude scans

      WRITE(6,*) '1 ... Process Latitude  Scans'
      WRITE(6,*) '2 ... Process Longitude Scans'
      Write(6,*) '                              '
      Write(6,111) 'Enter 1,2  => '
      READ(5,*) Lat
111   FORMAT (A,$)

      LatMax  = LatMin  + YSize
      LongMax = LongMin + XSize

      WRITE(6,222) 'Directory containing original scans =>  '
      Read(5,99) OldStr
99    FORMAT (A12)

C set up filenames and variables

      IF (Lat.EQ.1) THEN
           LogFile  = DirString//'scaley.log'
           CoordMax = (N-1)*BinWidth + LatMin
           CoordMin = LatMin
           SubFile = DirString//'py1.dat'
           OldScan = OldStr//'/'//obs//'y'
           StartNo  = 1
           EndNo    = M
      ELSE
           LogFile = DirString//'scalex.log'
           CoordMax = (M-1)*BinWidth + LongMin
           CoordMin = LongMin
           SubFile  = DirString//'px1.dat'
           OldScan = OldStr//'/'//obs//'x'
           StartNo  = 1
           EndNo    = N
      ENDIF
      LogFile = RmBlanks(LogFile)
      SubFile = RmBlanks(SubFile)
      Open(Unit=LogUnit,File=LogFile)
      Open(Unit=CoeffUnit,File=SubFile)

      WRITE(6,222) 'Noise diode temperature => '
      Read(5,*) NoiseTemp

C option of processing only a sub-section of the data

      WRITE(6,222) 'Select a sub section of the data (y/n) => '
      READ(5,101) Sub
222   FORMAT (A,$)
101   FORMAT (A1)

      IF ((SUB.EQ.'Y').OR.(SUB.EQ.'y')) THEN
        ScanCnt = 0
         WRITE(6,222) 'Bottom => '
         READ(5,*) Bottom
         WRITE(6,222) 'Top    => '
         READ(5,*) Top
         WRITE(6,222) 'Right  => '
         READ(5,*) Right
         WRITE(6,222) 'Left   => '
         READ(5,*) Left
         write(6,*) Top,bottom,left,right
         Right  = INT((Right-LongMin)/BinWidth)*BinWidth+
     &                LongMin
         Left   = INT((Left-LongMin)/BinWidth)*BinWidth+
     &                LongMin

         Bottom = INT((Bottom-LatMin)/BinWidth)*BinWidth+
     &                LatMin
         Top    = NINT((Top-LatMin)/BinWidth)*BinWidth+LatMin
                  write(6,*) left,right,top,bottom
         IF (LAT.EQ.1) THEN
            StartNo  = NINT((Right-LongMin)/XSize*(M-1))
            EndNo    = NINT((Left-LongMin)/XSize*(M-1))
            StartNo  = MAX(StartNo,1)
            EndNo    = MIN(EndNo,M)
            LongMax  = (EndNo-1)*BinWidth+LongMin
            LongMin  = (StartNo-1)*BinWidth+LongMin
            CoordMin = Right
            CoordMax = Left
```

```
      ELSE

         StartNo  = INT((Bottom-LatMin)/YSize*(N-1))
         EndNo    = NINT((Top-LatMin)/YSize*(N-1))
         StartNo  = MAX(StartNo,1)
         EndNo    = MIN(EndNo,N)
         LatMax   = (EndNo-1)*BinWidth+LatMin
         LatMin   = (StartNo-1)*BinWidth+LatMin
         CoordMin = Bottom
         CoordMax = Top
      END IF
    END IF

C process scans

    DO ScanNo = StartNo,EndNo

     IF ((SUB.EQ.'Y').OR.(SUB.EQ.'y')) THEN
        ScanCnt = ScanCnt + 1
        Write(6,*) 'Scan no ',ScanNo,' to scan no',ScanCnt
     ELSE
        ScanCnt = ScanNo
        Write(6,*) 'Scan no ',ScanNo
     END IF
     IStr = IntStr(ScanNo)
     JStr = IntStr(ScanCnt)

     IF (Lat.EQ.1) THEN
        NewFile = YScan//JStr//'.dat'
        OldFile = OldScan//IStr//'.dat'
     ELSE
        NewFile = XScan//JStr//'.dat'
        OldFile = OldScan//IStr//'.dat'
     END IF

     OldFile = RmBlanks(OldFile)
     NewFile = RmBlanks(NewFile)

     OPEN(InUnit, File = OldFile,STATUS = 'OLD')
     OPEN (OutUnit, Name = NewFile)

     Cnt = 0

C read header from input file
     DO L=1,3
        Read (InUnit,'(80A)') line
     end do
     Read (InUnit,*) junk1,junk2,NoiseDVM
     Write(LogUnit,*) SCanNo,' Noise = ',NoiseDVM
     DO L=1,2
           Read (InUnit,'(80A)') line
     END DO

C read scan data - store in arrays Coord, ScanCoord and Z
C       convert mV to K using noise diode calibration

5    READ(InUnit,55,END = 50) Zp,Yp,Xp
     IF (Yp.GE.180) THEN
         Yp = Yp - 360
     END IF
     IF (Xp.GE.180) THEN
         Xp = Xp - 360
     END IF
     IF (Lat.EQ.1) THEN
        IF ((Yp.GE.CoordMin-0.1).AND.
&         (Yp.LE.CoordMax+0.1)) THEN
           Cnt = Cnt + 1
           Coord(Cnt) = Xp
           ScanCoord(Cnt) = Yp
           Z(Cnt) = K*NoiseTemp*Zp/NoiseDVM
        ENDIF
     ELSE
        IF ((Xp.GE.CoordMin-0.1).AND.
&          (Xp.LE.CoordMax+0.1)) THEN
           Cnt = Cnt + 1
           ScanCoord(Cnt) = Xp
           Coord(Cnt) = Yp
           Z(Cnt) = NoiseTemp*Zp/NoiseDVM
        ENDIF
     ENDIF
     GOTO 5


50   CONTINUE
     Close (InUnit)


C correct for hysteresis
C corrected coordinates written to array NewCoord

     Delay = SampleRate * TDelay
     DO I = 1,Cnt
        Sample(I) = FLOAT(I)
     END DO


     DO I = 1,Cnt
        CALL Quad(Sample(Max(1,I-4)),Sample(I),
&                 ScanCoord(Max(1,I-4)),
&                 Min(Cnt,I+4) - Max(1,I-4)+1,A1,A2,A3)
        NewCoord(I) = A1*(0.0 - Delay)**2 +
&                   A2*(0.0 - Delay) + A3
     END DO

C find coefficients of a linear baseline

     First = 1
     Last  = Cnt
     X1  = (NewCoord(First) -CoordMin)/(CoordMax-CoordMin)
     X2  = (NewCoord(Last)-CoordMin)/(CoordMax-CoordMin)
     S = -(Z(Last)-Z(First))/(X2-X1)
     C = -Z(1) - S * X1
```

```
      WRITE(CoeffUnit,200) ScanCnt, 0, C
      WRITE(CoeffUnit,200) ScanCnt, 1, S

200   FORMAT (I4,I3,F12.5)

C write corrected and scaled scan to output file
      DO I = 1,Cnt
         IF (Lat.EQ.1) THEN
            Write(OutUnit,55) Z(I), NewCoord(I), Coord(I)
         ELSE
            Write(OutUnit,55) Z(I), Coord(I), NewCoord(I)
         END IF
      END DO
      Close(OutUnit)

55    FORMAT (F17.10,2F10.3)
      END DO
      CLOSE(20)
      Close(10)
      END
```

program Cross

```
C
C     finds the intersection points of the latitude and longitude scans
C     and writes results in xy.dat

      Implicit  None

      Integer M, N, LatSc, LongSc,I,LongCnt,LatCnt,L,
     &        ILong, ILat, Count,
     &        LatUnit,LongUnit,CrossUnit
      Character IStr*3,IntStr*3,XScan*25,YScan*25,
     &          FileName*40,NewFile*40,RmBlanks*40,
     &          Obs*8,DirString*15,CrossFile*40
      Real Lat, Long, ALat, ALong, XLong(1000,150),
     &    YLong(1000,150)
      Real XLat(1000),YLat(1000), LatMin,LatMax,LongMin,LongMax
      Real ScanWidth, YSize, XSize, Delta, DSearch, MLat,MLong,
     &     Xp,Yp,CLat,CLong, junk,X,Y,XXLat(150),YYLat(150),
     &     XXLong(150),YYLong(150)
      DATA LatUnit/10/,LongUnit/30/,CrossUnit/40/


      CALL ReadParam(M,N,LatMin,LongMin,YSize, XSize, ScanWidth,
     &               XScan,YScan,DirString,Obs)

      CrossFile = DirString//'xy.dat'
      CrossFile  = RmBlanks(CrossFile)
      Open(CrossUnit,Name=CrossFile)

      Delta = 0.04
      DSearch = 0.25

      LatMax    = (N-1)*ScanWidth + LatMin
      LongMax   = (M-1)*ScanWidth + LongMin

C read in all longitude scans
      DO LongSc = 1,N

         write (6,*) 'Reading in longitude scan no', LongSc
         IStr = IntStr(LongSc)
         FileName = XScan//IStr//'.dat'
         NewFile = RmBlanks(FileName)
         OPEN (LongUnit,File = NewFile,Status='OLD')

         LongCnt = 0
9         read (LongUnit,100,END=90) junk,Y,X
         LongCnt = LongCnt + 1
         XLong(LongCnt,LongSc) = X
         YLong(LongCnt,LongSc) = Y
         GOTO 9
90        CONTINUE
         CLOSE(LongUnit)
      END DO

C for each latitude scan find intersection points

      DO LatSc = 1,M
c     read a latitude scan into arrays X and Y

        Long = (LatSc -1) * ScanWidth + LongMin
        Write(6,*) 'Latscan no  ',LatSc

        IStr = IntStr(LatSc)
       FileName = YScan//IStr//'.dat'
        NewFile = RmBlanks(FileName)

        OPEN(LatUnit,File=NewFile,Status='OLD')

        LatCnt =  0
5       READ(LatUnit,100,END = 50) junk,Y,X
        IF ((Y.GE.LatMin-0.5).AND.(Y.LE.LatMax+0.5)) THEN
             LatCnt = LatCnt + 1
             XLat(LatCnt) = X
             YLat(LatCnt) = Y
        ENDIF

        GOTO 5

50     CONTINUE
       Close(LatUnit)

       DO LongSc = 1,N

          Lat = LatMin + (LongSc-1)*ScanWidth

C  calculate the actual average longitude of the scan

          Count = 0
          ALong = 0
          DO I=1,LatCnt
          IF ((YLat(I).GE.Lat-dSearch).AND.
     &        (YLat(I).LE.Lat+dSearch)) THEN
              Count = Count + 1
              Along = Along + XLat(I)
          END IF

          END DO

          ALong = ALong/Count

C         read from file the values in range ALong +/- DSearch
C         values read into the arrays XLong & YLong
C         values in range ALong +/- Delta stored in XXLong & YYLong
C         calculate the average latitude of the scan

          ALat = 0
          Count = 0
          ILong = 0

c calculate the ave lat of long scan ALong+/- DSearch
```

```
        DO L=1,LongCnt
             IF((ABS(XLong(L,LongSc)-ALong)).LE.DSearch) THEN
               aLat = ALat + YLong(L,LongSc)
               Count = Count + 1
             END IF
             IF ((ABS(XLong(L,LongSc)-ALong)).LE.Delta) THEN
                 ILong = ILong + 1
                 XXLong(ILong) = XLong(L,LongSc)
                 YYLong(ILong) = YLong(L,LongSc)
            END IF
        END DO
        ALat = ALat/Count

C     store the lat scan values in the range ALat +/- Delta
C     values stored in arrays XXlat,YYLat

      ILat = 0
      DO L = 1,LatCnt
         IF ((ABS(YLat(L)-ALat).LE.Delta)) THEN
              ILat = ILat + 1
              XXLat(ILat) = XLat(L)
              YYLat(ILat) = YLat(L)
         END IF
      END DO

C fit least squares straight lines to data

      CALL LSfit(XXLong,YYLong,ILong,MLong,CLong)
      CALL LSFit(YYLat,XXLat,ILat,Mlat,CLat)

C for longitude scan: x = Mlong*y + CLong
C for latitude scan:  y = Mlat * x + CLat
C these lines intersect at (Xp, Yp)

      Xp = (MLat*CLong + CLat)/(1-MLong*MLat)
      Yp = MLong * Xp + CLong

      write(CrossUnit,200)  LatSc,LongSc,Xp,Yp

      END DO

      END DO

100   FORMAT(F17.10,2F10.3)

200   FORMAT(2I4,2F11.4)

      close(CrossUnit)

      END

      SubRoutine LSFit (X,Y,N,M,C)
C     ==============
C
C fit least squares straight line

Implicit  None

Integer Nx, I, N
Real*8  Xp, Yp, Sx, Sy, Sxx, Sxy
Real    X(50), Y(50), M, C

Nx = 0
Sx = 0.0
Sy = 0.0
Sxx = 0.0
Sxy = 0.0

DO I = 1,N
     Xp = X(I)
     Yp = Y(I)
     Nx = Nx + 1
     Sx = Sx + Xp
     Sy = Sy + Yp
     Sxx = Sxx + Xp * Xp
     Sxy = Sxy + Xp * Yp
END DO

M = (Sxy - Sx*Sy/Nx)/(Sxx-Sx*Sx/Nx)
C = Sy/Nx-M*Sx/Nx

END
```

Program Bin1

```
C     ==============

C bins the scan on the intersection points
C the program must be run separately for latitude and longitude scans

C input files: crossunit (xy.dat) contains coordinates of intersection
C                    points
C            : coeffunit (px1.dat/Py1.dat) contain linear baseline
C                    coefficients as determined in Scale.f
C output files: OutUnit contains the binned scans (binx.dat/biny.dat)

      Implicit  None
      Integer M, N, Cnt,I,J,L,Count(200),BinNo,EmptyBins,
     &        Lat,ScanNo,NScan,NBin,Deg,
     & InUnit,CrossUnit,CoeffUnit,OutUnit,LogUnit
      Character IStr*3,IntStr*3,XScan*25,YScan*25, LogFile*40,
     &          FileName*40,BinFile*40, RmBlanks*40,DirString*15,
     &          Obs*8, CrossFile*40,SubFile*40,SubBase*1
      Real BinWidth, Coord(1000), Z(1000),LongMax,
     &     LatMax,YSize,XSize,Sub(0:1,200)
C
      Real MinBin,MaxBin, LatMin,LongMin,CoordMax,CoordMin
      Real Cross(200,200), OutBin(200), Xp, Yp, Zp,junk

      DATA InUnit/10/,CrossUnit/20/,CoeffUnit/30/,
     &     OutUnit/40/,LogUnit/50/

      CALL ReadParam(M,N, LatMin, LongMin, YSize, XSize,
     &               BinWidth, XScan, YScan, DirString,Obs)

      WRITE(6,*) '1 ... Process Latitude Scans  '
      WRITE(6,*) '2 ... Process Longitude Scans '
      WRITE(6,111) 'Enter choice =>  '

      READ(5,*) Lat

      WRITE(6,111) 'Subtract linear baseline from the scans? (Y/N)  '
      READ(5,555) SubBase

555   FORMAT (A1)
111   FORMAT (A,$)


      LatMax = LatMin + YSize
      LongMax = LongMin + XSize

      IF (Lat.EQ.1) THEN
           CoordMax = LatMax
           CoordMin = LatMin
           BinFile  = DirString//'biny.dat'
           BinFile  = RmBlanks(BinFile)
           NScan = M
           NBin  = N
           LogFile = DirString//'bin1y.log'
           SubFile = DirString//'py1.dat'
      ELSE
           CoordMax = LongMax
           CoordMin = LongMin
           BinFile  = DirString//'binx.dat'
           BinFile  = RmBlanks(BinFile)
           NScan = N
           NBin  = M
           LogFile  = DirString//'bin1x.log'
           SubFile  = DirString//'px1.dat'
      ENDIF


      FileName  = DirString//'xy.dat'
      CrossFile = RmBlanks(FileName)
      SubFile   = RmBlanks(SubFile)
      LogFile   = RmBlanks(LogFile)
      OPEN (OutUnit,Name=BinFile)
      OPEN (LogUnit,Name=LogFile)

C read in linear baseline coefficients

      IF ((SubBase.EQ.'Y').OR.(SubBase.EQ.'y')) THEN
        SubBase = 'Y'
        Open(CoeffUnit,File = SubFile,STATUS='OLD')
4       READ (CoeffUnit,400,END=14) I,Deg,Sub(Deg,I)
        GOTO 4
400     FORMAT (I4,I3,F12.5)
14      CONTINUE
        CLOSE(CoeffUnit)
      END IF

C     read in crossing points from the file crossxy.dat

      OPEN(CrossUnit,File = CrossFile,STATUS='OLD')
      IF (Lat.EQ.1) THEN
1        READ(CrossUnit,100,END=11) I,J,junk, Cross(I,J)
         GOTO 1
      ELSE
2        READ(CrossUnit,100,END=11) I,J,Cross(I,J), junk
         GOTO 2
      ENDIF

11    CONTINUE
      Close(CrossUnit)

100   FORMAT(2I4,2F11.4)

C     process each scan

      DO ScanNo = 1, NScan

        IStr = IntStr(ScanNo)

        IF (Lat.EQ.1) THEN
```

```
         FileName = YScan//IStr//'.dat'
         Write(6,*) 'Binning  latitude scan no ',ScanNo
      ELSE
         FileName = XScan//IStr//'.dat'
         Write(6,*) 'Binning longitude scan no ',ScanNo
      END IF

      FileName = RmBlanks(FileName)
      OPEN(InUnit,FILE=FileName,STATUS='OLD')
      Cnt = 0
5     READ(InUnit,55,END = 50) Zp,Yp,Xp

      IF (Lat.EQ.1) THEN
            Cnt        = Cnt + 1
            Coord(Cnt) = Yp
            IF (SubBase.EQ.'Y') THEN
               Z(Cnt)= Zp + (Yp-CoordMin)/YSize*Sub(1,ScanNo)+
     &                  Sub(0,ScanNo)
            ELSE
               Z(Cnt)     = Zp
            END IF
      ELSE
            Cnt        = Cnt + 1
            Coord(Cnt) = Xp

            IF (SubBase.EQ.'Y') THEN
               Z(Cnt)= Zp + (Xp-CoordMin)/XSize*Sub(1,ScanNo)+
     &                  Sub(0,ScanNo)
            ELSE
               Z(Cnt)     = Zp
            END IF
      ENDIF
      GOTO 5

55    FORMAT (F17.10,2F10.3)
50    CONTINUE
      Close (InUnit)

C     bin the scan

      DO BinNo = 1,NBin
          OutBin(BinNo) = 0.0
          Count(BinNo) = 0
      END DO

C include data points within +/- BinWidth/2
      DO BinNo = 1,NBin
        IF (Lat.EQ.1) THEN
           MinBin = Cross(ScanNo,BinNo) - BinWidth/2
           MaxBin = Cross(ScanNo,BinNo) + Binwidth/2
        ELSE
           MinBin = Cross(BinNo,ScanNo) - BinWidth/2
           MaxBin = Cross(BinNo,ScanNo) + Binwidth/2
        ENDIF

        DO L = 1,Cnt
          IF ((Coord(L).GE.MinBin).AND.(Coord(L).LE.MaxBin)) THEN
             OutBin(BinNo) = OutBin(BinNo) + Z(L)
             Count(BinNo) = Count(BinNo) + 1
          END IF
        END DO
      END DO

42    CONTINUE
      EmptyBins = 0
      DO BinNo = 1,NBin
         IF (Count(BinNo).GT.0) THEN
           OutBin(BinNo) = OutBin(BinNo)/Count(BinNo)
                    ELSE
           EmptyBins = EmptyBins + 1
           OutBin(BinNo) = -100000
         ENDIF
      END DO

      junk = 0.0
      DO BinNo = 1,NBin
         IF (Lat.EQ.1) THEN
             WRITE(OutUnit,800) ScanNo,BinNo, junk,
     &                  Cross(ScanNo,BinNo), OutBin(BinNo)
         ELSE
             WRITE(OutUnit,800) BinNo, ScanNo,Cross(BinNo,ScanNo),
     &                  junk, OutBin(BinNo)
         ENDIF
      END DO

      IF (EmptyBins.NE.0) THEN
         Write(6,*) 'There are ',EmptyBins,' empty bins'
         Write(LogUnit,*) 'There are ',EmptyBins,
     &              ' empty bins in scan no ',ScanNo
      END IF

800   FORMAT (2I4,2F10.3,F20.10)

      END DO
      close(LogUnit)
      close(OutUnit)
      END
```

program weave

```
      Implicit  None
      Integer   NDeg, M, N, Repeat,Choice
      Real     ALim,BinWidth
      Real    LatMin,LongMin,YSize, XSize
      Real*8    Px(0:10,440),Py(0:10,440)
      Character XScan*25,YScan*25,Obs*8,MapFile*40,
     &           RmBlanks*40,LogFile*40,FileName*40,DirString*15

C NDeg is order of polynomial
C ALim is limit

      CALL ReadParam(M,N,LatMin,LongMin,YSize,XSize,
     &               BinWidth, XScan, YScan, DirString, Obs)


      MapFile = DirString//'trial.dat'
      MapFile = RmBlanks(MapFile)
      LogFile = DirString//'weave.log'
      LogFile = RmBlanks(LogFile)

C     XMap contains the longitude scans - unit 20
C     YMap contains the latitude  scans - unit 30

      Open(10,Name=LogFile)
      FileName = DirString//'binx.dat'
      FileName = RmBlanks(FileName)
      OPEN  (UNIT=20,FILE=FileName,STATUS='OLD')
      FileName = DirString//'biny.dat'
      FileName = RmBlanks(FileName)
      OPEN (UNIT=30,FILE=FileName,STATUS='OLD')
      FileName = DirString//'px.dat'
      FileName = RmBlanks(FileName)
      Open (40,Name=FileName)
      FileName = DirString//'py.dat'
      FileName =  RmBlanks(FileName)
      Open (50,Name=FileName)
      Open (60,Name=MapFile)

C     set the coeff of the polys to zero
10     FORMAT (A,$)

333   Write(6,10) 'Enter the limit =>'
      Read(5,*) ALim
      Write(6,10) 'Enter the max degree polynomial to fit =>'
      Read(5,*) NDeg

       write (10,*) 'Degree of polynomial fitted = ',NDeg
       write (10,*) 'Limit                       = ',Alim

       CALL FITPOLY (NDeg,M,N,Px,Py,ALim,
     &               YSize,XSize,LatMin,LongMin)

      IF (Repeat.EQ.1) GOTO 333
100   End

      Subroutine FitPoly
C      ==================
     & (NDeg,M,N,SPx,SPy,ALim,YSize, XSize,
     &  LatMin,LongMin)

        Implicit none

        Integer M, N, NDeg, Deg, Flag1, I, J, L, K
        Real    A(400,400), B(400,400),OutA(400,400),
     &          OutB(400,400),Diff(400,400),C(400,400),
     &          XOBS(400),YOBS(400), XGrid(400,400),
     &          Px(0:10,400), Py(0:10,400), CPoly(0:10),
     &          SPx(0:NDeg,N),YSize,XSize,LatMin,LongMin,
     &          SPy(0:NDeg,N), ALim, junk, YGrid(400,400),
     &          OldAS, AC, AD, AS, AR

      DO K=0,10
          DO L=1,400
             Px(K,L)=0.0
             Py(K,L)=0.0
          END DO
      END DO

C  read from files into XMap and YMap

      WRITE(6,*) 'Reading in the longitude scans'
11    READ (20,100,END=110) I, J,XGrid(I,J), junk,
     &                      A(I,J)
      GOTO 11
110   CONTINUE
      CLOSE(20)

      WRITE(6,*) 'Reading in the latitude scans'
22    READ (30,100,END=120) I, J, junk, YGrid(I,J),B(I,J)
      GOTO 22
120   CONTINUE
      CLOSE(30)

100   FORMAT(2I4,2F10.3,F17.10)

        DO J=1,N
           DO I=1,M
              OutA(I,J) = A(I,J)
              OutB(I,J) = B(I,J)
              Diff(I,J) = A(I,J) - B(I,J)
              C(I,J) = (A(I,J) + B(I,J))/2
           END DO
        END DO
        FLag1 = 1

c       repeat
```

```
10        write (6,*) 'fitting poly to x scans'
          DO  J=1,N                    ! for each row
            DO I=1,M
               XOBS(I)= (XGrid(I,J) -LongMin)/XSize
               YOBS(I)= Diff(I,J)
            END DO

            CALL LMSP(M,XOBS,YOBS,NDeg,CPoly)

            DO L=0,NDeg
                  Px(L,J)=-0.7*CPoly(L)               !Px(N,J) holds
coeff for ls poly fit to the J scan
                  SPx(L,J)=SPx(L,J)-0.7*CPoly(L)  !    sums ACOR

            end do

          END DO

C         write (6,*)  'subtracting from x scans'

          DO  J=1,N
            DO  I=1,M
              Deg=NDeg
              AC=0.
              DO L=0,Ndeg
                AC=AC*(XGrid(I,J)-LongMin)/XSize
                AC=AC+Px(Deg,J)     ! value of ls poly fitted to rows at
KA,JA
               Deg=Deg-1
              end do
              OutA(I,J) = OutA(I,J) + AC
            END DO
          END DO


20          write (6,*)  'fitting poly to y scans'
          DO  I=1,M
               DO J=1,N
                 XOBS(J) = (YGrid(I,J) - LatMin)/YSize
                 YOBS(J) = Diff(I,J)
               END DO

               CALL LMSP(N,XOBS,YOBS,Ndeg,CPoly)
              DO  L=0,Ndeg
                Py(L,I) = 0.7*CPoly(L)              !Py(N,J) holds N
coeff for J ploy
1401            SPy(L,I)= SPy(L,I)+0.7*CPoly(L)
              end do
          END DO

C          write (6,*) 'subtracting from y scans'

          DO  J=1,N
            DO  I=1,M
              Deg=NDeg
              AD=0.
              DO L=0,Ndeg
                AD=AD*(YGrid(I,J)-LatMin)/YSize
                AD=AD+Py(Deg,I)     ! value of ls poly fitted to col at
KA,JA
               Deg=Deg-1
              end do

              OutB(I,J) = OutB(I,J) + AD
            END DO
          END DO

          AR=0.
          AS=0.
          DO  J=1,N
            DO  I=1,M
              Deg=NDeg
              AC=0.
              AD=0.
              DO L=0,Ndeg
                AC=AC*(XGrid(I,J) - LongMin)/XSize
                AD=AD*(YGrid(I,J) -  LatMin)/YSize
                AC=AC+Px(Deg,J)     ! value of ls poly fitted to rows at
KA,JA
                AD=AD+Py(Deg,I)     ! value of ls poly fitted to col at
KA,JA
               Deg=Deg-1
              end do

              AC=AC-AD                  ! diff between row and col poly
                   Diff(I,J)=Diff(I,J)+AC       ! add this difference to
the map
              C(I,J) = (OutA(I,J) + OutB(I,J))/2
              AS=AS+ABS(AC)             !sum of these difference over all
points in map
              AR=AR+1.

            end do
          end do
          AS=AS/AR                      ! average diff between maps
          write (6,*) 'Average difference =  ', AS
          write (10,*) 'AS= ',AS

          IF (AS.LE.ALIM) GO TO 15        ! continue until this diff is
< alim
C          write(6,*) AS,AS-OldAS
          IF (Flag1.EQ.0) THEN
             write(10,*) AS-OldAS
             IF (ABS(AS-OldAS).LE.ALim) GOTO 15
          ELSE
             Flag1 = 0
          END IF

          OldAS = AS
          GO TO 10
C     write coefficients to files
```

```
15    DO I=1,N
         DO Deg=0,NDeg
            WRITE (40,400) I,Deg, SPx(Deg,I)
            WRITE (50,400) I,Deg, SPy(Deg,I)
         END DO
      END DO
      junk = 0.0
      DO J=1,N
         DO I=1,M
            WRITE(60,100) I,J,junk,junk,c(I,J)
         END DO
      END DO
      close(40)
      close(50)
      close(60)

400   FORMAT (I4,I3,F17.5)


      RETURN
      END



       SUBROUTINE LMSP (N, XData, YData, NDeg, CPoly)
C       ===============

       IMPLICIT NONE
       INTEGER MaxN, MaxDeg
       PARAMETER (MaxN = 1024, MaxDeg = 10)
       INTEGER N, NDeg, i, IDeg, Ierr
       REAL XData(N), YData(N), CPoly(0:NDeg)
       DOUBLE PRECISION P(2*MaxN), T(4*(MaxDeg+1))
       REAL X(MaxN), C(MaxDeg+3),A(MaxDeg),B(MaxDeg),Stats(MaxDeg+3)

       DO I = 1,N
          X(I) = XData(I)
       END DO
       CALL RLFOTH(X,YData, N,100.0, NDeg,IDeg,P,C,Stats,
     &             A,B,Ierr)
       CALL RLDOPM (C,NDeg,A,B,T)
       DO i = 0,NDeg
         CPoly(i) = C(i+1)
       END DO
       RETURN
       END
```

program FinalBin

```
C   subtracts the polynomials generated by weave from the original data
C   bins the data on a regular grid using a circular sinc
C   input files:  Px.dat & py.dat - polynomial coefficients
C   output files: ****.dat contains final map (where **** is obs name)
C                 ****x.dat contains final longitude map
C                 ****y.dat contains final latitude map

      Implicit  None
      Integer  Deg,NDeg, I,J,N,M,Width,HPW,Choice
      Real      LatMin,LongMin,YSize,XSize,WFunc(0:200),
     &          SubX(0:1,200),SubY(0:1,200),Xp,Yp,Zp,dR
      Real*8    Px(0:10,200),Py(0:10,200)
      Real BinWidth, XMap(200,200),YMap(200,200)
           Logical  XFlags(200),YFlags(200)
      Character XScan*25,YScan*25,
     &          FileName*40, RmBlanks*40, Obs*8,SubBaseX*1,
     &          SubBaseY*1, DirString*15


      CALL ReadParam(M,N,LatMin,LongMin,YSize, XSize,
     &                   BinWidth,XScan,YScan,DirString,Obs)

      CALL Exclude(M,N,XFlags,YFlags,DirString)

      FileName = DirString//'bin2.log'
      FileName = RmBlanks(FileName)
      OPEN (20,Name=FileName)

      FileName = DirString//'px.dat'
      FileName = RmBlanks(FileName)
      OPEN (UNIT=30,FILE=(FileName),  STATUS='OLD')

      FileName = DirString//'py.dat'
      FileName = RmBlanks(FileName)
      OPEN (UNIT=40,FILE=(FileName),  STATUS='OLD')

      FileName = DirString//Obs//'x.dat'
      FileName = RmBlanks(FileName)
      OPEN (80,Name = FileName)

      FileName = DirString//Obs//'y.dat'
      FileName = RmBlanks(FileName)
      OPEN (90,Name = FileName)

      FileName = DirString//Obs//'.dat'
      FileName = RmBlanks(FileName)
      OPEN (70,NAME = FileName)

      Write(6,*) '1 ... Gaussianfunction'
      Write(6,*) '2 ... Weighted sinc function '
      Write(6,*) '3 ... Tapered bessel function '
      REad(5,*) Choice

      GOTO (11,22,33) Choice
11    Write(6,10) 'Gaussian half power width  => '
      Read (5,*) HPW
      Width = HPW
      GOTO 21
22    CALL Sinc(WFunc,dR)
      GOTO 21

33    CALL Bessel(WFunc,dR)
      width = binwidth * 1000
      GOTO 21

21    CONTINUE

      Write(6,10) 'Subtract a linear baseline from lat scans?'
      Read(5,1) SubBaseY
      IF (SubBaseY.EQ.'y') THEN
          SubBaseY = 'Y'
      END IF

      Write(6,10) 'Subtract a linear baseline from long scans?'
      Read(5,1) SubBaseX
      IF (SubBaseX.EQ.'y') THEN
          SubBaseX = 'Y'
      END IF

1     FORMAT (A1)
10    FORMAT (A,$)

      WRITE(6,*) 'Reading in the coefficients of the polynomials'
3     READ(30,800,END=13) I,Deg,Px(Deg,I) ! read in from Xpoly.dat
      GOTO 3

13    CONTINUE
      NDeg = Deg
      CLOSE(30)

4     READ(40,800,END=14) I,Deg,Py(Deg,I) ! read in from YPoly.dat
      GOTO 4
14    CONTINUE
      CLOSE(40)

      IF (SubBaseX.EQ.'Y') THEN
         FileName = DirString//'px1.dat'
         OPEN (UNIT=30,FILE=RmBlanks(FileName),  STATUS='OLD')
7          READ(30,300,END=17) I,Deg,SubX(Deg,I)
         GOTO 7
17       CONTINUE
         CLOSE(30)

         DO J = 1,N
            Px(0,J) = Px(0,J) + SubX(0,J)
            Px(1,J) = Px(1,J) + SubX(1,J)
         END DO
      END IF
```

```
      If (SubBaseY.EQ.'Y') THEN
         FileName = DirString//'py1.dat'
         OPEN (UNIT=40,FILE=RmBlanks(FileName),  STATUS='OLD')
8          READ(40,300,END=18) I,Deg,SubY(Deg,I)
         GOTO 8
18       CONTINUE
         CLOSE(40)
         DO I = 1,M
            Py(0,I) = Py(0,I) + SubY(0,I)
            Py(1,I) = Py(1,I) + SubY(1,I)
         END DO

      END IF

300   FORMAT (I4,I3,F12.5)
800   FORMAT (I4,I3,F17.5)

      DO J=1,N
         DO I=1,M
            XMap(I,J) = 0.0
            YMap(I,J) = 0.0
         END DO
      END DO

C Bin the longitude scans
      CALL BinScans(XScan, N, .TRUE., Px, M, LatMin,LongMin, XSize,
     &              XFlags,XMap,BinWidth,WFunc,NDeg,Width,dR)

C bin the latitude scans
      CALL BinScans(YScan, M, .FALSE., Py, N, LatMin,LongMin, YSize,
     &              YFlags,YMap,BinWidth,WFunc,NDeg,Width,dR)

       DO J = 1,N
          DO I = 1,M
            Xp = (I-1) * BinWidth + LongMin
            Yp = (J-1) * BinWidth + LatMin
            IF (XFlags(J)) THEN
               Zp = YMap(I,J)
            ELSE
               IF (YFlags(I)) THEN
                  Zp = XMap(I,J)
               ELSE
                  Zp = (XMap(I,J) + YMap(I,J))/2
               END IF
            END IF
            IF ((XFlags(J)).AND.(YFlags(I))) THEN
               Zp = -100 000
               write(20,*) 'X and Y empty at',I,J
               END IF

            WRITE(70,700) I,J,Xp,Yp,Zp
            WRITE(80,700) I,J,Xp,Yp,XMap(I,J)
            WRITE(90,700) I,J,Xp,Yp,YMap(I,J)

         END DO
      END DO

700   FORMAT(2I4,2F10.3,F20.10)

      CLOSE(70)
      CLOSE(80)
      CLOSE(90)

999   close(20)
      END


      SUBROUTINE BinScans(ScanNm, NScan, Long,Poly, NBin,LatMin,LongMin,
     &                    Size,Flags, Map,BinWidth,WFunc,NDeg,Width,dR)


      implicit none
      INTEGER NScan,SCanNo,Cnt,Counts(200),NBin,NSam(200,200),
     & Count,DEg,NDeg,L,Start,End,BinNo,ScanCnt,Width
      LOGICAL Flags(200), Long
      REAL X(1500,200), Y(1500,200), Z(1500,200),Xp,Yp,Zp,
     &     Size,Map(200,200), A, Value, BinLAt, BinLong,
     &     Wgt, dX, dY,R,dR,WFUNC(0:101),BinWidth,LongMin,LatMin
      REAL*8 Poly(0:10,200)
      CHARACTER ScanNm * 25, FileName*40, RmBlanks*40,Istr*3,
     &          IntStr*3

      DO ScanNo=1,NScan
         DO BinNo = 1,NBin
            IF (Long) THEN
               NSam(BinNo,ScanNo) = 0
            ELSE
               NSam(ScanNo,BinNo) = 0
            END IF
         END DO
      END DO
      IF (Long) THEN
           WRITE(6,*) 'Reading Longitude Scans '
        ELSE
           WRITE(6,*) 'Reading Latitude Scans  '
        END IF

      DO ScanNo = 1,NScan

        IF (.NOT. Flags(ScanNo)) THEN
             IStr = IntStr(ScanNo)

        FileName = ScanNm//IStr//'.dat'
        FileName = RmBlanks(FileName)
        OPEN(UNIT=50,FILE=FileName,STATUS='OLD')
        Cnt = 0

5       READ(50,55,END=50) Zp,Yp,Xp
           Cnt = Cnt + 1
           X(Cnt,ScanNo) = Xp
```

```
          Y(Cnt,ScanNo) = Yp
          Z(Cnt,ScanNo) = Zp
        GOTO 5

50      CONTINUE
        Counts(ScanNo) = Cnt

        Close (50)
55      FORMAT (F17.10,2F10.3)

C subtract the polynomials
        DO Count = 1,Counts(ScanNo)
          Deg = NDeg
          A = 0
          DO L = 0,NDeg
             IF (Long) THEN
                 A = A*(X(Count,ScanNo)-LongMin)/Size
             ELSE
                 A = A*(Y(Count,ScanNo)-LatMin)/Size
             END IF
             A = A + Poly(Deg,ScanNo)
             Deg = Deg - 1
          END DO

          Z(Count,ScanNo) = Z(Count,ScanNo) + A


        END DO

        END IF
       END DO

      DO ScanNo = 1, NScan  !   for each longitude scan
          IF (.NOT. Flags(ScanNo)) THEN
          IF (Long) THEN
             WRITE(6,*) 'Binning Longitude Scan No  ',ScanNo
          ELSE
             WRITE(6,*) 'Binning Latitude Scan No  ',ScanNo
          END IF
          Start = MAX(1,SCanNo-6)
          End   = MIN(ScanNo+6,NScan)

          IF (Long) THEN
             BinLat  = (ScanNo-1)*BinWidth + LatMin
          ELSE
             BinLong = (ScanNo-1)*BinWidth + LongMin
          END IF


          DO BinNo = 1,NBin
             VAlue = 0.0
             Wgt = 0.0
             IF (Long) THEN
                BinLong = (BinNo-1)*BinWidth + LongMin
             ELSE
                BinLat  = (BinNo-1)*BinWidth + LatMin
          END IF

          DO ScanCnt = Start,End
            IF (.NOT. FLags(ScanCnt)) THEN
            DO Cnt = 1,Counts(ScanCnt)
               dX = ABS(X(Cnt,ScanCnt)-BinLong)
               dY = ABS(Y(Cnt,ScanCnt)-BinLat)
               R = NINT(SQRT(dX**2 + dY**2)/dR)

               IF (R.LE.101) THEN
                  NSam(BinNo,ScanNo) = NSam(BinNo,ScanNo)+1
                  Value = Value + WFUNC(R) * Z(Cnt,ScanCnt)
                  Wgt   = Wgt + WFUNC(R)
             end if

            END DO
            END IF
          END DO

          IF (Wgt.EQ.0.0) THEN

             Write(20,*) 'Empty Bin at ',ScanNo,BinNo
          ELSE
              IF (Long) THEN
                Map(BinNo,ScanNo) = Value/Wgt
              ELSE
                 Map(ScanNo,BinNo) = Value/Wgt
              ENDIF
          ENDIF

        END DO
        END IF
        END DO
        RETURN
        END


SUBROUTINE Sinc(Func,dR)

Integer N
Real Y,dR,Func(0:101),width
DATA pi / 3.14159 /

N = 101
width = 0.040
CALL GetRealParameter('Sample spacing [deg] ',width)
write(6,*) width
dR = (4*Width)/N
Func(0) = 1.0
DO I = 1,N
   Y = SIN(I*PI/(N/4))/(I*PI/(N/4))
   Func(I) = Y * (N-I)/N
END DO

END
```

```
      REAL FUNCTION Gauss(Width,X)

      Integer Width,X

      IF (X.GT.0) THEN
         Gauss = EXP(FLOAT(-(x**2)*log(2.0)/(width**2)))
      ELSE
         Gauss = 1.0
      END IF
      RETURN
      END


      SubRoutine Bessel(Func,dR)

      REAL Func(0:200),Arg,HF,Jn(0:1),pi
      INTEGER N,ier
      DATA pi / 3.14159 /

C         Tapered Bessel

          N = 250
          HF = 12.6
          Call GetRealParameter ('Highest Frequency[cyc/deg]', HF)
C          N = NINT(4.2/(0.001*2*HF))
C          write(6,*) 'N = ',N
          dR = 4.2 / N / (2.0*HF)
          Func(0) = 1.0
          Func(1) = 1.0
C          Do i = 1 , N - 1
           DO i = 1 , N-1
            Arg = i * dR * HF * 2.0 * pi
            Call mmbsjn (Arg, 2, Jn, ier)
            Func(i+1) = 2.0 * Jn(1) / Arg * (N - 1 - i) / (N - 1.0)
          End Do
          DO i = 0,N
          write(20,*) i,func(i)
          end do
          write(20,*) 'HF = ',HF,' dR = ',dR
          RETURN
      END
```

Program ToFits

```
C     ==============
C
C     write my data to a fits file
C     convert from antenna to full beam brightness temperature
C     missing values can be fudged
C
      Implicit  None
      Integer   Nwork, Nlogical, Nreal,Choice
      Parameter (Nwork = 600 000, Nlogical = 4, Nreal = 4)
      Logical   Flags(400,400)
      Byte      Work(NWork)
      Integer    Istat, UnitIn, UnitOut,
     &            Ctype, Blank,
     &          M, N, P1,P2,P3,
     &          I,J
      Real      Left, Right, dX, Bottom, Top, dY,
     &          Frequency, Bandwidth,    BinWidth,
     &          Map(400,400),XSize, YSize,Factor
      Real*4   TempReal
      Real*8    Matrix(9), Work1(3,3), Work2(3,3),
     &          Equinox, Bscale, Bzero, Long, Lat
      Character ImageName*80, Bunit*1,INFile*25,cjunk*25,Obs*8,
     &          DirString*15, RmBlanks*40, FileName*40,Fudge
      Data      UnitIn /10/, UnitOut /20/
C
      DO J = 1,M
         DO I=1,N
            Map(I,J) = 0.0
            Flags(I,J) = .FALSE.
         END DO
      END DO

      CALL ReadParam(M,N,Bottom,Right,YSize,XSize,
     &               BinWidth,cjunk,cjunk,DirString,Obs)


      dX = -ABS(BinWidth)
      dY = ABS(BinWidth)
      Top = Bottom + YSize
      Left = Right + XSize
      write(6,10) 'Name of input data file => '
      Read(5,'(20A)') InFile
      FileName = DirString//InFile
      OPEN(10,File = RmBlanks(FileName), STATUS = 'OLD')
      P1 = 1
      P2 = P1 + M*N*Nreal
      P3 = P2 + M*N*NLogical

      Write(6,*) '1 .. Equatorial coordinates'
      write(6,*) '2 .. Galactic coordinates'
      write(6,*) '3 .. Field centred equatorial coordinates'

       Read(5,*) Choice

      IF (Choice.EQ.1) THEN
         CType = 1
         Write(6,*) 'Equinox '
         Read(5,*) Equinox
         DO I = 1,9
           Matrix(I) = 0.0D+00
         END DO
         Matrix(1) = 1.0D+00
         Matrix(5) = 1.0D+00
         Matrix(9) = 1.0D+00
      ELSE IF (Choice.EQ.2) THEN
          Ctype = 2
          CALL GalacticToB1950Matrix(Matrix)
           Equinox = 0
      ELSE
         Ctype = 4
         CALL GetRealParameter('Source longitude [deg]',TempReal)
         Long = TempReal
         CALL GetRealParameter('Source latitude [deg]',TempReal)
         Lat = TempReal
         CALL Rotation3Matrix(-Long,Work1)
         CAll Rotation2Matrix(Lat,Work2)
         CALL MatrixMatrix (Work1,Work2,Matrix)
       END IF
      Frequency = 8400.0
      CALL GetRealParameter('Frequency /[MHZ]',Frequency)
      Frequency = Frequency * 1E6

      Factor = 2.2
      CALL GetRealParameter('Factor ',Factor)

      write(6,10) 'Fudge undefined values (y/n)?'
      read(5,'(A1)') Fudge

        IF (Fudge.Eq.'y') THEN
           Fudge = 'Y'
        END IF
      Bandwidth =    30 000 000.0
      Bunit = 'K'
      Bscale = 1.0D-03

      Bzero = 0.0D+00
      Blank = -100 000

      Equinox = 1950


        Call PutImage (UnitOut, Left, Right, dX,
     &                 Bottom, Top, dY,
     &                 Ctype, Frequency, Bandwidth,
     &                 Bunit, Bscale, Bzero, Blank, Equinox, Matrix)
C


        Call SetMap(UnitIn,Work(P1),Work(P2),M,N,Factor,Fudge)
```

```
      Call PutPixels (UnitOut,Work(P1),Work(P2),M*N)
200   Call FTCLOS (UnitOut, Istat)
10    FORMAT (A,$)
      End


      SUBROUTINE SetMap(UnitIn,Map,Flags,M,N,Factor,Fudge)
C     -----------------

      IMPLICIT none
      REAL Map(M,N), junk1,junk2, Temp(400),Flags(M,N),
     *     MT(100,2), Factor, Value
      INTEGER I,J,Blank,UnitIn,MTCount,L,No,ICnt,JCnt,
     *        M,N
      CHARACTER Fudge*1
      LOGICAL MTBin(200,200)

      Blank = -100 000

10    Read(10,100,END = 20) I,J,junk1,junk2,Map(I,J)
      GOTO 10
100   FORMAT(2I4,2F11.4,F20.10)

20    DO J = 1,N
         DO I = 1,M
            MTBin(I,J) = .FALSE.
         END DO
      END DO

      MTCount = 0
      DO J = 1,N
         DO I = 1,M
            Temp(I) = Map(M-I+1,J)
         END DO
         DO I = 1,M
            IF (Temp(I).LE.-1000.0) THEN
               MTCount = MTCount + 1
               MT(MTCount,1) = I
               MT(MTCount,2) = J
               MTBin(I,J) = .TRUE.
               Map(I,J)  = 0.0
            ELSE
               Map(I,J) = Temp(I)*Factor
            ENDIF
         END DO
      END DO
      CLOSE(UnitIn)
      IF (Fudge.EQ.'Y') THEN
      DO L = 1,MTCount
         I = Mt(L,1)
         J = Mt(L,2)
         Value = 0.0
         No = 0
         DO JCnt = MAX(J-1,1),MIN(J+1,N)
            DO ICnt = MAX(I-1,1),MIN(I+1,M)
               IF (.NOT.((ICnt.EQ.I).AND.(JCnt.EQ.J))) THEN
                  IF (.NOT.MTBin(ICnt,JCnt)) THEN
                     Value = Value + Map(ICnt,JCnt)
                     No = No + 1
                  END IF
               END IF
            END DO
         END DO
         Map(I,J) = Value/No

      END DO

      ENDIF

END
```