

Instructions for Converting Scanned TIF Images to  
Georeferenced Coverages  
—Version 1.1—

Richard E. Roger  
Spatial Information Officer Systems  
Resource Information Unit  
NSW Agriculture,  
161 Kite Street,  
ORANGE NSW 2800

September, 2001

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Two approaches to the project . . . . .	5
1.2	A worked example . . . . .	5
<b>2</b>	<b>STAGE 1</b>	<b>11</b>
2.1	Set up directory structure . . . . .	11
<b>3</b>	<b>STAGE 2</b>	<b>12</b>
3.1	Make a coverage of uniquely-numbered tic points corresponding to the corners of the LGA's map tiles . . . . .	12
3.2	Make map box coverages . . . . .	12
3.2.1	When the LGA is covered either by 1:25,000 maps only or by 1:50,000 maps only . . . . .	12
3.2.2	When the LGA is covered by both 1:25,000 maps and 1:50,000 maps . . . . .	13
3.2.3	Adding LGA tics . . . . .	13
3.3	Convert map boxes to AMG projection . . . . .	14
3.4	Checking map box coverage for an LGA . . . . .	14
<b>4</b>	<b>STAGE 3—Raster to Vector Conversion</b>	<b>16</b>
4.1	Installing and checking the TIFF images . . . . .	16
4.1.1	Listing TIFF header information . . . . .	16
4.1.2	Checking TIFF header information is consistent . . . . .	16
4.2	Thin the scanned images . . . . .	18
4.3	Convert thinned images to vector coverages . . . . .	18
<b>5</b>	<b>STAGE 4—Processing Individual Vector Coverages</b>	<b>20</b>
5.1	Edit each coverage . . . . .	20
5.1.1	Add georeference tics to coverages . . . . .	20
5.1.2	Clip off map annotation . . . . .	21
5.1.3	Correct the topology of each coverage . . . . .	21
5.2	Add the polygon attributes to each coverage . . . . .	22
5.2.1	Define items in coverage's PAT table . . . . .	22
5.2.2	Manually label the polygons . . . . .	23
5.3	Georeference and 'box' the coverage . . . . .	23
5.4	Check the coverage against the thinned raster image . . . . .	24
<b>6</b>	<b>STAGE 5—Making the Shire Maps</b>	<b>25</b>
6.1	Make "shire coverage" ALC map . . . . .	25
6.2	Impose proper shire boundary . . . . .	25

6.3	Clean up border . . . . .	25
6.3.1	Automated cleaning . . . . .	25
6.3.2	Manual check and edit . . . . .	25
<b>7</b>	<b>STAGE 6</b>	<b>26</b>
7.1	compile the metadata statement . . . . .	26
<b>A</b>	<b>Detailed Descriptions and Listings of Scripts</b>	<b>27</b>
A.1	Introduction . . . . .	27
A.1.1	Capturing output produced by scripts . . . . .	28
A.2	Main scripts relevant to both 1:25,000 and 1:50,000 map tiles . . . . .	29
A.2.1	tilek-box.sh . . . . .	29
A.2.2	tilek-bigbox.sh . . . . .	32
A.2.3	tilek-coords.sh . . . . .	33
A.2.4	tilek-grid.sh . . . . .	35
A.2.5	lga-mapboxes.sh . . . . .	37
A.2.6	lga-mapbigboxes.sh . . . . .	38
A.2.7	lga-mapoctag.sh . . . . .	38
A.3	Scripts relevant to 1:25,000 map tiles . . . . .	40
A.3.1	lga-map25boxes.sh . . . . .	40
A.3.2	lga-map25bigboxes.sh . . . . .	41
A.3.3	lga-map25coords.sh . . . . .	41
A.3.4	shire25find . . . . .	42
A.3.5	Producing the file shire25mapsheets.txt . . . . .	43
A.3.6	sheet25find . . . . .	44
A.3.7	tile25k-coords.sh . . . . .	44
A.3.8	tile25k-box.sh . . . . .	45
A.3.9	tile25k-bigbox.sh . . . . .	46
A.3.10	tile25k-box-exactname.sh and tile25k-bigbox-exactname.sh . . . . .	46
A.3.11	tile25k-grid.sh . . . . .	46
A.4	Scripts relevant to 1:50,000 map tiles . . . . .	47
A.4.1	tile50k-octag.sh . . . . .	47
A.5	Scripts for checking TIFF header information . . . . .	51
A.5.1	make-tiffdir-info.sh . . . . .	51
A.5.2	check-tifs.sh . . . . .	51
A.6	Scripts for thinning images . . . . .	52
A.7	Scripts for converting images to vector coverages . . . . .	54
A.7.1	conv2vector.aml . . . . .	54
A.7.2	t2v-one.aml . . . . .	55
A.8	Scripts to help create topologically clean coverages . . . . .	56
A.8.1	Extracting dangling lines to a new coverage . . . . .	56
A.8.2	Deleting dangling lines from a coverage . . . . .	56
A.8.3	Clipping off . . . . .	57
A.9	Scripts for processing topologically clean coverages . . . . .	59
A.9.1	Adding label points to cleaned coverages . . . . .	59
A.9.2	Adding items to coverage PAT table . . . . .	59
A.9.3	Georeferencing . . . . .	59
A.9.4	Back-transformation . . . . .	61

A.10 Utility scripts specifically written for CANRI project . . . . .	63
A.10.1 lga-tics.sh . . . . .	63
A.10.2 Imposing a common set of tics on LGA map box coverages with make-add-tics.sh . . . . .	64
A.11 Existing scripts adapted for CANRI project . . . . .	65
A.11.1 Projecting map box coverages to AMG Zone . . . . .	65
A.11.2 Killing coverages . . . . .	66

# Chapter 1

## Introduction

This document sets out how to convert scanned images of line-maps to ESRI coverages. Map coverages for a Local Government Authority (LGA) are then joined together to provide a single “shire coverage” map. The process to do this has several stages, many steps of which are automated via shell scripts and ArcInfo macros:

- STAGE 1
  - Create the directory structure in which all the work will be done.
- STAGE 2
  - make a list of uniquely-numbered tic points corresponding to the corners of the LGA’s map tiles;
  - for each LGA, set up coverages defining the bounding boxes of the map tiles it covers or overlaps, and give them the tics created in the previous step;
  - convert the coverages to projections corresponding to those used by the maps (AMG Zones 55 and 56, mostly);
- STAGE 3
  - thin each scanned image to single-pixel wide lines (as far as is possible);
  - convert each thinned image (raster data) to a coverage (vector data);
- STAGE 4
  - attribute each coverage;
  - add georeferencing tics to the coverage, giving them IDs to match the tics for its map box coverage created in the previous stage;
  - check the coverage against the thinned raster image;
  - georeference the coverage by transforming it to its map box coverage;
  - impose the map box on the ALC coverage;
  - manually check and edit the clipped coverage;
- STAGE 5

- join all the attributed, georeferenced, clipped and checked coverages for a single LGA into a final “shire coverage” map.
- STAGE 6
  - compile the metadata statement.

To help make coverages and get information about map tiles, I have written a number of scripts and created a number of data files, all of which can be found in /d8/scripts/. The scripts are presented in the Appendix, along with examples of their use.

## 1.1 Two approaches to the project

I have considered two approaches to georeferencing and cleaning up the data:

1. leave the original scanned map images ‘as is’, produce the vector coverages, and check one against the other; then georeference the vector coverags.
2. georeference the original map images (e.g., in ArcInfo’s GRID extension, register and rectify the images), and from them produce georeferenced thinned raster images and vector coverages.

I have decided to take the first approach. It has the disadvantage that once vector coverages have been georeferenced, they cannot be compared directly against the original images, but they can be back-transformed for comparison. Checking has to be done before georeferencing takes place. On the other hand, you don’t have to go through the register/rectify steps.

## 1.2 A worked example

This section contains a complete record of my workings to process the first map for a shire (the map sheet Bombala for the shire of the same name). The usage of all the AML macros is fully described in subsequent chapters. The macros themselves are listed in the Appendix. This is a refined version of what I actually did for this particular map sheet, one I arrived at after I had converted five map sheets.

1. cd to directory /d8/canri/final
2. mkdir bombala
3. cd to directory /d8/canri/work
4. There are two short AML macros in this directory: `copt.aml` and `c2f.aml`.
  - (a) You must edit `copt.aml` and set the variable, `indir`, to name what will be the source directory for the thinned images, in this case /d8/canri/thinnedtifs/bombala.
  - (b) Similarly, you must edit `c2f.aml` and set the variable, `outdir`, to name the final directory where the completed coverages will be put, in this case /d8/canri/final/bombala.

5. mkdir bombala  
cd bombala

6. Make the AML macros to create the coverages for the map boxes and the over-size boxes:

```
/d8/scripts/lga-mapboxes.sh bombala  
/d8/scripts/lga-mapbigboxes.sh bombala
```

7. Make the box coverages, and tidy up, too (remember, you may have to edit do-projectx and do-projecty to correct the AMG zone for some maps):

```
arc  
cw .  
&run makebox  
&run ../project-xll 55  
&run do-projectx  
&run kill-xll  
&run makebigbox  
&run ../project-yll 55  
&run do-projecty  
&run kill-yll  
q  
rm do-projectx.aml do-projecty.aml makebox.aml makebigbox.aml
```

8. cd to /d8/canri/thinnedtifs/bombala.

9. Convert thinned TIF images to grids. To convert a single TIF image (bombala01.tif) to a coverage (bombala):

```
arc  
&run ../t2v-one bombala01 bombala  
q
```

To convert all TIF images at once:

```
arc  
&run ../conv2vector  
q
```

10. cd back to /d8/canri/work/bombala/.

11. Copy the vectorised coverage to this workspace, and give it a conveniently short name:

```
arc  
&run ../copt bombala  
rename bombala bomb01  
q
```

12. Use ArcEdit (or ArcTools) to position the tic marks correctly on the new coverage. Save the coverage. Check the location of all four tic marks. Exit ArcEdit. At this stage, I would advise you to make no other changes to the coverage—don't start correcting errors in the topology.

13. Copy this coverage with the correct tics to the final directory, /d8/canri/final/bombala/.

```
arc
&run ../c2f bomb01
q
```

14. Clip off the map legend, etc., using the over-size map box:

```
cd ..
Edit clipoff.aml to specify the coverages you will use :
  unclipped (input)      : bomb01
  amgbigbox (input)     : bombala25y55
  bigbox (temporary)    : bombbigbox
  clipped (output)      : bomb02
```

The 'bigbox' coverage is deleted by this macro at its end (though you can comment out the kill line if you want to keep it).

15. Now run clipoff.aml, and then extract the dangling lines from the resulting coverage:

```
cd bombala
arc
&run ../clipoff
/* Inspect the errors of the transformation.  They ought
/* to be better than 0.01.
&run ../dangle-extract bomb02 bomb02dang
q
```

16. Inspect for dangles. ArcEdit or use ArcTools to fix up any problems. Save edits to a new coverage, bomb03, and re-build it:

```
build bomb03 poly
build bomb03 line
```

17. Check dangles again (extract them, view them, fix up remaining any errors):

```
arc
&run ../dangle-extract bomb03 bomb03dang
q
```

18. Before you proceed to the next step, you must be happy that you have incorporated all the lines in the coverage, and that its topology is sound and correct. There will still be dangling lines left, namely the numerals written on the map that denote the land class. There may also be map annotations such as BOMBALA TOWNSHIP or

BASALTIC INFLUENCE. You may want to delete these now because the holes in letters such as A, B, O, and P form polygons which you will later have to delete (along with their labels points)

Really, the only dangling lines left must be ones you can delete in a single, automatic, global operation later on.

Remember! You must have built the coverage as `poly` and as `line`:

```
build bomb03 poly
build bomb03 line
```

19. Once you are happy with the topology of this coverage and its remaining dangling lines, keep this coverage, and work on a copy of it:

```
arc
copy bomb03 bomb04
q
```

20. Now create labels and add items to the copy's PAT table:

```
arc
&run ../addlabels bomb04
&run ../additems bomb04
q
```

21. Identify unlabelled polygons, and add their correct labels. **For a coverage with many polygons, this is one of the most time-consuming steps of the whole process.** I do it in ArcTools.

You can add annotations to a polygon in the AGTEXT field. For example, you could add a town's name as well as labelling the polygon 9. With forest shown as PRIVATE PLANTATION, I have labelled it as 7 but put PRIV PLANT in the Ag Text field. You might also want to select all polygons labelled, e.g., SIGNIFICANT BASALTIC INFLUENCE, as, say, SBI, in the AGTEXT field.

22. Check that all the polygons (that ought to be labelled) are labelled. List the attribute statistics to screen, and see how many have an AGNUM of 0 or 99:

```
arc
&run ../stats-make bomb04
q
```

23. Correct any errors or fix up any missing labels. Here, I do it using tables though you could use ArcEdit:

```
arc
tables
select bomb04.pat
resel ( bomb04# >= 3 and bomb04# <= 7 ) or bomb04# = 9
calc agnum = 4
```

```
nselect
resel bomb04# = 14
calc agnum = 5
q
```

24. Once I am confident that all polygons are correctly closed and labelled, I copy this coverage to the final directory:

```
arc
&run ../c2f bomb04
q
```

25. Next, I delete all the dangles:

```
arc
&run ../dangle-delete bomb04 bomb05
q
```

26. Now this coverage has to be georeferenced and have the map box imposed on it  
cd ..

Edit the AML macro, `georef.aml`, and specify the right coverages:

```
amgbox (existing)  bombala25x55
unref (existing)   bomb05
newbox (new)      bombbox
georef (new)      bombg06
georefel (new)    bombg07
```

Then run it:

```
cd bombala
arc
&run ../georef
q
```

Note the transform errors—they should be about the size of a pixel (4 to 5 metres).

27. Check the georeferenced coverage thoroughly to find any unlabelled polygons, especially adjacent to the new, imposed map box boundary. Assuming the previous step created the coverage `bmbvg07`, it helps to check its statistics:

```
arc
&run ../stats-make bombg07
q
```

Correct any errors or fix up any missing labels.

28. **You should now have a completed map sheet coverage with the correct boundary, no dangles, and all polygons labelled correctly.**

29. When you're happy that this is so, copy the coverage to the final directory:

```
arc
&run ../c2f bombg07
q
```

30. You might want to produce a coverage dissolved on the attribute items:

```
arc
dissolve bombg07 bombg08 #ALL POLY
q
```

31. You may want to check this coverage against the original TIF image. To do this, you must back-transform the coverage to be in scanner coordinates. It is complicated to do this exactly but, assuming the forward transformation for the georeference was of high quality, you can do it very well using the AML macro, `back-trans.aml`. Edit it and specify the right coverages:

```
initialcov (existing)  bomb01    its tics must be in their correct places.
georef (existing)     bombg07
checkcover (new)     bombu01
```

Then run it:

```
arc
&run ../back-trans
q
```

Note the transform errors—they should be about the size of a pixel in scanner coordinates (0.01 inches).

View the `checkcover` (here, `bombu01`) against the original TIF image, `/d8/canri/scannedtifs/bombala/bombala.tif`.

# Chapter 2

## STAGE 1

### 2.1 Set up directory structure

All the work is done within a single directory, `/d8/canri/`. This has four main sub-directories:

- Scanned TIFF image files, `scannedtifs`;
- Thinned TIFF image files, `thinnedtifs`;
- Work area, `work`;
- Final sets of files, `final`.

Within each of these directories, a separate sub-directory is created for each LGA, as listed in the table below. Information about the TIFF image files is put in the directory `scannedtifs/tiffinfofiles`.

bombala	mulwaree
cessnock	muswellbrook
cooma_monaro	rylstone
crookwell	scone
dungog	singleton
gunning	snowyriver
hunter_region	sydney_basin
merriwa	tumbarumba
murrurundi	yarrowlumla

Table 2.1: LGAs with Agricultural Land Classification data to be converted

# Chapter 3

## STAGE 2

### 3.1 Make a coverage of uniquely-numbered tic points corresponding to the corners of the LGA's map tiles

You can create the AML code to generate a tic coverage for an LGA using this command, though it only produce data for 1:25,000 and 1:50,000 map tiles:

```
lga-coords2tics.sh <LGA-NAME>
```

This example shows how you can make an AML script (`maketics.aml`) to generate all the tics for Bombala shire:

```
lga-coords2tics.sh bombala > maketics.aml
```

Once you have made the AML script, just run it in the correct workspace, where it will create the tic coverage `ticbombalall`, i.e., the coverage's name is the name of the LGA with the prefix `tic` and the suffix `ll` (denote that the coverage is in latitude/longitude coordinates).

The script `lga-coords2tics.sh` calls two scripts:

- `/d8/scripts/lga-map25coords.sh`, and
- `/d8/scripts/lga-map50coords.sh`,

which, in turn, call other scripts (`tile25k-coords.sh`, `tile50k-coords.sh`, `shire25find.sh`, and `shire50find.sh`).

### 3.2 Make map box coverages

#### 3.2.1 When the LGA is covered either by 1:25,000 maps only or by 1:50,000 maps only

In this step, coverages are created which define the bounding boxes of the map tiles which an LGA covers or overlaps. This is simply done in two steps. First, the shell script, `lga-mapboxes.sh`, creates an AML macro, `makebox.aml`, in the current directory. The map box coverages are made by running this macro in ArcInfo. The shell script's usage is:

```
lga-mapboxes.sh <LGA-NAME>
```

For example, `lga-mapboxes.sh bombala`, followed by running `makebox.aml` in ArcInfo to make the box coverages.

Because the map boxes are defined in lat/long coordinates (and on the AGD66 datum), the coverages this procedure makes are also in lat/longs on the AGD66 datum.

### 3.2.2 When the LGA is covered by both 1:25,000 maps and 1:50,000 maps

Some LGAs are covered by a maps at the two scales, e.g., Bombala and Crookwell shires. In this case, two 1:25,000 maps may abut a single side of a 1:50,000 map. The three maps match exactly along the edge when the map boxes are defined just by their corners *and* in geographics. The edges are, of course, just formed by drawing a straight line between corners. Now a corner of each 1:25,000 map falls at the *exact* centre point of the corresponding edge of the 1:50,000 map. Hence these edges coincide.

When the boxes are individually projected to AMGs, then the edges of the 1:25,000 map boxes do *not* coincide with the edge of the 1:50,000 map box. This is because the projection only affects the corner points, and the edges are now formed by straight lines joining the corners *in AMG coordinate space*.

In this project, paper maps represent the projected data in AMG coordinates. The bounding box for a map consists of straight lines in AMG coordinate space. But these straight lines do not project to straight lines in geographics (latitude/longitude) space. Thus although the bounding box has its corners at “nice” latitude/longitude coordinates (whole, half or quarter degrees), its sides do *not* coincide the corresponding lines of latitude and longitude.

Hence, when the map boxes are defined by just their four corner points in geographics, they fit together seamlessly. But when they are projected to an AMG zone, there will be a gap between the box for the 1:50,000 map and those for the 1:25,000 maps. Try it and see! The degree of mismatch is small and falls within the accuracy limits associated with that scale of mapping. Nevertheless, it is troublesome as it introduces slivers when the map boxes are joined together.

It is easy to get around this problem by introducing points exactly at the middle of the edges of a 1:50,000 map box when it is defined by its four corners’ lat/long coordinates. These new points coincide with corners of 1:25,000 maps. Of course, this procedure constructs an (irregular) octagon as the map’s boundary.

A shell script, `lga-mapoctag.sh`, creates an AML macro, `makebox.aml`, in the current directory. When it is run in ArcInfo, it makes boxes for the 1:25,000 map sheets for an LGA, and octagons for the 1:50,000 map sheets. The shell script’s usage is:

```
lga-mapoctag.sh <LGA-NAME>
```

For example, `lga-mapboxes.sh crookwel`, followed by running `makebox.aml` in ArcInfo to make the coverages.

The script is not clever—it introduces mid-points in all the edges of all the 1:50,000 map. I manually edit out the points that are not needed, i.e., the ones in sides not abutting 1:25,000 maps. This is a bit tedious but it makes joining the ALC maps together later one really simple, as there are no slivers to be edited out.

It is wise to check that you have created a seamless map-box coverage for the LGA—see 3.4 below about how I do this.

### 3.2.3 Adding LGA tics

If you want, you can impose a consistent set of tics on all the map boxes (rather than have their tics numbered 1 to 4 in each). A script, `make-add-tics.sh` (see Appendix A.10.2), generates the AML code. You just run the script in the workspace where your coverages are. The script, as written, makes two assumptions:

1. there is a whole-of-shire tic coverage whose name is of the form `tic*ll`;

2. all the coverages you want to impose the tics on have names ending in \*ll.

You can change these conditions quite easily—see the code. The usage of the code is simply:

```
make-add-tics.sh <LGA-NAME>
```

For example, `make-add-tics.sh bombala` assumes there is a coverage containing the tics for the whole shire called `ticbombalall`, and creates AML code like the following:

```
generate bukalon25xll
copytics ticbombalall
q
generate candelo25xll
copytics ticbombalall
q
generate tombong50xll
copytics ticbombalall
q
```

You just capture the output of this command to file, as an AML macro, and run this in ArcInfo.

### 3.3 Convert map boxes to AMG projection

Since the original maps are in AMG projections, it is necessary to project the map boxes and tic coverages to the maps' native projection.

An AML macro, `project-xll.aml` (see Appendix A.11.1), can be run to create another AML which will do the projection. The reason for this two stage process is that some LGAs overlap two AMG zones. `project-xll.aml` creates the ArcInfo commands to project to just one zone, storing the commands in the file `do-projectx.aml`. You can edit this file to change the zone for whichever coverages you need to. The usage of `project-xll.aml` is:

```
&run project-xll <amg-zone-number>
```

where `<amg-zone-number>` must be 54, 55, or 56 (the only zones relevant to us). `project-xll.aml` is set up to re-project all coverages whose names end in `xll`, but you can easily change this. You can also change the name of the output AML file from `do-projectx.aml`. Read the AML macro.

Once you have finished with the original lat/long map box coverages, you can delete them. (You can always re-create them quickly by running the `makebox.aml` macro again.) The AML macro, `kill-xll.aml` (see Appendix A.11.2), will kill all coverages whose names end in `xll` in the current workspace. You can change the letters the coverages end in quite simply—see the code in the Appendix—or use `kill-tail.aml` (see Appendix A.11.2).

### 3.4 Checking map box coverage for an LGA

After you have created the map box coverages in AMG projections, it is wise to check they fit together seamlessly. Assuming you have used the procedure described in the previous sections for making the map boxes, then the box coverages will be the only files/directories in the current directory which end in the characters `25x55` or `50x55`. These three commands will generate an ArcInfo macro which makes one coverages showing all the map boxes joined together. Run them, then run `joinem.aml` in ArcInfo.

```
echo mapjoin lgatiles POLY FEATURES > joinem.aml  
ls -d *25x55 *50x55 >> joinem.aml|  
echo END >> joinem.aml
```

Once you have made the coverage, here called `lgatiles`, display it in ArcView. Look at its table of attributes, and sort into ascending order by area. If there are any sliver polygons, they'll be at the top of the list and obvious by their areas' smallness.

# Chapter 4

## STAGE 3—Raster to Vector Conversion

### 4.1 Installing and checking the TIFF images

The TIFF image files for an LGA's maps are copied from CDROM to the corresponding subdirectory in `scannedtifs`.

The TIFF header information in all the files in one LGA's subdirectory were checked to make sure they were consistent. This was done in two stages. First, the information from the TIFF files for an LGA was listed to a single file. Secondly, this file was processed to present the information so any discrepancies could be spotted easily.

#### 4.1.1 Listing TIFF header information

Change directory to `scannedtifs`. In this directory, there is a script `make-tiffdir-info.sh` (see Appendix A.5.1). To make a list of the header information for every TIFF image in an LGA's directory, run

```
make-tiffdir-info.sh <lga-directory-name>
```

which will create an output file `<lga-directory-name>.tiffinfo`. For example, `make-tiffdir-info.sh bombala` creates `bombala.tiffinfo`, containing the information for all 18 TIFF images for the Bombala LGA. I move all the generated `.tiffinfo` files into the sub-directory `tiffinfofiles`.

(This script lists the TIFF header information using a program, `tiffinfo`, which comes as part of the free `libtiff` library, available on the WWW. `tiffinfo` has been installed into the usual location on a Sun Solaris system (`/usr/local/bin/`).

#### 4.1.2 Checking TIFF header information is consistent

Because I had to write my own thinning program (see Section 4.2 below), I had to check that all the TIFF images had essentially the same format, as I only programmed limited TIFF read/write functionality to handle the TIFF files I was faced with. This subsection describes how I did the checking.

A simple script, `check-tifs.sh` (see Appendix A.5.2), was written to list each type of TIFF tag information from the `.tiffinfo` files. Once you have generated the `.tiffinfo` file for an LGA, and stored it in the directory `tiffinfofiles`, you can present the information conveniently using this script by running

```
check-tifs.sh <lga-directory-name>
```

which lists information for all the TIFF images for that LGA to the screen. You can capture this to file and inspect it at your leisure, or pipe it through the more command; continuing the previous example,

```
check-tifs.sh bombala | more
```

Here is a cut-down version of the output for Bombala, bombala.tiffinfo. Under each TIFF tag heading like Subfile Type or Resolution, you should get a line for each image. This listing shows two entries under each tag heading. In actuality, there are 18 entries, one for each of the 18 TIFF files. You can simply scan this output to see if there are any differences between the files—the only ones I have found were related to image size, under the tag Image Width where the widths and lengths can differ.

```
tiffinfofiles/bombala
```

```
Subfile Type
```

```
Subfile Type: (0 = 0x0)
Subfile Type: (0 = 0x0)
```

```
Image Width
```

```
Image Width: 4944 Image Length: 5566
Image Width: 4944 Image Length: 5534
```

```
Resolution
```

```
Resolution: 150, 150 pixels/inch
Resolution: 150, 150 pixels/inch
```

```
Bits/Sample
```

```
Bits/Sample: 1
Bits/Sample: 1
```

```
Compression Scheme
```

```
Compression Scheme: none
Compression Scheme: none
```

```
Photometric Int
```

```
Photometric Interpretation: min-is-black
Photometric Interpretation: min-is-black
```

```
Thresholding
```

```
Thresholding: bilevel art scan
Thresholding: bilevel art scan
```

```
FillOrder
```

```
FillOrder: msb-to-lsb
FillOrder: msb-to-lsb
```

```
Orientation
```

```
Orientation: row 0 top, col 0 lhs
```

```
Orientation: row 0 top, col 0 lhs

Samples/Pixel
  Samples/Pixel: 1
  Samples/Pixel: 1

Rows/Strip
  Rows/Strip: 512
  Rows/Strip: 512

Min Sample
  Min Sample Value: 0
  Min Sample Value: 0

Max Sample
  Max Sample Value: 1
  Max Sample Value: 1

Planar Config
  Planar Configuration: single image plane
  Planar Configuration: single image plane
```

## 4.2 Thin the scanned images

It is necessary to thin each scanned image to single-pixel wide lines (as far as is possible). I did not have thinning done at the time of scanning as I wished to have the original scanned images available. Unfortunately, ArcInfo's "thinning" function (thin) in its GRID extension is very poor. Its use is difficult as the help file does not say what ESRI mean by the object's "maximum thickness". When used according to the advice in the help file, it does dreadful things to the numerals on the maps which indicate a polygon's land class. For example, it reduces a "3" to a straight-ish line in places, and even breaks open the loop in a "4". Given these problems, which I discovered some time back, I decided to write my own thinning algorithm. There are lots described in the literature, and I selected an iterative morphological method as it is easy to understand; implementing it efficiently was another matter.

I also found that ArcInfo's gridline function did not work well with my thinned images. I therefore made a copy of the thinning program and modified it to remove remaining 'corner' pixels. gridline worked correctly on the 'corner-removed' images. ESRI Australia also provided a work-around which involved using thin on my thinned images in a way inconsistent with their help file's advice.

The script, do\_thin.sh, produces the thinned, corner-removed images as TIFF files, placing them in the thinnedtifs sub-directory. It processes all the images I had copied to disk at the time. It is listed in Appendix A.6.

## 4.3 Convert thinned images to vector coverages

Once the thinned images have been produced, vector linework must be created from them. I did this using ArcInfo's gridline function and partially automated the procedure as two AML

macros, `conv2vector.aml` and `t2v-one.aml`, the code for which is listed in Appendix A.7.1 and A.7.2. At present, you have to edit this file and specify which LGA you are interested in through the variable `thisw`, but that's all you have to change. The calls to both `gridline` and `clean` use conservative values for the `thickness` and `dangle` parameters, corresponding to less than the width of 3 cells (pixels). Since the image resolution is 1/150 in (about 1/6 mm), the width of a pixel corresponds to about 4 metres on the ground at 1:25,000 scale or 8 metres at 1:50,000. Hence, the tolerances used for cleaning correspond to less than 12 m or 25 m. In terms of cutting off dangles, this is definitely conservative because all short dangles are thinning artifacts—they lie within original, unthinned lines.

# Chapter 5

## STAGE 4—Processing Individual Vector Coverages

### 5.1 Edit each coverage

Four steps now need to be performed:

- add georeference tics to coverages;
- clip off map legend, scale bar, etc.;
- correct the topology of each coverage;
- label the polygons.

All steps except the second are manual, and they require the use of ArcEdit.

#### 5.1.1 Add georeference tics to coverages

Add georeferencing tics to a coverage, giving them IDs to match the tics for its map box coverage created in Stage 2. At this stage, I intend to add tics only at a map's corner points as these are the only points for which the geographic coordinates are known for all maps.

Using Arctools; add coverage with tic as edit feature; use Env to display tic labels and lines; use Back env - Image to add original .tif image; use Edit button to set transparent, and select (say) red for line colour. Click Draw, and Apply.

Check visually that the tic marks have the correct Tic-IDs (or IDTIC in the coverages .tic table):

Tic position	Tic label
North-West	1
North-East	2
South-East	3
South-West	4

If the Tic-IDs are not correct, you can change them using calc in tables.

1. Zoom in to one corner, so you can see both the tic mark and the linework in the corner. In the feature selection window, click New. Click on the tic mark, press 9.

In the ... window, click on Move Node. Click on the tic mark (it should already be yellow as it is the currently selected set). Point the cursor to the corner of the map. Press 1. The tic mark will move to this point.

Zoom in on this corner, and repeat the procedure to move the tic to a more precise location. I move it to be one pixel from the inside edge of the boundary in both directions.

In practice, I position the tics roughly on their corners whilst viewing the whole coverage. Then I position each tic finely.

Save. That's it.

### 5.1.2 Clip off map annotation

Now use the 'big box' coverages created in Stage 2. Their tics are the map corner points but their box is slightly larger than the map box. The 'big box' will be used to clip off most of the legend, scale-bars, etc., from the vector coverage.

1. In ArcInfo, make an empty coverage, <newbigbox>, with the same tics as thinned coverage whose tic points you have just correctly positioned.
2. Transform the 'big box' coverage to this empty coverage.
3. Now clip your coverage using this new box.
4. ... and build as line.

I have set up a general-purpose AML macro, clipoff.aml (see Appendix A.8.3), to do this work. You have to edit clipoff.aml manually to specify all the coverages. Here is an example of how to specify the names in it for this Bombala map sheet:

Description	Variable name	Coverage example
The existing AMG over-size box coverage in metric coordinates, correct AMG zone	amgbigbox	bombala25y55
The existing unclipped coverage in scanner coordinates (inches)	unclipped	bombala
A new coverage—the over-size map box in scanner coordinates	bigbox	bombbox
The new clipped coverage you want (unclipped clipped by bigbox)	clipped	bmbv01

### 5.1.3 Correct the topology of each coverage

Polygons and lines can co-exist in a coverage. At this stage, the coverage has been clean-ed and has polygon topology. The coverage has also been build-ed as line. If you display both in ArcView, there may be extra lines (long dangles) that the (deliberate) choice of dangle length in the clean caused not to be deleted. A map may be incorrect in that some lines might be missing, leaving some polygons unclosed. Someone has to ArcEdit the coverages manually to fix up any problems with their topology, such as long dangles. Label points need to be added to the coverage. In summary, the process is:

- identify dangles that need to be corrected (see below);
- manually correct the coverage's topology in ArcEdit;
- delete any text labels embedded in the map, e.g., annotations like significant basaltic influence or township;
- clean and/or build the coverage;
- add polygon labels using the AML macro, `addlabels` (see Appendix A.9.1), with :  
`&run addlabels <coverage>`

### 5.1.3.1 Check for dangles

I have written an AML macro, `dangle-extract.aml` (see Appendix A.8.1), which extracts just the dangling lines from an existing coverage, and saves them to a new coverage:

```
&run dangle-extract <incover> <outcover>
```

This makes it easy to check for errors in topology. Just make the “dangling line” coverage and, in ArcView or ArcEdit, display it over the coverage from which it comes. There will be lots of dangles—namely, all the numerals and annotations denoting the polygons' classes. There may also be dangles which are not errors—for example, they form part of a polygon whose boundary is not properly closed elsewhere. Nevertheless, you should be able to scan the coverage quickly to identify the dangles you need to fix up manually.

## 5.2 Add the polygon attributes to each coverage

Once a coverage has correct polygon topology and the label points have been added, the polygons have to be labelled with their Agricultural Land Classification Class, the only attribute of real interest to us.

### 5.2.1 Define items in coverage's PAT table

An attribute field must be added to the coverage's .PAT table; to ensure consistent definitions, use the simple AML macro, `additems.aml` (see Appendix A.9.2). Its usage in ArcInfo is simply `&run additems <coverage-name>`, for example, `&run additems bombala`. It adds three items to the .PAT: `AGNUM` (defined as 4 5 B), `AGCLASS` (2 2 N 0) and `AGTEXT` (32 32 C).

The `AGNUM` field is initialized to 99 except for the universal polygon which is set to 100. `AGNUM` is an intermediate field which will be copied to `AGCLASS` later, and then deleted. It is needed as ArcEdit appears only to fill polygons with colour according to an attribute value if the attribute is defined as B. (I have tried “colouring” up on `AGCLASS` but this attribute is not given in the pick list as a possible item.)

The `AGCLASS` field is initialized to 0, and `AGTEXT` left blank, except for the universal polygon for which both are set to -2. The `AGTEXT` field is not strictly necessary but can store any annotations or sub-class definitions. It can always be dropped at a later stage.

## 5.2.2 Manually label the polygons

Use ArcEdit to add the correct values for labels to each coverage's PAT table. The editfeature is polygon. In the drawenvironment, for polygon, select ON,FILL,SYMBOLS. . . . Click on SYMBOLS. . . ., and specify items and Agnum; display LINES as well. The polygons will not be filled to start with, but they will be filled as you assign values to their agnum attribute.

Re-build as poly and as line. Check for unlabelled polygons—use stats-make.aml to list statistics of the AGNUM attribute:

```
tables
&run stats-make bomabal04
q
```

View the coverage in ArcView. Identify the ID number of unlabelled polygons and find out their correct labels. It's easiest to add these in tables in ArcInfo, I think.

## 5.3 Georeference and 'box' the coverage

The steps to georeference the coverage you have just corrected and attributed are similar to those in sub-section 5.1.2. Here, you just use the map box coverage created in Stage 2.

1. In ArcInfo, make an empty coverage, <newbox>, with the same tics as the map box coverage:  
create <newbox> <boxcover>
2. Transform the coverage to this empty coverage:  
transform <corrected-cover> <newbox>
3. Now impose the original box's boundary on your coverage<sup>1</sup>:  
identity <boxcover> <newbox> <newcov> poly 0.1
4. Give an AGNUM of 0 to polygons outside the <newbox> coverage but inside the correct map box boundary (boxcover), to make them easy to detect and correct.
5. Eliminate 'unlabelled' polygons (those with AGNUM= 0)—these should all be at the edge of the coverage, and have small areas. I have also put in an area test for the eliminate which requires the polygons to have an area of less than 1000 m<sup>2</sup> if they are to eliminated.

Here is an example of how to specify the names in georef.aml for this Bombala map sheet:

---

<sup>1</sup>Note the use of ArcInfo's **identity** command instead of **clip**; **clip** would not impose the new boundary but just get rid of linework outside it)

Description	Variable name	Coverage example
The existing AMG box coverage in metric coordinates, correct AMG zone	amgbox	bombala25x55
The existing attributed coverage in scanner coordinates (inches)	unref	bombala03
A new coverage—it will contain the georeferenced, attribute coverage	newbox	bombbox
A new coverage—newbox modified to have the correct map box boundary	georef	bmbvg05
A new coverage—georef, with small, unlabelled polygons eliminated	georefel	bmbvg06

## 5.4 Check the coverage against the thinned raster image

At this stage, your coverage should be complete: georeferenced, no dangling lines, all polygons having correct topology, all polygons having correct label points and labels.

Check the correctness of the work done against the original scanned image. Check all polygons have correct topology, that no polygons are missing, and that polygons have the correct labels.

To check the final coverage against the original image, you have to transform it back from AMG coordinates to scanner units, which takes two steps:

```
create <checkcover> <initialcover>
transform <finalcover> <checkcover>
```

The AML macro, `backtrans.aml`, does this for you. You must edit it to specify the coverages. Here is an example of how to specify the names in `back-trans.aml` for the Bombala map sheet:

Description	Variable name	Coverage example
The original vector coverage in scanner coordinates with its tics properly positioned	initialcover	bmbv01
The completed georeferenced and attributed coverage	finalcover	bmbvg07
The new coverage for checking	checkcover	bmbu01

View the `<checkcover>` versus the original .TIF image.

# Chapter 6

## STAGE 5—Making the Shire Maps

### 6.1 Make “shire coverage” ALC map

Join all the attributed, georeferenced, ‘box’-ed and checked coverages for a single LGA into a final “shire coverage” map.

### 6.2 Impose proper shire boundary

We have GIS data defining LGA boundaries in NSW which were supplied by Land And Property Information NSW (Bathurst). The data are cadastrally coincident. It is straightforward to impose this cadastral-quality boundary on the joined coverage.

1. Make a coverage showing just the LGA in question (you may need to delete islands if it's a coastal LGA). Project it to the AMG projection of the ALC coverage.
2. Use ArcInfo's identity command to impose the LGA boundary on the ALC coverage.
3. Check for unlabelled polygons abutting the LGA's border.

### 6.3 Clean up border

#### 6.3.1 Automated cleaning

#### 6.3.2 Manual check and edit

# **Chapter 7**

## **STAGE 6**

### **7.1 compile the metadata statement**

# Appendix A

## Detailed Descriptions and Listings of Scripts

### A.1 Introduction

All the scripts and files discussed in this section are stored in `/d8/canri/scripts/`. In what follows I use the terms “map sheet” and “map tile” interchangeably.

These four scripts were written some time ago for my convenience:

```
shire25find  shire50find
sheet25find  sheet50find
```

These scripts were written specifically to support the CANRI map conversion project:

```
tilek-coords.sh      tilek-grid.sh
tilek-box.sh         tilek-bigbox.sh
lga-mapboxes.sh     lga-mapbigboxes.sh
lga-map25boxes.sh   lga-map50boxes.sh
lga-map25bigboxes.sh lga-map50bigboxes.sh
lga-map25coords.sh  lga-map50coords.sh
lga-tics.sh         make-add-tics.sh
lga-mapoctag.sh     tile50k-octag.sh
```

These ‘one-line’ scripts were also written to support the CANRI map conversion project—they call the complex scripts listed above:

```
tile25k-coords.sh    tile50k-coords.sh
tile25k-grid.sh      tile50k-grid.sh
tile25k-box.sh       tile50k-box.sh
tile25k-box-exactname.sh tile50k-box-exactname.sh
tile25k-bigbox.sh    tile50k-bigbox.sh
tile25k-bigbox-exactname.sh tile50k-bigbox-exactname.sh
```

All coordinates output by these scripts are geographics (lat/longs); at the time of writing, the original data from which the coordinates are derived are for the AGD66 datum.

I have added the directory in which all these scripts are located to the `PATH` variable. A script can be run just typing its name on the command line—remember, you can use the `<ESC>` key in the C Shell and the `<TAB>` key in the bash shell to auto-complete names.

### A.1.1 Capturing output produced by scripts

Most of the scripts write their output directly to the screen. To capture it to file, you can use the Unix re-direction facility when you issue the command. For example,

```
lga-tics.sh bathurst
```

writes code for an AML script to the screen. To capture this to a file (e.g., bathtics.aml), so you can run the AML in ArcInfo later, re-direct the output to this file this way:

```
lga-tics.sh bathurst > bathtics.aml
```

## A.2 Main scripts relevant to both 1:25,000 and 1:50,000 map tiles

### A.2.1 tilek-box.sh

This script produces the AML code to create a rectangular box representing the outline of a named 1:25,000 or 1:50,000 NSW map sheet. The coordinates are degrees longitude and latitude, presently for the AGD66 datum. This is a general purpose script serving the creation of code for the two map scales and for both abbreviated and unabbreviated map sheet names. The tics for the coverage that the AML code creates coincide with the box's corners, and are numbered the same.

Rather than use this script, you would be better advised to you one of the four scripts which call it:

```
tile25k-box.sh           tile50k-box.sh
tile25k-box-exactname.sh tile50k-box-exactname.sh
```

For examples of the output this script produces, see App. A.3.8.

The usage is explained at the start of the code below. The case of the map sheet's name does not matter; it can be upper-, lower- or mixed-case. The optional flag does not need to be given, but if it is it must be a number in the range 1–4. If it is not given, it defaults to 1. Here are four examples of usage which implicitly use the default value of flag:

```
tilek-box.sh 25 0 bathur
tilek-box.sh 50 0 crookwe
tilek-box.sh 25 1 bathurst
tilek-box.sh 50 1 crookwell
```

And here are four examples which produce no output:

```
tilek-box.sh 25 1 bathur
tilek-box.sh 50 1 crookwe
tilek-box.sh 50 1 bathurst
tilek-box.sh 25 1 crookwell
```

The first two don't work because the names are abbreviated and an "exact name" has been specified. The second two don't work because there is no 1:50,000 map sheet for Bathurst, and no 1:25,000 map sheet for Crookwell.

**WARNING:** If the name is abbreviated, and matches two or more tile names, then you will get output for all matched tiles, e.g., `tilek-box.sh 25 0 bat` produces AML code to make the boxes for both Bathurst and Batlow. Note that `tilek-box.sh 25 1 bat` produces no output.

Now for the code:

```
1  #!/bin/sh
2  # Assume the data file is in the same directory as this script.
3  indir=`dirname $0`
4
5  # Multi-purpose script to deal with both 1:25,000 and 1:50,000
6  # map tiles.
7  sz=$1
8
9  if [ $# -lt 3 ] ; then
10 echo "Usage : tilek-box.sh size exact-or-not name flag
```

```

11 where :
12     size is either \"25\" to specify a 1:25,000 map sheet,
13         or \"50\" to specify a 1:50,000 map sheet, and
14     exact-or-not has the value 0 or 1, and indicates
15         0 : map name need not be exact, but can be abbreviated
16         1 : map name must be exact
17     name is the name of the \"${sz}\"K map sheet,
18     flag has the value 1, 2, 3, or 4, and indicates
19         1 : top-left origin and coords listed clockwise
20         2 : top-left origin and coords listed anti-clockwise
21         3 : bottom-left origin and coords listed clockwise
22         4 : bottom-left origin and coords listed anti-clockwise"
23 exit
24 fi
25
26 if [ \"${sz}\" != \"25\" \) -a \"${sz}\" != \"50\" \) ]; then
27     echo \"Map size parameter must be 25 or 50; it is \" ${sz}
28     exit 1
29 fi
30
31 # Match name exactly or not; 0 for not exact, 1 for exact
32 exct=${2}
33
34 if [ \"${exct}\" != \"0\" \) -a \"${exct}\" != \"1\" \) ]; then
35     echo \"Exact match\" parameter must be 0 (non-exact) or 1 (exact); \
36     it is \" ${exct}
37     exit 1
38 fi
39
40 # Check optional flag if it is specified
41
42 if [ $# -eq 4 ]; then
43     if [ $4 -lt 1 -o $4 -gt 4 ]; then
44         echo \"Optional flag value must be in range 1-4. It is $4.
45         exit 1
46     fi
47 fi
48
49 infil=${indir}/tile\"${sz}\"k-coord.txt"
50
51 nawk -v naml=${3} -v flag=${4} -v size=${sz} -v exact=${exct} \
52 'BEGIN{ RS=""
53     FS="\n"
54     OFS = ", "
55 # suffix to add to first seven letters of map sheet name to
56 # give the name of the coverage generated.
57     coversuffix = size "xll"
58 # ensure name is in upper case
59     gsub( /\_/, " ", naml )
60     naml = toupper( naml )
61 # Make pattern ...

```

```

62     if( exact == 0 )    # to match at start of name
63         patt1 = "^" nam1
64     else                # match name exactly
65         patt1 = "^" nam1 ", "
66 # Set default
67     if( flag == "" ) flag = 1
68 #
69 # In the following, "1" signifies minimum value,
70 # and "2" maximum value.
71 # top-left origin, clockwise
72     if( flag == 1 ) {
73         ax[1] = 1; ay[1] = 2
74         ax[2] = 2; ay[2] = 2
75         ax[3] = 2; ay[3] = 1
76         ax[4] = 1; ay[4] = 1
77 # top-left origin, anti-clockwise
78     } else if( flag==2 ) {
79         ax[1] = 1; ay[1] = 2
80         ax[2] = 1; ay[2] = 1
81         ax[3] = 2; ay[3] = 1
82         ax[4] = 2; ay[4] = 2
83 # bottom-left origin, clockwise
84     } else if( flag==3 ) {
85         ax[1] = 1; ay[1] = 1
86         ax[2] = 1; ay[2] = 2
87         ax[3] = 2; ay[3] = 2
88         ax[4] = 2; ay[4] = 1
89 # bottom-left origin, anti-clockwise
90     } else if( flag==4 ) {
91         ax[1] = 1; ay[1] = 1
92         ax[2] = 2; ay[2] = 1
93         ax[3] = 2; ay[3] = 2
94         ax[4] = 1; ay[4] = 2
95     }
96 }
97 match( $0, patt1 ) {
98     split( $1, nm1, ", " )
99     fullnam = nm1[1]
100    gsub( " ", "", nm1[1] )
101    nm2 = tolower( substr( nm1[1],1,7) ) coversuffix
102    print "/* " $1
103    print "generate " nm2
104
105 # Work out the max and min x and y coords, use
106 # them to compute the centroid of the box.
107 # Angles should be no larger than +360 degs nor
108 # smaller than -360 degs.
109    xm[1] = 400; ym[1] = 400; xm[2] = -400; ym[2] = -400;
110    for( i=2; i<=NF; i++ ) {
111        split( $i, x, ", " )
112        if( x[1] < xm[1] ) xm[1] = x[1]

```

```

113     if( x[2] < ym[1] ) ym[1] = x[2]
114     if( x[1] > xm[2] ) xm[2] = x[1]
115     if( x[2] > ym[2] ) ym[2] = x[2]
116 }
117
118 # Force the tics to be in the same order as the corners of
119 # the box...
120 # Put the tics at the corners of the map sheets...
121 print "tics"
122 print "1", xm[ ax[1] ], ym[ ay[1] ]
123 print "2", xm[ ax[2] ], ym[ ay[2] ]
124 print "3", xm[ ax[3] ], ym[ ay[3] ]
125 print "4", xm[ ax[4] ], ym[ ay[4] ]
126 print "END"
127
128 print "polygons"
129 print "1," (xm[2]+xm[1])/2 ", " (ym[2]+ym[1])/2
130 print xm[ ax[1] ], ym[ ay[1] ]
131 print xm[ ax[2] ], ym[ ay[2] ]
132 print xm[ ax[3] ], ym[ ay[3] ]
133 print xm[ ax[4] ], ym[ ay[4] ]
134 print "END"
135 print "END"
136 print "q"
137 print "build " nm2 " poly"
138 print "tables"
139 print "additem " nm2 ".pat name 32 32 C"
140 print "select " nm2 ".pat"
141 # %c prints the ASCII character 35 here, which is the hash
142 # mark. I can not put it here explicitly as it is the awk
143 # comment mark.
144 printf( "resel %s%c > 1\n", nm2, 35 )
145 # %c prints the ASCII character 39 here, which is a
146 # single-quote. I can not put that mark in here explicitly as
147 # it ends the awk code.
148 printf( "move %c%s%c to name\n", 39, fullnam, 39 )
149 print "q"
150 }' $infil

```

## A.2.2 tilek-bigbox.sh

tilek-bigbox.sh serves a very similar purpose to tilek-box.sh except that it produces a box which is 1% larger all around than the box produced by tilek-box.sh. The ways they are invoked are the same.

There are just three differences between the two scripts:

1. '-bigbox' replaces the one occurrence of '-box' (line 6);
2. coversuffix now set by coversuffix = size "y11" (line 56);
3. Immediately after the code to print tic information and before the statement print "polygons" (line 126), code is inserted to make the box bigger by 1% all around:

```

1   # ... but make the box bigger
2   xdiff = ( xm[2] - xm[1] ) * 0.01
3   xm[1] -= xdiff
4   xm[2] += xdiff
5
6   ydiff = ( ym[2] - ym[1] ) * 0.01
7   ym[1] -= ydiff
8   ym[2] += ydiff
9

```

### A.2.3 tilek-coords.sh

This script lists the coordinate pairs of the four corners of the named map tile. At present, the list can start with either the top-left or bottom-left corner, and the corner coordinates can be listed out in a clockwise or anti-clockwise order. The script's usage is:

```
tilek-coords.sh <size> <map name> {optional flag}
```

**WARNING:** If the name is abbreviated, and matches two or more tile names, then you will get output for all matched tiles, e.g., `tilek-coords.sh 25 bat` lists the coordinates of the corners of the map sheets for Bathurst and Batlow. Try also

```
tilek-coords.sh 50 boo.
```

Now for the code:

```

1   #!/bin/sh
2   # Assume the data file is in the same directory as this script.
3   indir=`dirname $0`
4
5   # Multi-purpose script to deal with both 1:25,000 and 1:50,000
6   # map tiles.
7   sz=$1
8
9   if [ $# -lt 2 ] ; then
10  echo "Usage : tilek-coords.sh size name flag
11  where :
12     size is either \"25\" to specify a 1:25,000 map sheet,
13         or \"50\" to specify a 1:50,000 map sheet, and
14     name is the name of the 25K or 50K map sheet, and
15     flag has the value 1, 2, 3, or 4, and indicates
16     1 : top-left origin and coords listed clockwise (default)
17     2 : top-left origin and coords listed anti-clockwise
18     3 : bottom-left origin and coords listed clockwise
19     4 : bottom-left origin and coords listed anti-clockwise"
20  exit 1
21  fi
22
23  if [ \"${sz}\" != \"25\" \] -a [ \"${sz}\" != \"50\" \] ; then
24    echo "Map size parameter must be 25 or 50; it is " ${sz}
25    exit 1
26  fi
27
28  # Check optional flag if it is specified

```

```

29
30 if [ $# -eq 3 ]; then
31     if [ $3 -lt 1 -o $3 -gt 4 ]; then
32         echo Optional flag value must be in range 1-4.  It is $3.
33         exit 1
34     fi
35 fi
36
37 infil=${indir}"/tile${sz}k-coord.txt"
38
39 nawk -v nam1=$2 -v flag=$3 'BEGIN{ RS=""
40     FS="\n"
41     OFS = "\n"
42     # ensure name is in upper case
43     gsub( /\_/, " ", nam1 )
44     nam1 = toupper( nam1 )
45     # Make pattern to match at start of name
46     patt1 = "^" nam1
47     # Set default
48     if( flag == "" ) flag = 1
49     # top-left origin, clockwise
50     if( flag == 1 ) {
51         a[1] = 2
52         a[2] = 4
53         a[3] = 3
54         a[4] = 1
55     # top-left origin, anti-clockwise
56     } else if( flag==2 ) {
57         a[1] = 2
58         a[2] = 1
59         a[3] = 3
60         a[4] = 4
61     # bottom-left origin, clockwise
62     } else if( flag==3 ) {
63         a[1] = 1
64         a[2] = 2
65         a[3] = 4
66         a[4] = 3
67     # bottom-left origin, anti-clockwise
68     } else if( flag==4 ) {
69         a[1] = 1
70         a[2] = 3
71         a[3] = 4
72         a[4] = 2
73     }
74 }
75 NF == 5 && match( $0, patt1 ) {
76     print $1
77     print $(1+a[1])
78     print $(1+a[2])
79     print $(1+a[3])

```

```

80     print $(1+a[4])
81     next
82 }
83 match( $0, patt1 ) { print $0 }' $infil

```

## A.2.4 tilek-grid.sh

This script creates AML code which creates a 1 km grid covering the named 1:25,000 or 1:50,000 map sheet(s). The grid entirely covers the map sheet and is forced to start and end on exact kilometre coordinates. Because map sheets are defined on AMG projections, the created grid is in one the AMG Zone 54, 55, or 56 projections relevant to NSW.

```
tilek-grid.sh <size> <map name>
```

**WARNING:** If the name is abbreviated, and matches two or more tile names, then you will get output for all matched tiles, e.g., `tilek-grid.sh 25 bat` produces AML code to create grids for the map sheets for Bathurst and Batlow. Try `tilek-grid.sh 50 boo` for a voluminous example.

Now for the code:

```

1  #!/bin/sh
2  # Assume the data file is in the same directory as this script.
3  indir=`dirname $0`
4
5  # Multi-purpose script to deal with both 1:25,000 and 1:50,000
6  # map tiles.
7  sz=$1
8
9  if [ \( "${sz}" != "25" \) -a \( "${sz}" != "50" \) ]; then
10     echo "Map size parameter must be 25 or 50; it is " ${sz}
11     exit
12 fi
13
14 infil=${indir}/tile${sz}k-coord.txt"
15
16 if [ $# -lt 2 ] ; then
17 echo "Usage : tilek-grid.sh size name
18 where :
19     size is either \"25\" to specify a 1:25,000 map sheet,
20           or \"50\" to specify a 1:50,000 map sheet, and
21     name is the name of the 25K or 50K map sheet."
22 exit
23 fi
24
25 # The grid spacings in the x- and y-directions in metres.
26 xresol=1000
27 yresol=1000
28
29 nawk -v nam1=$2 -v size=${sz} -v xres=${xresol} -v yres=${yresol} \
30 'BEGIN{ RS=""
31         FS="\n"
32         OFS = ", "

```

```

33 # suffix to add to first seven letters of map sheet name to give
34 # the name of the coverage generated.
35     coversuffix = size "x11"
36 # ensure name is in upper case
37     gsub( /\_/, " ", nam1 )
38     nam1 = toupper( nam1 )
39 # Make pattern to match at start of name
40     patt1 = "^" nam1
41 }
42 match( $0, patt1 ) {
43     split( $1, nm1, "," )
44     fullnam = nm1[1]
45     gsub( " ", "", nm1[1] )
46     nm2 = tolower( substr( nm1[1],1,7) ) coversuffix
47     print "/* " $1
48     print "generate " nm2
49
50 # Work out the max and min x and y coords, use
51 # them to compute the centroid of the box.
52 # Angles should be no larger than +360 degs nor
53 # smaller than -360 degs.
54     xmin = 400; ymin = 400; xmax = -400; ymax = -400;
55     for( i=2; i<=NF; i++ ) {
56         split( $i, x, "," )
57         if( x[1] < xmin ) xmin = x[1]
58         if( x[2] < ymin ) ymin = x[2]
59         if( x[1] > xmax ) xmax = x[1]
60         if( x[2] > ymax ) ymax = x[2]
61     }
62
63 # Force the tics to be in the same order as the corners of the box...
64 # Put the tics at the corners of the map sheets...
65     xcen = (xmax+xmin)/2
66     print "polygons"
67     print "1," xcen "," (ymax+ymin)/2
68     print xmin, ymin
69     print xmin, ymax
70     print xmax, ymax
71     print xmax, ymin
72     print "END"
73     print "END"
74     print "q"
75     print "build " nm2 " poly"
76     print "tables"
77     print "additem " nm2 ".pat name 32 32 C"
78     print "select " nm2 ".pat"
79 # %c prints the ASCII character 35 here, which is the hash
80 # mark. I can not put it here explicitly as it is the nawk
81 # comment mark.
82     printf( "resel %s%c > 1\n", nm2, 35 )
83 # %c prints the ASCII character 39 here, which is a

```

```

84 # single-quote. I can not put that mark in here explicitly as
85 # it ends the nawk code.
86 printf( "move %c%s%c to name\n", 39, fullnam, 39 )
87 print "q"
88 if( xcen >= 138 && xcen < 144 )
89     zone =54
90 else if( xcen >= 144 && xcen < 150 )
91     zone = 55
92 else if( xcen >= 144 && xcen < 150 )
93     zone = 55
94 else if( xcen >= 150 && xcen < 156 )
95     zone = 56
96 else {
97     print "This longitude, " xcen ", is not in an AMG zone covering NSW"
98     exit
99 }
100 nm3 = tolower( substr( nm1[1],1,7) ) size "x" zone
101 nm4 = tolower( substr( nm1[1],1,7) ) size "g" zone
102 print "project cover " nm2 " " nm3 " /d8/macros/lltoamg" zone ".aml"
103 print "build " nm3 " poly"
104 print "&describe " nm3
105 print "&setvar xmn = ~"
106 print " [ truncate [ calc %dsc$xmin% / " xres " ] ] * " xres
107 print "&setvar xmx = ( 1 + ~"
108 print " [ truncate [ calc %dsc$xmax% / " xres " ] ] ) * " xres
109 print "&setvar ymn = ~"
110 print " [ truncate [ calc %dsc$ymin% / " yres " ] ] * " yres
111 print "&setvar ymx = ( 1 + ~"
112 print " [ truncate [ calc %dsc$ymax% / " yres " ] ] ) * " yres
113 print "generate " nm4
114 print "fishnet"
115 print "%xmn%,%ymn%"
116 print "%xmx%,%ymx%"
117 print xres ", " yres
118 print "0"
119 print "%xmx%,%ymx%"
120 print "q"
121 print "build " nm4 " line"
122 }' $infil

```

## A.2.5 lga-mapboxes.sh

Now for the code:

```

1  #!/bin/sh
2
3  # Assume the data file is in the same directory as this script.
4  indir=`echo $0`
5  indir=`dirname $indir`"/"
6
7  if [ -f makebox.aml ] ; then

```

```

8   rm -f makebox.aml
9   fi
10
11  # Translate " " in a map name to "_" to allow for names like
12  # "Lake George" or "Central Tilba",
13  # "North Star" or "Croppa Creek".
14
15  # Ensure it's all done in double precision.
16  # NB precision not set back to whatever it was!
17
18  echo "precision double double" > makebox.aml
19
20  for i in `>${indir}shire25find $1 | cut -d, -f2 | tr " " "\_ "`
21  do
22      echo "${indir}tile25k-box-exactname.sh $i >> makebox.aml"
23      ${indir}tile25k-box-exactname.sh $i >> makebox.aml
24  done
25
26  for i in `>${indir}shire50find $1 | cut -d, -f2 | tr " " "\_ "`
27  do
28      echo "${indir}tile50k-box-exactname.sh $i >> makebox.aml"
29      ${indir}tile50k-box-exactname.sh $i >> makebox.aml
30  done

```

## A.2.6 lga-mapbigboxes.sh

The code for this script is identical to that for `lga-mapboxes.sh` except that the character string `box` is replaced by the string `bigbox` wherever it occurs (and it occurs in only seven lines). Hence, this script calls the scripts `tile25k-bigbox-exactname.sh` and `tile50k-bigbox-exactname.sh`.

## A.2.7 lga-mapoctag.sh

Now for the code:

```

1   #!/bin/sh
2
3   # Assume the data file is in the same directory as this script.
4   indir=`echo $0`
5   indir=`dirname $indir`"/"
6
7   if [ -f makebox.aml ] ; then
8       rm -f makebox.aml
9   fi
10
11  # Translate " " in a map name to "_" to allow for names like
12  # "Lake George" or "Central Tilba",
13  # "North Star" or "Croppa Creek".
14
15  # Ensure it's all done in double precision.
16  # NB precision not set back to whatever it was!

```

```
17
18 echo "precision double double" > makebox.aml
19
20 for i in `>${indir}shire25find $1 | cut -d, -f2 | tr " " "\_"`
21 do
22     echo "${indir}tile25k-box-exactname.sh $i >> makebox.aml"
23     ${indir}tile25k-box-exactname.sh $i >> makebox.aml
24 done
25
26 for i in `>${indir}shire50find $1 | cut -d, -f2 | tr " " "\_"`
27 do
28     echo "${indir}tile50k-octag.sh 1 $i >> makebox.aml"
29     ${indir}tile50k-octag.sh 1 $i >> makebox.aml
30 done
```

## A.3 Scripts relevant to 1:25,000 map tiles

### A.3.1 lga-map25boxes.sh

This script produces the AML code to create the map boxes for the 1:25,000 map sheets that intersect the named shire.

Here is an extract of the output run for Bombala Shire, produced by the command `lga-map25boxes.sh bombala`. See that it treats correctly map tiles whose names are more than one word long.

```
/* BUKALONG,M8724N3
generate bukalon25x11
polygons
1,149.125,-36.8125
149,-36.750000
149.250000,-36.750000
149.250000,-36.875000
149,-36.875000
END
END
q
build bukalon25x11 poly
tables
additem bukalon25x11.pat name 32 32 C
select bukalon25x11.pat
resel bukalon25x11# > 1
move 'BUKALONG' to name
q
/* MOUNT TENNYSON,M8723S4
generate mountte25x11
polygons
1,149.125,-37.1875
149,-37.125000
149.250000,-37.125000
149.250000,-37.250000
149,-37.250000
END
END
q
build mountte25x11 poly
tables
additem mountte25x11.pat name 32 32 C
select mountte25x11.pat
resel mountte25x11# > 1
move 'MOUNT TENNYSON' to name
q
```

The code:

```
1  #!/bin/sh
```

```

2  # Assume the data file is in the same directory as
3  # this script.
4  indir=`dirname $0`"/"
5
6  if [ -f makebox.aml ] ; then
7      rm -f makebox.aml
8  fi
9
10 # Translate " " in a map name to "_" to allow for names like
11 # "Lake George" or "Central Tilba"
12
13 # Ensure it's all done in double precision.
14 # NB precision not set back to whatever it was!
15
16 echo "precision double double" > makebox.aml
17
18 for i in `>${indir}shire25find $1 | cut -d, -f2 | tr " " "\_"`
19 do
20     echo "${indir}tile25k-box-exactname.sh $i >> makebox.aml"
21     ${indir}tile25k-box-exactname.sh $i >> makebox.aml
22 done

```

### A.3.2 lga-map25bigboxes.sh

The code for this script is identical to that for `lga-map25boxes.sh` except that the character string `box` is replaced by the string `bigbox` wherever it occurs (and it occurs in only five lines). Hence, this script calls the script `tile25k-bigbox-exactname.sh`.

### A.3.3 lga-map25coords.sh

`lga-map25coords.sh` uses `shire25find` to extract (from the file `shire25mapsheets.txt`) the names of the map tiles which are covered by or overlapped by the named LGA. For each map tile which is named, it then runs `tile25k-coords.sh` to extract the coordinates of the corners of that map tile. The coordinates are listed from the top-left corner in a clockwise order. To list the corner coordinates of the tiles covering Evans Shire, try

```
lga-map25coords.sh evan
```

The name can be abbreviated and in lower-, upper- or mixed case. You should give only one name as any on the command line after the first are ignored. Here is part of the output generated by this example:

```

ARKSTONE,M8829N4
149.500000,-34
149.750000,-34
149.750000,-34.125000
149.500000,-34.125000
BATHURST,M8831S3
149.500000,-33.375000
149.750000,-33.375000
149.750000,-33.500000

```

```
149.500000,-33.500000
WATTLE FLAT,M8831S4
149.500000,-33.125000
149.750000,-33.125000
149.750000,-33.250000
149.500000,-33.250000
```

The code:

```
1  #!/bin/sh
2
3  # Assume the data file is in the same directory as
4  # this script.
5  indir=`echo $0`
6  indir=`dirname $indir`"/"
7
8  # Translate " " in a map name to "_" to allow for names like
9  # "Lake George" or "Central Tilba"
10
11 for i in `>${indir}shire25find $1 | cut -d, -f2 | tr " " "\_ "`
12 do
13 #   j=`echo $i | tr '[:upper:]' '[:lower:]'`
14 #   echo "tile25k-box.sh $i > ${j}box.aml"
15 #   tile25k-box.sh $i > ${j}box.aml
16 #   echo "tile25k-coords.sh $i
17     ${indir}tile25k-coords.sh $i
18 done
```

### A.3.4 shire25find

This script lists the names of the map sheets which a shire covers or overlaps. Its usage is:

```
shire25find <shirename> ...
```

where you can specify one or more shires. If you want to list which map sheets cover, say, Blayney, try this:

```
shire25find blay
```

which produces this output:

```
BLAYNEY,CARCOAR,8730-N-4
BLAYNEY,HOBBYS YARDS,8730-S-1
BLAYNEY,MANDURAMA,8730-S-4
BLAYNEY,MILLTHORPE,8731-S-3
BLAYNEY,NEWBRIDGE,8730-N-1
BLAYNEY,VITTORIA,8731-S-2
```

Again, you can abbreviate the name, and case does not matter. You can use it to find the sheets covering several shires at once, for example:

```
shire25find cabonn evans blayne
```

shire25find is a very simple shell script implementing a grep command, where the \$1 below gets the value blay in the first example above:

```
grep -i ^$1 shire25mapsheets.txt
```

The -i switch tells grep to ignore the case of the word. The ^ tells it that the word must be at the start of the line in the file (so you'll get Carrathool but not Wingecarribee with shire25find carr). Try grep -i winge shire25mapsheets.txt and grep -i ^winge shire25mapsheets.txt.

The code for this script:

```
1  #!/bin/sh
2  # Assume the data file is in the same directory as
3  # this script.
4  indir=`dirname $0`
5  #
6  for i in "$@"
7  do
8    grep -i "^$i"  ${indir}/shire25mapsheets.txt
9  done
```

### A.3.5 Producing the file shire25mapsheets.txt

The file shire25mapsheets.txt lists, by shire, which 1:25,000 map sheets are within or overlap each shire's boundary. Each line of the file has three fields: shire name, map sheet name, map sheet number. The data is in a comma-separated form.

I produced the data by unioning /d21/final\_state\_data/aanswlicll/nsw\_lga, the LGA coverage, with a 1:25,000 map sheet coverage, /d21/final\_state\_data/25-mapshll, then unloading the attributes ATTR1 (the LGA name), NAME, and TILE-NAME, to a file. This was processed using the ArcInfo and Unix commands shown below.

```
1  arc
2  union /d21/final_state_data/aanswlicll/nsw_lga ~
3    /d21/final_state_data/25_mapshll maplga 0.0000001
4  tables
5  select maplga.pat
6  unload maplga.txt ATTR1 NAME TILE-NAME
7  q
8  q
9
10 # Delete single quote marks.
11 tr -d "\047" < maplga.txt > maplga.tx2
12
13 # Get rid of blank entries, and re-format the map's
14 # numerical designator.
15 nawk 'BEGIN{FS=",";OFS=","}
16 $1 != "" && $1 != "UNINCORPORATED" && $2 != "" && $3 != "NA" {
17 a=substr($3,2,4)
18 b=substr($3,6,1)
19 c=substr($3,7,1)
20 print $1,$2,a "-" b "-" c }' maplga.tx2 > maplga.tx3
21
22 # Sort, and get rid of any duplicate entries.
```

```
23 sort maplga.tx3 | uniq > shire25mapsheets.txt
24 rm maplga.txt maplga.tx1 maplga.tx2 maplga.tx3
```

### A.3.6 sheet25find

This script lists the names of the shires which named map sheets overlap. Its usage is:

```
sheet25find <sheetname> ...
```

e.g., `sheet25find wingello` or `sheet25find orange bathurst`. The latter example generates this output:

```
CABONNE,ORANGE,8731-N-3
ORANGE,ORANGE,8731-N-3
BATHURST,BATHURST,8831-S-3
EVANS,BATHURST,8831-S-3
OBERON,BATHURST,8831-S-3
```

The code for this script:

```
1  #!/bin/sh
2  # Assume the data file is in the same directory as
3  # this script.
4  indir=`dirname $0`
5  #
6  for i in "$@"
7  do
8      grep -i ",$i" ${indir}/shire25mapsheets.txt
9  done
```

### A.3.7 tile25k-coords.sh

The code:

```
1  /d8/canri/scripts/tilek-coords.sh 25 $1 $2
```

This script lists the coordinate pairs of the four corners of the named map tile. At present, the list can start with either the top-left or bottom-left corner, and the corner coordinates can be listed out in a clockwise or anti-clockwise order. The script's usage is:

```
tile25k-coords.sh <map name> {optional flag}
```

As with the other scripts, the case of the map name does not matter, and it can be abbreviated. See ??? for details about the value of the optional flag.

Here is the output of `tile25k-coords.sh bathurs`

```
BATHURST,M8831S3
149.500000,-33.375000
149.750000,-33.375000
149.750000,-33.500000
149.500000,-33.500000
```

and of `tile25k-coords.sh bathurs 4`

```
BATHURST,M8831S3
149.500000,-33.500000
149.750000,-33.500000
149.750000,-33.375000
149.500000,-33.375000
```

### A.3.8 tile25k-box.sh

The code:

```
1 /d8/canri/scripts/tilek-box.sh 25 0 $1 $2
```

This script generates the AML code to build a coverage representing a map tile's bounding box. The coverage consists of a single polygon, with its label point in its middle, labelled with the map sheet's name (as the attribute name in the coverage's .pat table). At present, the polygon can start with either the top-left or bottom-left corner, and the corner coordinates can be listed out in a clockwise or anti-clockwise order. The script's usage is:

```
tile25k-box.sh <map name> {optional flag}
```

You can write the generated code to a file (called, e.g., my-file.aml) using:

```
tile25k-box.sh <map name> {optional flag} > my-file.aml
```

As with the other scripts, the case of the map name does not matter, and it can be abbreviated. **WARNING:** If the name is abbreviated, and matches two or more tile names, then you will get output for all matched tiles—see tile25k-box-exactname.sh below. See ??? for details about the value of the optional flag.

Here is the output of tile25k-box.sh bath

```
/* BATHURST,M8831S3
generate bathurs25x11
polygons
1,149.625,-33.4375
149.500000,-33.375000
149.750000,-33.375000
149.750000,-33.500000
149.500000,-33.500000
END
END
q
build bathurs25x11 poly
tables
additem bathurs25x11.pat name 32 32 C
select bathurs25x11.pat
resel bathurs25x11# > 1
move 'BATHURST' to name
q
```

### A.3.9 tile25k-bigbox.sh

```
1 /d8/canri/scripts/tilek-bigbox.sh 25 0 $1 $2
```

This script generates the AML code to build a coverage representing a map tile's bounding box. The coverage consists of a single polygon, with its label point in its middle, labelled with the map sheet's name (as the attribute name in the coverage's .pat table). At present, the polygon can start with either the top-left or bottom-left corner, and the corner coordinates can be listed out in a clockwise or anti-clockwise order. The script's usage is:

### A.3.10 tile25k-box-exactname.sh and tile25k-bigbox-exactname.sh

tile25k-box-exactname.sh

```
1 /d8/canri/scripts/tilek-box.sh 25 1 $1 $2
```

tile25k-bigbox-exactname.sh

```
1 /d8/canri/scripts/tilek-bigbox.sh 25 1 $1 $2
```

These scripts' functions are identical to tile25k-box.sh's and tile25k-bigbox.sh's respectively except that you must specify the tile's name *in full*, that is unabbreviated, though its case can still be mixed.

If the tile's name is more than one word long, then join the words using an underscore, e.g.,

```
tile25k-box-exactname.sh botany_bay
tile25k-box-exactname.sh cherry_tree_hill
tile25k-bigbox-exactname.sh willi_willi
```

### A.3.11 tile25k-grid.sh

The code:

```
1 /d8/canri/scripts/tilek-grid.sh 25 $1
```

This script generates the AML code to build a coverage representing a map tile's bounding box. The coverage consists of a single polygon, with its label point in its middle, labelled with the map sheet's name (as the attribute name in the coverage's .pat table). At present, the polygon can start with either the top-left or bottom-left corner, and the corner coordinates can be listed out in a clockwise or anti-clockwise order. The script's usage is:

```
tile25k-grid.sh <map name> {optional flag}
```

You can write the generated code to a file (called, e.g., my-file.aml) using:

```
tile25k-grid.sh <map name> {optional flag} > my-file.aml
```

## A.4 Scripts relevant to 1:50,000 map tiles

There is only one script specific to this scale, `tile50k-octag.sh`, which is discussed in detail below.

The eleven other scripts named below accomplish the same ends for the 1:50,000 map tiles as the corresponding scripts in the previous section do for the 1:25,000 map tiles.

They use the data contained in `shire50mapsheets.txt` and `tile50k-coord.txt`. Where the equivalently named scripts for the 1:25,000 tiles have “25”, these scripts have “50”, and that is the only difference between them but for some names of map sheets in comment lines. (I have checked this by comparing each pair of files using the Unix `diff` command.)

1. `lga-map50boxes.sh`
2. `lga-map50bigboxes.sh`
3. `lga-map50coords.sh`
4. `shire50find`
5. `sheet50find`
6. `tile50k-coords.sh`
7. `tile50k-box.sh`
8. `tile50k-box-exactname.sh`
9. `tile50k-bigbox.sh`
10. `tile50k-bigbox-exactname.sh`
11. `tile50k-grid.sh`

### A.4.1 `tile50k-octag.sh`

The code for this script is very similar to that for `tilek-box.sh`. It is effectively `tilek-box.sh` specialized for use with only 1:50,000 map sheets, and with four lines inserted to write out mid-points of edges. These are the four lines numbered 127, 130, 133, and 136 in the following code:

```
1  #!/bin/sh
2  # Assume the data file is in the same directory as this script.
3  indir=`dirname $0`
4
5  # This script makes an AML to construct an octagonal box for a
6  # 1:50,000 map tile but puts mid-points into its sides so that
7  # it matches any 1:25,000 map tile to which it is adjacent.
8
9  # Specify 1:50,000 map tiles.
10 sz="50"
11
12 if [ $# -lt 2 ] ; then
13 echo "Usage : tile50k-octag.sh exact-or-not name flag
14 where :
15     exact-or-not has the value 0 or 1, and indicates
16     0 : map name need not be exact, but can be abbreviated
17     1 : map name must be exact
18     name is the name of the \"${sz}\"K map sheet,
19     flag has the value 1, 2, 3, or 4, and indicates
```

```

20     1 : top-left origin and coords listed clockwise
21     2 : top-left origin and coords listed anti-clockwise
22     3 : bottom-left origin and coords listed clockwise
23     4 : bottom-left origin and coords listed anti-clockwise"
24 exit
25 fi
26
27 # Match name exactly or not; 0 for not exact, 1 for exact
28 exct=$1
29
30 if [ \( "${exct}" != "0" \) -a \( "${exct}" != "1" \) ]; then
31     echo "\"Exact match\" parameter must be 0 (non-exact) or 1 (exact); \"
32     it is \" ${exct}
33     exit 1
34 fi
35
36 # Check optional flag if it is specified
37
38 if [ $# -eq 3 ]; then
39     if [ $3 -lt 1 -o $3 -gt 4 ]; then
40         echo Optional flag value must be in range 1-4. It is $3.
41         exit 1
42     fi
43 fi
44
45 infil=${indir}"/tile"${sz}"k-coord.txt"
46
47 nawk -v nam1=$2 -v flag=$3 -v size=${sz} -v exact=${exct} \
48 'BEGIN{ RS=""
49     FS="\n"
50     OFS = ", "
51 # suffix to add to first seven letters of map sheet name to
52 # give the name of the coverage generated.
53     coversuffix = size "xll"
54 # ensure name is in upper case
55     gsub( /\_/, " ", nam1 )
56     nam1 = toupper( nam1 )
57 # Make pattern ...
58     if( exact == 0 ) # to match at start of name
59         patt1 = "^" nam1
60     else # match name exactly
61         patt1 = "^" nam1 ", "
62 # Set default
63     if( flag == "" ) flag = 1
64 #
65 # In the following, "1" signifies minimum value,
66 # and "2" maximum value.
67 # top-left origin, clockwise
68     if( flag == 1 ) {
69         ax[1] = 1; ay[1] = 2
70         ax[2] = 2; ay[2] = 2

```

```

71         ax[3] = 2; ay[3] = 1
72         ax[4] = 1; ay[4] = 1
73 # top-left origin, anti-clockwise
74     } else if( flag==2 ) {
75         ax[1] = 1; ay[1] = 2
76         ax[2] = 1; ay[2] = 1
77         ax[3] = 2; ay[3] = 1
78         ax[4] = 2; ay[4] = 2
79 # bottom-left origin, clockwise
80     } else if( flag==3 ) {
81         ax[1] = 1; ay[1] = 1
82         ax[2] = 1; ay[2] = 2
83         ax[3] = 2; ay[3] = 2
84         ax[4] = 2; ay[4] = 1
85 # bottom-left origin, anti-clockwise
86     } else if( flag==4 ) {
87         ax[1] = 1; ay[1] = 1
88         ax[2] = 2; ay[2] = 1
89         ax[3] = 2; ay[3] = 2
90         ax[4] = 1; ay[4] = 2
91     }
92 }
93 match( $0, patt1 ) {
94     split( $1, nm1, "," )
95     fullnam = nm1[1]
96     gsub( " ", "", nm1[1] )
97     nm2 = tolower( substr( nm1[1],1,7) ) coversuffix
98     print "/* " $1
99     print "generate " nm2
100
101 # Work out the max and min x and y coords, use
102 # them to compute the centroid of the box.
103 # Angles should be no larger than +360 degs nor
104 # smaller than -360 degs.
105 xm[1] = 400; ym[1] = 400; xm[2] = -400; ym[2] = -400;
106 for( i=2; i<=NF; i++ ) {
107     split( $i, x, "," )
108     if( x[1] < xm[1] ) xm[1] = x[1]
109     if( x[2] < ym[1] ) ym[1] = x[2]
110     if( x[1] > xm[2] ) xm[2] = x[1]
111     if( x[2] > ym[2] ) ym[2] = x[2]
112 }
113
114 # Force the tics to be in the same order as the corners of
115 # the box...
116 # Put the tics at the corners of the map sheets...
117 print "tics"
118 print "1", xm[ ax[1] ], ym[ ay[1] ]
119 print "2", xm[ ax[2] ], ym[ ay[2] ]
120 print "3", xm[ ax[3] ], ym[ ay[3] ]
121 print "4", xm[ ax[4] ], ym[ ay[4] ]

```

```

122     print "END"
123
124 # Add mid-points of lines
125     print "polygons"
126     print "1," (xm[2]+xm[1])/2 "," (ym[2]+ym[1])/2
127
128     print xm[ ax[1] ], ym[ ay[1] ]
129     print (xm[ ax[1] ]+xm[ ax[2] ])/2, (ym[ ay[1] ]+ym[ ay[2] ])/2
130
131     print xm[ ax[2] ], ym[ ay[2] ]
132     print (xm[ ax[2] ]+xm[ ax[3] ])/2, (ym[ ay[2] ]+ym[ ay[3] ])/2
133
134     print xm[ ax[3] ], ym[ ay[3] ]
135     print (xm[ ax[3] ]+xm[ ax[4] ])/2, (ym[ ay[3] ]+ym[ ay[4] ])/2
136
137     print xm[ ax[4] ], ym[ ay[4] ]
138     print (xm[ ax[4] ]+xm[ ax[1] ])/2, (ym[ ay[4] ]+ym[ ay[1] ])/2
139
140     print "END"
141     print "END"
142     print "q"
143     print "build " nm2 " poly"
144     print "tables"
145     print "additem " nm2 ".pat name 32 32 C"
146     print "select " nm2 ".pat"
147 # %c prints the ASCII character 35 here, which is the hash
148 # mark. I can not put it here explicitly as it is the awk
149 # comment mark.
150     printf( "resel %s%c > 1\n", nm2, 35 )
151 # %c prints the ASCII character 39 here, which is a
152 # single-quote. I can not put that mark in here explicitly as
153 # it ends the awk code.
154     printf( "move %c%s%c to name\n", 39, fullnam, 39 )
155     print "q"
156 }' $infil

```

## A.5 Scripts for checking TIFF header information

### A.5.1 make-tiffdir-info.sh

```
1  indir=${1%/}
2  echo ${indir}
3  #PROGRAM=/d22/rogerr/free/tiff-v3.5.5/tools/tiffinfo
4  PROGRAM=tiffinfo
5
6  (for i in `ls ${indir}/*.tif`
7  do
8      echo $i
9      ${PROGRAM} $i
10     echo
11     echo
12 done ) > ${indir}.tiffinfo
```

### A.5.2 check-tifs.sh

```
1  indir=tiffinfofiles/${1%/}
2  echo ${indir}
3
4  for var in "Subfile Type" \
5  "Image Width" \
6  "Resolution" \
7  "Bits/Sample" \
8  "Compression Scheme" \
9  "Photometric Int" \
10 "Thresholding" \
11 "FillOrder" \
12 "Orientation" \
13 "Samples/Pixel" \
14 "Rows/Strip" \
15 "Min Sample" \
16 "Max Sample" \
17 "Planar Config"
18 do
19     echo
20     echo $var
21     egrep "$var" ${indir}.tiffinfo
22 done
```

## A.6 Scripts for thinning images

do\_thin.sh

```
1  DIRLIST="bombala cessnock cooma_monaro crookwell gunning \  
2          hunter_region mulwaree muswellbrook rylstone \  
3          snowyriver"  
4  
5  TMPDIR="/d22/rogerr/thin/Images"  
6  
7  #OUTDIR="/d8/canri/thinnedtifs"  
8  OUTDIR="/d8/canri/t2innedtifs"  
9  
10 rootdir=`pwd`  
11  
12 #THINPROG=${rootdir}/thinner  
13 THINPROG=/d22/rogerr/thin/Versions/m15pre  
14 REMCPROG=${rootdir}/removecorners  
15  
16 for i in ${DIRLIST}  
17 do  
18     echo  
19     echo  
20     cd $i  
21     for j in *.tif  
22     do  
23         echo $i/$j  
24         k=${j%.tif}  
25  
26         ${THINPROG} $j ${TMPDIR}/${k}.1 ${TMPDIR}/${k}.2 0 72  
27  
28 # Identify which file the thinned image is in; the other file  
29 # has been deleted by the thinning program. Remove corners  
30 # from thinned image so that Arc/Info's GRIDLINE function  
31 # works better.  
32 # N.B. "removecorners" needs a couple of dummy parameters on  
33 # its command line, here "0 72"  
34  
35 # Add "01" suffix to thinned, corner-removed image's  
36 # file name.  
37  
38     if [ -f ${TMPDIR}/${k}.1 ] ; then  
39         ${REMCPROG} ${TMPDIR}/${k}.1 \  
40             ${OUTDIR}/${i}/${k}01.tif ${TMPDIR}/${k}.3 0 72  
41         rm -f ${TMPDIR}/${k}.{1,3}  
42         echo WROTE THINNED IMAGE : ${OUTDIR}/${i}/${k}01.tif  
43  
44     elif [ -f ${TMPDIR}/${k}.2 ] ; then  
45         ${REMCPROG} ${TMPDIR}/${k}.2 \  
46             ${OUTDIR}/${i}/${k}01.tif ${TMPDIR}/${k}.3 0 72  
47         rm -f ${TMPDIR}/${k}.{2,3}  
48         echo WROTE THINNED IMAGE : ${OUTDIR}/${i}/${k}01.tif
```

```
49
50     else
51         echo "Could not find file for image " $i/$j
52         echo '${TMPDIR}/${k}.1 : ' ${TMPDIR}/${k}.1
53         echo '${TMPDIR}/${k}.2 : ' ${TMPDIR}/${k}.2
54     fi
55
56 done
57 cd ..
58 done
```

## A.7 Scripts for converting images to vector coverages

### A.7.1 conv2vector.aml

conv2vector.aml is the script to convert the thinned images to vector coverages; it is in the directory /d8/canri/thinnedtifs.

```
1  &if ^ [ exists . -workspace ] &then
2  &do
3    &type Creating workspace here ...
4    cw .
5  &end
6
7  &setvar here = [ show &workspace ]
8  &type %here%
9
10 &setvar macroloc = /d8/canri/thinnedtifs/
11 &setvar thisw = /d8/canri/thinnedtifs/cooma_monaro
12
13 &if ^ [ exists %thisw% -workspace ] &then
14 &do
15   &type Creating workspace ... %thisw%
16   cw %thisw%
17 &end
18
19 workspace %thisw%
20
21 &if [ exists outfile -file ] &then
22   &setvar ok = [ delete outfile -file ]
23
24 &setvar count = [ filelist *.tif outfile -file ]
25
26 &if %count% <= 0 &then
27   &return Error generating output file in workspace %thisw%.
28
29 &setvar unit = [ open outfile ok -read ]
30
31 &do I = 1 &to %count%
32   &setvar intiff = [ before [ read %unit% ok ] .tif ]
33   &setvar outgrid = [ before %intiff% 01 ]
34   &if [ length %outgrid% ] > 9 &then
35     &setvar outgrid = [ substr %outgrid% 1 9 ]
36   /* Convert thinned grid to coverage.
37     &run %macroloc%t2v-one %intiff% %outgrid%
38 &end
39
40 &setvar ok = [ close %unit% ]
41 &setvar ok = [ delete outfile -file ]
42
43 workspace %here%
```

## A.7.2 t2v-one.aml

```
1  &args intiff outgrid
2
3  &type intiff - %intiff%      outgrid - %outgrid%
4
5  &setvar outcov = %outgrid%
6
7  imagegrid %intiff% %outgrid%
8
9  /* The images have a resolution of 150 dpi which gives a cell
10 /* size of 0.0067.  In converting to lines, use tolerances of
11 /* about two and three cells.
12  gridline %outgrid% %outcov%01 POSITIVE NOTHIN NOFILTER ~
13  SHARP # 0.017 0.012 0.0
14  clean %outcov%01 %outcov%02 0.012 0.0195 poly
15  build %outcov%02 line
16  kill %outgrid%
17 /* Remove pseudo-nodes
18  arccedit
19  editcoverage %outcov%02
20  editfeature line
21  select all
22  unsplit none
23  save
24  quit
25 /* Re-build ...
26  build %outcov%02 poly
27  build %outcov%02 line
28
29 /* Clean up ...
30  kill %outcov%01 all
31  rename %outcov%02 %outcov%
32
33 &type %outcov% is the cleaned, thinned coverage.
34
35 &label END_OF_AML
```

## A.8 Scripts to help create topologically clean coverages

### A.8.1 Extracting dangling lines to a new coverage

This AML macro, `dangle-extract.aml`, extracts dangling lines from one coverage, putting the dangling lines into a new coverage, and building it as lines. Its usage is `&run dangle-delete <incov> <outcov>`.

```
1  &args incov outcov
2  /* Extract all (and only) dangles from a coverage, unsplits
3  /* the lines to remove pseudo-nodes, and re-builds lines.
4  /* The purpose of this AML macro is to create a coverage you
5  /* can easily check to find dangles you need to keep!
6  &if [null %incov%] or [null %outcov%] &then
7    &return Usage : dangle-extract <incov> <outcov>
8  arccedit
9  editcoverage %incov%
10 editfeature line
11 select lpoly# <> rpoly#
12 delete
13 nsel
14 unsplit none
15 save %outcov%
16 quit
17
18 build %outcov% line
```

### A.8.2 Deleting dangling lines from a coverage

This AML macro, `dangle-delete.aml`, deletes dangling lines from one coverage, putting the polygon linework into a new coverage, and building it. Its usage is `&run dangle-delete <incov> <outcov>`.

```
1  &args incov outcov
2
3  /* Deletes all dangles from a coverage, unsplits the lines
4  /* to remove pseudo-nodes, and re-builds polygons and lines.
5
6  &if [null %incov%] or [null %outcov%] &then
7    &return Usage : dangle-delete <incov> <outcov>
8
9  arccedit
10 editcoverage %incov%
11 editfeature line
12 select lpoly# = rpoly#
13 delete
14 nsel
15 unsplit none
16 save %outcov%
17 quit
18
```

```
19 build %outcov% poly
20 build %outcov% line
```

### A.8.3 Clipping off

The script is `clipoff.aml`, and its usage is simply `&run clipoff`. You must, though, edit the macro and specify the correct values for the four variables `amgbigbox`, `unclipped`, `bigbox`, and `clipped`.

```
1  &args dummy
2  &if ^ [null %dummy%] &then
3  &do
4    &type "clipoff" does not take arguments.
5    &type Edit file and specify coverages.
6    &return
7  &end
8
9  /* These two coverages must already exist...
10 /* The over-size AMG box in metric coordinates,
11 /* correct AMG zone.
12 &setvar amgbigbox = kydra25y55
13
14 &setvar prfx = kydr
15
16 /* The unclipped coverage in scanner coordinates (inches).
17 &setvar unclipped = %prfx%01
18
19 /* This AML macro creates these two coverages; they must not
20 /* exist on entry.
21 &setvar bigbox = %prfx%bigbox
22 &setvar clipped = %prfx%02
23
24
25 /* **** You should not need to change anything below here ****
26
27
28 &if ( ^ [ exists %amgbigbox% -cover ] ) OR ~
29     ( ^ [ exists %unclipped% -cover ] ) &then
30 &do
31   &type These coverages must both exist : ~
32   %amgbigbox% and %unclipped%.
33   &type At least one does not.
34   &return
35 &end
36
37 &if [ exists %bigbox% -cover ] OR ~
38     [ exists %clipped% -cover ] &then
39 &do
40   &type Neither of these coverages must exist : ~
41   %bigbox% and %clipped%.
42   &type At least one does.
```

```
43     &return
44 &end
45
46 create %bigbox% %unclipped%
47
48 /* Write the output of the 'transform' to a watch file
49 /* for the record.
50 &watch %unclipped%.transf.wat
51 transform %amgbigbox% %bigbox%
52 &watch &off
53
54 clip %unclipped% %bigbox% %clipped% poly 0.0001
55 build %clipped% line
56 kill %bigbox% all
57
58 &label END_OF_AML
```

## A.9 Scripts for processing topologically clean coverages

### A.9.1 Adding label points to cleaned coverages

The script is `addlabels.aml`, and its usage is simply `&run addlabels <coverage>`.

```
1  &args coverage
2  /* The ArcInfo help for CREATELABELS says:
3  /*  "If you are creating labels for a new coverage which does
4  /*    not yet contain attributes, use an \{id\_base\} of 0 and
5  /*    then run BUILD"
6
7  createlabels %coverage% 0
8  build %coverage% poly
9  build %coverage% line
10 /* Check all is OK - the universal polygon is
11 /* probably unlabelled.
12 labelerrors %coverage% all
```

### A.9.2 Adding items to coverage PAT table

The script is `additems.aml`, and its usage is simply `&run additems <coverage>`.

```
1  &args covername
2
3  additem %covername%.pat %covername%.pat AGNUM 4 5 B
4  additem %covername%.pat %covername%.pat AGCLASS 2 2 N 0
5  additem %covername%.pat %covername%.pat AGTEXT 32 32 C
6
7  /* Initialize the table entries; leave AGTEXT blank but
8  /* for the universal polygon.
9
10 tables
11 select %covername%.pat
12 resel %covername%# = 1
13 calc agnum = 100
14 calc agclass = -2
15 move '-2' to agtext
16 nsel
17 calc agnum = 99
18 calc agclass = 0
19 q
20
21 &label END_OF_AML
```

### A.9.3 Georeferencing

The script is `georef.aml`, and its usage is simply `&run georef`. You must, though, edit the macro and specify the correct values for the five variables `amgbox`, `unref`, `newbox`, `georefand` and `georefel`.

```

1  /* These two coverages must already exist...
2
3  /* The AMG box in metric coordinates, correct AMG zone.
4  &setvar amgbox = kydra25x55
5
6  &setvar prfx = kydr
7
8  /* The attributed coverage in scanner coordinates (inches).
9  &setvar unref = %prfx%05
10
11 &if ( ^ [ exists %amgbox% -cover ] ) OR ~
12     ( ^ [ exists %unref% -cover ] ) &then
13 &do
14     &type These coverages must both exist : %amgbox% and ~
15     %unref%.
16     &type At least one does not.
17     &return
18 &end
19
20
21 /* This AML macro creates these three coverages; they must not
22 /* exist on entry.
23
24 &setvar newbox = %prfx%box
25 &setvar georef = %prfx%g06
26 &setvar georefel = %prfx%g07
27
28 &if [ exists %newbox% -cover ] OR [ exists %georef% -cover ] ~
29     OR [ exists %georefel% -cover ] &then
30 &do
31     &type None of these coverages must exist : ~
32     %newbox%, %georef%, and %georefel%.
33     &type At least one does.
34     &return
35 &end
36
37 /* &setvar clipcover [getcover]
38 /* &setvar outlayer [response 'Enter layer to use ']
39
40 create %newbox% %amgbox%
41 &watch %unref%.transf.wat
42 transform %unref% %newbox%
43 &watch &off
44 identity %amgbox% %newbox% %georef% poly 0.1
45
46 /* set the attributes correctly for polygons inside the new
47 /* box boundary but outside the scanned and corrected
48 /* coverage.
49 tables
50 select %georef%.pat
51 resel %amgbox%# <> 1 and %newbox%# = 1

```

```

52 /* This re-selection will report the number of polygons
53 list
54 calc AGNUM = 0
55 calc AGCLASS = 0
56 move '0' to AGTEXT
57 /* 'name' is the AMG box's name in its original coverage.
58 dropitem %georef%.pat %amgbox%# %amgbox%-id name %newbox%# ~
59   %newbox%-id
60 q
61
62 /* Eliminate these 'unlabelled' border polygons.
63 eliminate %georef% %georefel% nokeepedge poly # border
64 resel agnum = 0 and area < 1000.0
65 ~
66 n
67 n

```

#### A.9.4 Back-transformation

backtrans.aml

```

1 /* These two coverages must already exist...
2
3 /* One of the coverages with the correct tics in scanner
4 /* coordinates (inches).
5 &setvar initialcov = bmbv01
6
7 /* The attributed coverage in AMG metric coordinates.
8 &setvar georef = bmbvg07
9
10 /* This AML macro creates this coverage; it must not
11 /* exist on entry.
12 &setvar checkcover = bmbu01
13
14 /* Check coverage existence/non-existence...
15
16 &if ( ^ [ exists %initialcov% -cover ] ) OR ~
17   ( ^ [ exists %georef% -cover ] ) &then
18 &do
19   &type These coverages must both exist : ~
20   %initialcov% and %georef%.
21   &type At least one does not.
22   &return
23 &end
24
25 &if [ exists %checkcover% -cover ] &then
26   &return This coverage, %checkcover%, already exists. ~
27   It should not.
28
29 create %checkcover% %initialcov%
30 &watch %georef%.transf.wat

```

```
31 transform %georef% %checkcover%  
32 &watch &off
```

## A.10 Utility scripts specifically written for CANRI project

### A.10.1 lga-tics.sh

This script produces AML code to generate a tic coverage with tics corresponding to the corners of all 1:25,000 and 1:50,000 (but not 1:100,000) map tiles that are in or overlap a shire. Directly or indirectly, it uses many of the scripts discussed in the next section. The usage is:

```
lga-tics.sh <lga-name>
```

For example, `lga-tics.sh bath` generates the following AML code:

```
generate ticbathll
tics
1,149.250000,-33.500000
2,149.250000,-33.375000
3,149.500000,-33.625000
4,149.500000,-33.500000
5,149.500000,-33.375000
6,149.500000,-33.250000
7,149.750000,-33.625000
8,149.750000,-33.500000
9,149.750000,-33.375000
10,149.750000,-33.250000
END
q
```

Generally, I would capture this to a file (e.g., `maketics.aml`) using `lga-tics.sh <lga-name> > maketics.aml`. You can abbreviate the name, as you see, and you can give it in lower-, upper- or mixed case. You should give only one name as any after the first on the command line are ignored. The script automatically generates the tic coverage name by adding the prefix `tic` and the suffix `ll` (to denote that the coordinates are in lat/longs) to the LGA name you entered on the command line (`bath` in this example). To ensure the tic coverage name is no longer than 13 characters (an ArcInfo limitation), the LGA name you enter on the command line is truncated to 8 characters.

The code:

```
1  #!/bin/sh
2
3  indir="/d8/scripts/"
4  tmpfil="walwalwal.tmp"
5  if [ -f $tmpfil ]; then
6    rm -f $tmpfil
7  fi
8
9  ${indir}lga-map25coords.sh $1 > $tmpfil
10 ${indir}lga-map50coords.sh $1 >> $tmpfil
11
12 # Select only lines beginning with numbers, sort them,
13 # remove duplicates, and prefix with a unique ID number
```

```

14 # when printing.
15 egrep "^[^A-Z]" $tmpfil | \
16 sort -t, -k 1,1n -k 2,2n | \
17 uniq | \
18 nawk -v LGA=$1 '
19 BEGIN{ print "generate tic" substr(LGA,1,8) "ll"
20         print "tics" }
21       { print ++i ", " $0 }
22 END{   print "END"; print "q" }'
23
24 rm -f $tmpfil

```

## A.10.2 Imposing a common set of tics on LGA map box coverages with `make-add-tics.sh`

The script `make-add-tics.sh` generates AML code to copy the tics of one coverage to a series of other coverages (with all the coverages in the same directory). The tic coverage must have a name of the form `tic*ll`, and there must be just one sub-directory with this form of name. Every other coverages whose name ends in `ll` will have code generated to replace its tics. A coverage directory is identified by having an `arc.adf` file in it. Usage is just to run the script in the directory where all the coverages are:

```
make-add-tics.sh
```

Here is an extract of a file captured from this command:

```

generate bombala25xll
copytics ticbombalall
q
generate bukalon25xll
copytics ticbombalall
q
generate candelo25xll
copytics ticbombalall
q

```

The code:

```

1  #!/bin/sh
2  ticcov=`ls -d tic*ll`
3  if [ ! -d $ticcov ]; then
4    echo "There is no tic cover in this directory"
5    exit
6  fi
7
8  for i in `ls -d *ll`
9  do
10   if [ -d $i -a -f $i/arc.adf -a $i != $ticcov ]; then
11     echo "generate $i
12 copytics $ticcov
13 q"
14   fi
15 done

```

## A.11 Existing scripts adapted for CANRI project

### A.11.1 Projecting map box coverages to AMG Zone

#### A.11.1.1 project-xll projects normal map boxes

```
1  &args zone
2
3  &if [NULL %zone%] &then
4  &do
5      &type You must give the AMG Zone (54, 55, or 56)
6      &return Usage: &run project-xll <zone-number>
7  &end
8
9  &if [NULL %zone%] OR %zone% <> 54 AND %zone% <> 55 AND ~
10     %zone% <> 56 &then
11  &do
12     &type You must give the AMG Zone (54, 55, or 56)
13     &return Usage: &run project-xll <zone-number>
14  &end
15
16 /* In the workspace in which it is run, project all coverages
17 /* whose names end in %tail% from lat/longs to AMG Zone %zone%
18 /* with names ending in %newtail%, and build these coverages.
19
20 &setvar tail = xll
21 &setvar newtail = x%zone%
22 &setvar outaml = do-projectx.aml
23
24 &if [ exist %outaml% -file ] &then
25     &sys rm -f %outaml%
26
27 /* generate temporary file name of form xxx*.tmp
28 &setvar outfile = ~
29     [pathname xxx[after [date -ftag] .].tmp -file]
30
31 &setvar count = [ filelist *%tail% %outfile% -cover -all ]
32
33 &if %count% <= 0 &then
34     &return Error generating output file.
35
36 &setvar unit = [ open %outfile% ok -read ]
37
38 &do i = 1 &to %count%
39     &setvar incov = [ read %unit% ok ]
40     &setvar outcov = [before %incov% %tail%]%newtail%
41     &type To project %incov% to %outcov%
42     &sys echo project cover %incov% %outcov% ~
43     /d8/macros/lltoamg%zone%.aml >> %outaml%
44     &sys echo build %outcov% poly >> %outaml%
45     &sys echo build %outcov% line >> %outaml%
46 /* kill %thiscov% all
```

```

47 &end
48
49 &sys rm -f %outfile%
50
51 &type AML script, %outaml%, created to project coverages to ~
52 AMG Zone %zone%.
53 &type Edit to correct AMG Zone numbers, if necessary.

```

### A.11.1.2 project-yll projects big map boxes

The macro project-yll differs from project-xll in only three lines, where different values are assigned to three variables:

```

&setvar tail = yll
&setvar newtail = y%zone%
&setvar outaml = do-projecty.aml

```

## A.11.2 Killing coverages

### A.11.2.1 kill-head.aml

This AML macro kills all coverages whose names start with the string given as the macro's argument. For example,

```
&run kill-head bomb
```

will kill all coverages whose names start with the string bomb.

```

1  &args head
2  /* Put this test in; otherwise, accidentally running just
3  /* "&run kill-head" would kill all the coverages in the
4  /* workspace, an undesirable effect!
5  &if [null %head%] &then
6    &return Usage: kill-head <end-of-name>
7
8  &type Killing all coverages whose names start with %head% in ~
9  this workspace ...
10
11 /* In the workspace in which it is run, kills all coverages
12 /* whose names start with %head%
13
14 /* generate temporary file name off form xxx*.tmp
15 &setvar outfile = ~
16     [pathname xxx[after [date -ftag] ].tmp -file]
17
18 &setvar count = [ filelist %head%* %outfile% -cover -all ]
19
20 &if %count% <= 0 &then
21 &do
22   &sys rm -f %outfile%
23   &return Error generating output file.
24 &end
25

```

```

26 &setvar unit = [ open %outfile% ok -read ]
27
28 &do i = 1 &to %count%
29     &setvar thiscov = [ read %unit% ok ]
30     kill %thiscov% all
31 &end
32
33 &sys rm -f %outfile%

```

### A.11.2.2 kill-tail.aml

This AML macro kills all coverages whose names end in the string given as the macro's argument. For example,

```
&run kill-tail x55
```

will kill all coverages whose names end in the string x55.

```

1  &args tail
2  /* Put this test in; otherwise, accidentally running just
3  /* "&run kill-tail" would kill all the coverages in the
4  /* workspace, an undesirable effect!
5  &if [null %tail%] &then
6     &return Usage: kill-tail <end-of-name>
7
8  &type Killing all coverages whose names end in %tail% in ~
9  this workspace ...
10
11 /* In the workspace in which it is run, kills all coverages
12 /* whose names end in %tail%
13
14 /* generate temporary file name off form xxx*.tmp
15 &setvar outfile = ~
16     [pathname xxx[after [date -ftag] .].tmp -file]
17
18 &setvar count = [ filelist *%tail% %outfile% -cover -all ]
19
20 &if %count% <= 0 &then
21     &return Error generating output file.
22
23 &setvar unit = [ open %outfile% ok -read ]
24
25 &do i = 1 &to %count%
26     &setvar thiscov = [ read %unit% ok ]
27     kill %thiscov% all
28 &end
29
30 &sys rm -f %outfile%

```

### A.11.2.3 kill-xll.aml and kill-yll.aml

These two AML macros are specialized forms of kill-tail.aml in which the tail of the name is xll or yll, respectively. The two macros take no argument.