AUSTRALIAN OS9 NEWSLETTER

EDITOR :
Gordon Bentzen
8 Odin Street
SUNNYBANK Qld 4109

(07) 345-5141

NOVEMBER 1989

ATTR Syntax: Attr filename [permissions] Usage : Examine or change the security permissions of a file Opts: -perm = turn off specified permission perm= turn on specified permission -a = inhibi rms : d - directory file s - nc to owner w - write permit to own r - read permit to public pw - e permit to public BACKUP Syntax ge : Copies all data from one de ead error occurs s = single rites BASIC09 Syntax : Basic0 ge BUILD Syntax: Build filenar s from standard input CHD S nge working directory to specifi > Usage: Change execution directory to specified path CMP Syntax: Cmp filename1 filename2 Usage : File comparison utility COBBLER Syntax: Cobbler devname Usage : Creates OS-9 bootstrap file from current boot CONFIG Syntax: n disks COPY Syntax data from one fil E Syntax : Date [t Opts: t = specify ame> Usage : Check directory for wor sters -m = save of unused cluster nly -o = print <devname> }<devn : Del [-x] filenam s : -x = delete x: Deldir directo yntax: Dir [e] [x] the file names y x=print executi Usage : Display s converted characters to standard output DSAVE Syntax : Dsave [-opts] [dev] [pathname] Usage : Generates procedure file to copy all files in a directory system Opts : -b make a system disk by using OS9boot if present -b=<path> = make system disk using path do not process b makdir command o num K ECHO Syn tandard output ED oriented text edito s text error messages for given error numbers EX Syntax: ex <modname> Usage: Chain to the given module FORMAT Syntax : Format <devname> Usage : Initializes an OS-9 diskette Opts ; R - Ready L - Logical format only "disk name" 1/2 number of sides 'No of

AUSTRALIAN OS9 NEWSLETTER
Newsletter of the National OS9 User Group

EDITOR : Gordon Bentzen

HELPERS : Bob Devries and Don Berrie

SUPPORT : Brisbane OS9 Level 2 User Group.

November, 1989

So, here we are at November already. Another year has almost gone, and Christmas is just around the corner. Everyone is making their Christmas wish list. The same goes for us died-in-the-wool OS9 users. But where do we go to get our goodies? Now that InterTan Aust has left us in the lurch, where can we go?

The answer to that is really quite plain. We will go to the places which have patiently provided support all these years without necessarily making a big name for themselves. These places include Blaxland Computers, Computer Hut Software, Paris Radio, and, of course, there are a large number of software and hardware sellers in the USA and Canada who can be relied upon to provide value for money and good service.

Of course, we will need to be careful about buying from overseas companies, and I can only say that I, personally, will only buy from companies which get the 'Rainbow Seal' from 'the Rainbow' magazine. I strongly recommend that you read their reviews and write-ups before committing yourself. Now don't get me wrong, this is not an advertisement for 'The Rainbow', but should be read as a mini review of their services. Advertisements for Australian companies may be found in CoCo-Link magazine.

In this issue of our newsletter you will see another section of Ole Eskildsen's discussion of fourth generation languages and 'Sculptor', a review of the CRC Super controller, and a Basic09 programme by Don Berrie for those of you who have purchased 'Window Writer'.

I would like to take this opportunity to welcome a number of new members in our midst, people who have heard of us from the CoCo-Link magazine (thanks to Robbie Dalzell), from John Belshaw at Blaxland Computers (thanks John), or simply by word-of-mouth. Welcome !

Some of you people out there may not realise it, but we try to make this a user supported newsletter. 'Sure, I paid my $18.00' I hear you say. Yes, you did, and I thank you for that, without that commitment we would not be here at all, but do you realise how hard it is to fill ten to a dozen pages each month with something that is of interest to every-one? I would like to get some ideas of what to write about. Would some-one volunteer to do an article or two on how to use 'Dynacalc'? How about 'PhantomGraph'? We would even consider printing the results in the newsletter of a graph made from a Dynacalc file.

I've been assured by Bob Devries that if anyone has an idea for a small utility type programme (nothing too fancy), he is willing to 'give it a bash' writing in 'C'(preferably) or Basic09. How about it, any takers?

Are there any people reading this who have any knowledge about other OS9 systems besides Colour Computer level one and two, and are willing to share their knowledge or give me a pointer to users of such systems, I would be most grateful. I'd like to think that this newsletter is not just for Colour Computer users, but can service the wider OS9 community as well.

Like every-one else, we would like a holiday too, so we intend publishing a bumper December-January issue. Because it will be larger than normal, we will be unable to get it compiled before the beginning of December, but we hope to be able to mail it during the second week of the month.

Well, that's it from me for this month, keep the OS9 flag flying, and happy computing,

Regards,
The Editor.

The following submission is part two of the series by Ole Eskildsen on fourth generation languages. The content of this series serves as an example of just what can be done with OS9 (even with the humble Color Computer) and another reason why we all need to maintain interest in this versatile Operating System. Please let us have your comments so that we can endeavour to cater to your needs. Remember, no feedback can be very difficult to deal with.

Editor.

FOURTH GENERATION LANGUAGES (4GLs) AND OS-9

PART 2 -SCULPTOR

This is the second part of my three part article on 4GLs for the OS-9 operating system. How did you find Part 1? I hope you liked it. In this part I will describe one of the 4GLs available under OS-9, namely SCULPTOR. Sculptor was developed in the UK by Microprocessor Developments Ltd and has been available for some time for OS-9. Under OS-9 it is a true multi-user system with efficient (and automatic) record locking. The programs are compiled which results in faster running programs. Besides OS-9, Sculptor is available for a great many other operating systems including MS-DOS (single-user or networked multi-user), UNIX and XENIX, DEC VAX, QNX and many, many others. Not only that, but it is object code compatible, which means that a program compiled on ANY version will execute on ANY OTHER version WITHOUT recompilation. Can you imagine: You develop a program or a complete system in Sculptor for instance on a CoCo3 (Sculptor is only available for OS-9 Level 2) and then copy it to a UNIX machine or MS-DOS or whatever and it is instantly ready to run. Of course to achieve this result it is necessary to have a run time version of Sculptor on the target machine because Sculptor compiles its programs into an intermediate object code (I-code) much like BASIC09.

So what do you get when you purchase a copy of Sculptor Development System? You get a good looking A5 size manual in a 4-ring binder in a slip case in the by now familiar PC tradition containing a couple of hundred pages. There are actually three manuals: 1) Installation Manual; 2) An Introduction to Sculptor; and 3) Sculptor Reference Manual. You further get two diskettes with the system and utilities, and the Australian Distributor has added a third diskette with a collection of useful little programs and numerous samples. I found the Sculptor Introduction very easy to go through and after spending only a short time (a coulple of hours) I was able to start experimenting with the language, and what a delight this turned out to be.

But first, what modules does the Sculptor Development System consist of? It consists of 1) A Data Dictionary. 2) Compilers for interactive Screen (also called Forms) type programs and Report type programs. These are called CF and CR respectively. 3) The run time modules to execute the programs compiled with CF and CR, called Sage (for screen type programs) and Sagerep (for report type programs). 4) Two programs which will automatically generate complete screen or report type programs from your data dictionary specifications. These are called SG for screen generator and RG for report generator. 5) A Menu program to automatically generate menus. 6) Various utility programs to perform file handling tasks, and to specify different screen and printer types, and finally a Query program to perform ad-hoc queries and reports. In other words, a complete development and run time environment with facilities to run on many different computers, terminals and printers without reprogramming.

We will now take a closer look at how all this will work for you. The best way, I think, would be by describing an example, so I will now describe the typical work sequence in a simplified form.

1. Systems Design - You define what the system is going to accomplish.
2. Data Definition - What data do you need to store.
3. Development - Develop the programs, compile and test.
4. Production - Use the program(s).

To use a simple and well worn example. Let us say we want to develop an address book. What data would we need? Probably Name, Street, Town, State, Postcode (zip code for our American friends), telephone number, etc, etc. So we decide to create a file called address, which we do by using the dictionary program called "Describe", we simply type: 'describe address'. We now see:

```
Descriptors for address

For each field enter:
name,heading,type&size,format;validation
Type h for help.

KEY FIELDS
  1:
```

We enter the information about each field, selecting one or more fields to hold the key which must be unique. When we are finished, the record description might look like this:

```
Descriptors for address

KEY FIELDS
  1:name,NAME,a20,
DATA FIELDS
  2:street,STREET,a20,
  3:town,TOWN,a10,
  4:state,STATE,a3,
  5:postcode,PCODE,a6,
  6:phone,TELEPHONE,a10,
```

We save the record description which creates a file called "address.d" and we can now generate a standard Screen Form program by typing:

```
sg address
```

and in less than one minute we have a program where the source program is called "address.f" and the compiled program is called "address.g". Before we can execute the program we have to create the actual data file and its associated key file called "address" and "address.k" respectively. This is done by typing "newkf address" and in seconds the empty data file is created. The Screen Form program we created presents the following screen when we type: "sage address", where "sage" is the Screen Form program run-time interpreter:

```
/-------------------------------------------------------------------------\
|                        "ADDRESS"FILE MAINTENANCE                        |
|                                              Today's date [        ]    |
|                                                                         |
|                      NAME [                    ]                        |
|                                                                         |
|                         STREET [                    ]                   |
|                           TOWN [          ]                             |
|                          STATE [    ]                                   |
|                          PCODE [       ]                                |
|                      TELEPHONE [          ]                             |
|                                                                         |
|   i=insert  f=find  n=next  p=prev  m=match  a=amend  d=delete  e=exit  |
|                                                                         |
|                     Which option do you require?                        |
\-------------------------------------------------------------------------/
```

Neat, don't you agree? The second last line shows all the options, such as inserting records, finding records, amending records, and so on. All without any programming. Now go ahead and enter some records, find them again, amend them, delete them, etc.

Now that we have got some data in our file, let us create and print a report. We type: "rg address" and a moment later we have a source program called "address.r" and a compiled object program called "address.q".

To run the report we type "sagerep address" where "sagerep" is the run-time interpreter, but you had figured that out, hadn't you?

This is what the report would look like:

```
------------------------------------------------------------------------------

                        "ADDRESS" FILE REPORT

Page:    1  Date: 27/ 9/89

NAME                STREET              TOWN       STATE PCODE  TELEPHONE

Bentzen, Gordon     8 Odin Street       Sunnybank  QLD   4109   07-3455141
Berrie, Don         25 Irwin Terrace    Oxley      QLD   4075   07-3753236
Devries, Bob        21 Virgo Street     Inala      QLD   4077   07-3727816
Eskildsen, Ole      11 Monarch Street   Kingston   QLD   4114   07-2094322

                           END OF REPORT
------------------------------------------------------------------------------
```

How do you like that? In less than an hour we have created a complete little system to create, store and manipulate our address book and report on it!

Are you begining to see the power of 4GLs? I hope so, otherwise I have failed in describing the benefits to you. Remember, the source programs generated can be modified by the programmer, so what I do most of the time, is to generate the standard program first and then use that to develop the final product. In the extremely simple example above we have only used one file, but you can, of course, use multiple files. You, the programmer, have complete control over everything going on, if you desire. On the other hand, it only takes minutes to generate simple applications. Another point which shows the power of Sculptor: When the Screen Form program is being compiled, the compiler examines the way files are used in the program. If a file is being updated in any way (such as in the example above) then Sculptor creates the necessary record locking, so that two persons cannot update the same record at the same time by accident. Ask the DBase (IBM PC database) programmers if they can do that automatically, or whether they have to meticulously write the record locking every time. If a file is not used for updating then no record locking is being performed. Pretty powerful stuff!

Who is using Sculptor, you might ask? Well, quite a number of individuals and organizations. For instance, Australian Telecom, a Queensland Government Department, numerous software houses, and many others.

The above is only a very small, simple example, space does not permit any more. In Part 3 I will discuss a more complex example and make you an offer you cannot refuse, I hope. Address any correspondence concerning these articles to:

Ole Eskildsen
11 Monarch St
Kingston QLD 4114
Phone: (07) 209 4322

A TIDY-UP PROGRAMME FOR WINDOW WRITER*

(*) - Window Writer is a Copyrighted programme designed by Raj Dash, and published by OWL-WARE.

As promised in the last issue of the newsletter, here is my solution to return the system to the way it was before the Window Writer programme is run. See my article on Page 10 of the October issue for a description of the problem. Simply put, however, the programme Window Writer modifies your system to the extreme, and the only effective way to restore it is to reboot.

This programme was written in Basic09 so that the majority of users will be able to use the code. It could have been written in "C" and been considerably shorter (and easier), but that would have limited its usefulness. For those of you who don't have the Window Writer programme, the Basic09 code still forms a useful tutorial on some more-advanced Basic09 programming techniques.

The solution of the Window Writer problem is two-pronged. Firstly, modifications to the palette tables and window parameters can be solved by running the programme in a newly created window. When the window is finally closed, all of the above modifications die with the window. Because I wanted the fix to be system independant, I have used the /w descriptor to open the new window. This descriptor uses the next available window descriptor for the creation of the new window. The problem is however, that we don't know the name of the new window (it could be W1, W2, W9 etc or any other unused window descriptor). We have to know this in order to redirect the input and output (NOTE #1 in PROCEDURE wt) of the WW programme to this new window.

Because we know the path number that was returned when we opened the window (in the OPEN #wpath statement), we can use this value to read the path descriptor options section of the path descriptor, and get the actual window name (returned in the string variable winnam). This is accomplished in the PROCEDURE getdev.

So far so good. But now comes the difficult part.

The other part of the problem, as I explained in the last newsletter, was that a number of modules remained in memory. The cause of this problem is twofold. Firstly, some modules have a non-zero link count. Modules which have a link count are retained in memory. Secondly, modules which are merged together and loaded as a single file of less than 8K bytes (and are therefore in the same 8K block in memory), will not be unlinked until the link count of every module in that block becomes zero. Therefore some modules will be retained in memory even though their link counts are zero.

A compounding factor with this programme is that the link counts of the modules which are retained in memory (for whichever reason), change! As far as I can tell, the specific modules which remain linked, and their actual link counts, is dependant on exactly which parts of the programme are used.

My first approach involved simply using the OS9 UNLINK command via the Basic09 SHELL function, and continuing to unlink each module until all the link counts became zero. Unfortunately, while UNLINK does not report an error when it tries to unlink a module with a link count of zero, it does report an error when trying to unlink a module which is no longer there. This means that if you try to unlink a module which has been removed, a error is generated. Because of this, it is not possible to use an error trap (for error #221) to determine when all the link counts are zero. (Because it will generate that error as soon as the first 8K byte block is removed, and there is more than one block involved in this case).

After running the programme in a number of different ways, and using a number of different features, I determined that the two modules which always had the highest link counts were "Bchain" and "termset". Provided that I could find a way to ignore any ERROR #221's which were generated, I surmised that, provided I unlinked all the modules until those two modules were no longer linked, the all would be fine.

The method that I used involved using the system call UNLOAD via the Syscall procedure, (see page 8-42 of the OS9 Technical Referance Manual) and only checking the B register after each attempt to unlink all of the modules. Hey presto it worked! Also, it had the added advantage that the programme no longer requires that the UNLINK command be available, and is thus more system independant.

As yet I have been unable to develop a method to circumvent the merging of Window Writers custom modules with the

Runb module. The only thing that I can suggest is that you maintain a separate commands directory for this programme which has the modified Runb module located there.

To use this patch programme, simply type in the code, save it, then pack the code. Transfer the packed code to the execution directory from which you normally run Window Writer. Instead fo typing ww to start the programme, type wt (for Window Tidyup), and away you go.

If you have any problems, please contact me. I personally don't use Window Writer, as I do not own a copy of it. However, I have tried all sorts of combinations of commands while running Window Writer via this patch programme, and so far have not found any problems. Myself? I use Stylograph, and love it!!

Happy writing. Cheers ... Don Berrie (07) 375-3236.

<Listing>

```
PROCEDURE wt
DIM wpath:BYTE
DIM winnam:STRING[32]
DIM i:INTEGER
OPEN #wpath,"/w":UPDATE
SHELL "load termset"
RUN gfx2(wpath,"DWSET",2,0,0,80,24,4,1,1)
RUN gfx2(wpath,"SELECT")
RUN getdev(wpath,winnam)
SHELL "ww <>>"+winnam (*****    NOTE  #1    *****)
SHELL "unlink termset"
RUN unload
100 RUN gfx2(1,"SELECT")
RUN gfx2(wpath,"DWEND")
CLOSE #wpath


PROCEDURE getdev
TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
DIM regs:registers
PARAM wpath1:BYTE
PARAM winnam:STRING[32]
DIM i:INTEGER
DIM callcode:BYTE
regs.a=wpath1
regs.b=$0E
regs.x=ADDR(winnam)
callcode=$8D
RUN syscall(callcode,regs)
FOR i=1 TO 32
 EXITIF MID$(winnam,i,1)>CHR$(128) THEN
  winnam="/"+LEFT$(winnam,i-1)+CHR$(ASC(MID$(winnam,i,1))-128)
 ENDEXIT
NEXT i
END

PROCEDURE unload
TYPE registers=cc,a,b,dp:BYTE; x,y,u:INTEGER
DIM regs:registers
DIM i:INTEGER
DIM callcode:BYTE
DIM modul$(37):STRING[20]
DIM modtype(37):INTEGER
```

```
FOR i=1 TO 37
 READ modtype(i)
 READ moduls(i)
NEXT i
DATA 34,"get_name"
DATA 34,"get_menu"
DATA 34,"fcopy"
DATA 34,"frename"
DATA 33,"gfx2"
DATA 33,"SysCall"
DATA 33,"Inkey"
DATA 33,"wp_trulen"
DATA 33,"wp_putline"
DATA 33,"wp_sublen"
DATA 34,"cjustify"
DATA 34,"rjustify"
DATA 33,"wp_getline"
DATA 33,"empty_keys"
DATA 33,"space_count"
DATA 33,"cut_lead"
DATA 33,"get_ptr"
DATA 17,"PBaud"
DATA 33,"process"
DATA 33,"wp_insert"
DATA 33,"wp_delete"
DATA 33,"ctrl_search"
DATA 33,"space_search"
DATA 33,"unjustify"
DATA 34,"get_select"
DATA 33,"get_mouse"
DATA 34,"froll"
DATA 34,"fget"
DATA 33,"indent_count"
DATA 33,"dumb_prt"
DATA 34,"fjustify"
DATA 34,"fsort"
DATA 33,"wait"
DATA 33,"output"
DATA 33,"fork"
DATA 33,"BChain"
DATA 33,"termset"
callcode=$1D
WHILE 0<>1 DO
 FOR i=1 TO 37
  regs.a=modtype(i)
  regs.b=0
  regs.x=ADDR(moduls(i))
  RUN syscall(callcode,regs)
 NEXT i
 IF regs.b<>0 THEN
  GOTO 100
 ENDIF
ENDWHILE
100 SHELL "unlink runb"
END
```

<End of Listing>

Super Controller
a hardware review
by Bob Devries

Package: Super Controller II
Source : CRC Computers
11 Des Laurentides Blvd.
Laval. (Quebec)
CANADA. H7G 2S3
Price : US$130.00

The CRC Super Controller II is by far the best piece of Colour Computer hardware I own besides the CoCo 3 itself. It gives me the best of both worlds. It works fine as a standard controller under RSDOS (yech) and, with the special software drivers provided, it simply flies with OS9 Level 2.

Imagine this. You want to copy all the files from one disk to another using DSAVE, but, at the same time, you must get that article for the newsletter finished. Under normal conditions, you would put off the DSAVE until later, because the newsletter has higher priority. Well, now you don't have to put it off anymore. You can run your favourite word-processor AND DSAVE at the same time! (I'm doing that right now as I type this review. The only way you'll notice that DSAVE is running is from the whirring of your drives, and the flashing of the drive activity LEDs. The drive accessing simply does not interfere with other processes running any more than normal without the drive access! Sounds great? Here's what you get:

The SCII comes in a neat aluminium box of about the same dimensions as the older TANDY controller, painted beige with the CRC Disto logo on it. Inside, there is a neatly solder masked, silk-screened printed circuit board with 21 integrated circuits on it. These include a socketed EPROM with your choice of RSDOS1.1 or CDOS, (or you can put in your own 24 pin Tandy ROM), a WD1773 disk controller IC, a 6116 2K static RAM chip, and the rest are support chips. Also on the board is a 17 pin expansion plug. The edge connectors for the computer and drive connexions are gold plated and the computer end includes 2 ground connexion tabs.

The 17 pin plug is used to attach the expansion boards which are also available from CRC. These are:

Real Time Clock and Parallel Printer adaptor
Mini EPROM programmer
RS-232 adaptor
Hard disk adaptor (SCSI bus)
Hard disk adaptor with RS-232
3 in 1 multi-board adaptor
this includes: parallel printer port
real time clock
RS-232 port.
4 in 1 multi-board adaptor
this includes all of the 3 in 1 features plus:
Hard disk adaptor (SCSI bus)

I have had my SCII for some time now, and I am really thrilled by its performance. I have had no incompatibility problems, and all my software, both OS9 and RSDOS runs perfectly with it. On a scale of 1-10 I'd rate this package an 11 !

Regards,
Bob Devries

Here is a listing of the current membership, just in case you may wish to contact other OS9ers near you. We have not included phone numbers, as it may be that some numbers given to us are the unlisted type etc. So we suggest initial contact by the mailing addresses shown.

To members who have renewed for this subscripton year, our sincere thanks for your continued support, and to the new members we say WELCOME to the National OS9 user group. We hope that you all find something of interest in the pages of each edition.

NATIONAL OS9 USER GROUP MEMBERS as at 10/28/89 mm/dd/yy

AMBROSI	Jules	172 OGILVIE STREET	ESSENDON	VIC 3040
BENTZEN	Gordon	8 ODIN STREET	SUNNYBANK	QLD 4109
BERRIE	Don	25 IRWIN TERRACE	OXLEY	QLD 4075
BLANDFORD	George	27 CANBERRA STREET	MOE	VIC 3825
CHASE	Scott	3 THOMAS STREET	BAXTER	VIC 3911
COSSAR	Lin	12 RAKEIORA GROVE	PETONE	N.Z 6303
DALZELL	Robbie	31 NEDLAND CRES.	PT.NOARLUNGA STH	SA 5167
DEAN	James	P.O. BOX 549	Sth. WINDSOR	NSW 2756
DEANE	Robert	3 BLOOM COURT	CRANBROOK	QLD 4814
DEVRIES	Bob	21 VIRGO STREET	INALA	QLD 4077
DONGES	Geoff	2 BONNOR CLOSE	HOLT	ACT 2615
EATON	David	20 GREGSON PLACE	CURTIN.	ACT 2605
EDWARDS	Peter	40 DAVISON STREET	MITCHAM.	VIC 3132
ESKILDSEN	Ole	11 MONARCH STREET	KINGSTON	QLD 4114
EVANS	John	80 OSBURN DRIVE	MACGREGOR	ACT 2615
FRITZ	Terry	M/S 546	FOREST HILL	QLD 4342
FROST	Phil. A.	25 CHEETHAM STREET	KALGOORLIE.	WA 6430
HARRIS	Michael	P.O. BOX 25	BELMORE	NSW 2192
HUGHES	Peter	54 PRINCETON ST	KENMORE	QLD 4069
JACQUET	J.P.	27 HAMPTON STREET	DURACK	QLD 4077
JAMES	Phil	P.O. BOX 859	LIVERPOOL	NSW 2170
JENKINSON	Cec	49 HUTHWAITE ST.	WAGGA WAGGA	NSW 2650
KEWLEY	Douglas	2 DRYSDALE AVENUE	TEE TREE GULLY	SA 5091
MACKAY	Rob	27 MAWARRA STREET	KINGSTON	QLD 4114
MANNING	Peter	7 CALNON STREET	BASSENDEAN.	WA 6054
McKAY	Ross	56A CORNELIA ROAD	TOONGABBIE	NSW 2146
McLINTOCK	George	7 LOGAN STREET	NARRABUNDAH.	ACT 2604
McMASTER	Brad	119 WILLOUGHBY ROAD	CROWS NEST	NSW 2065
MORTON	David	c/o P.O.BOX 195	CONDOBOLIN	NSW 2877
MUNRO	Ron	45 SUNNYSIDE CRES.	NORTH RICHMOND	NSW 2754
O'DONNELL	Bill	47/2 FRANCIS STREET	ARTARMON.	NSW 2064
OBLAK	Gerd	58 ELIZABETH PARADE	LANE COVE	NSW 2066
PATRICK	Wayne	12 O'CONNELL ST.	GYMPIE	QLD 4570
PEARCE	W.Leigh	47 ALLENBY AVENUE	RESERVOIR.	VIC 3073
PRATT	Ross	31 CAMPBELL STREET	COOMA	NSW 2630
REID	Theo	35 AVONMORE AVE.	PORTLAND	VIC 3305
SCHIPPLOCK	Kevin	19 CELTIS STREET	ACACIA RIDGE	QLD 4110
SIMPSON	Andrew	17 GLENEFER STREET	RUNCORN	QLD 4113
SINGER	Maurice	217 PRESTON ROAD	WYNNUM WEST	QLD 4178
SKEBE	Jeff	92 BYNYA ROAD	PALM BEACH	NSW 2108
UNSWORTH	Rob	20 SALISBURY ROAD	IPSWICH	QLD 4305
USHER	John	47 POLARIS AVE.	KINGSTON	QLD 4114
WAGNITZ	Ken	2 DEPINDO AVE.	EDEN HILLS	SA 5050

Total Members 43