

CBM Amiga Version

AMOS

TOME

SERIES IV



SHADOW SOFTWARE

TOME is © Shadow Software 1991  
AMOS is © European Software 1990  
TOME Requires AMOS 1.3 or higher,  
and at least 1 meg of memory

## Foreword

1

Thank you for buying this latest version of the TOME system. It is because of honest users of the software that we have been able to keep TOME updated, and are able to release this version with a full manual (my apologies for the manual being on disk for Version 3, but Shadow Software is only a little outfit, and we couldn't afford to release TOME 3 with a full manual).

It is because of the requests phoned and written in by current TOME users that I have had added so many commands to TOME IV that it has nearly twice as many commands as the previous version ! I have also beefed up the editor somewhat, there is still nothing to rival it, on any computer !

Despite attempts by rival companies to hide TOME from programmers, it has managed to get itself used by ALL the major AMOS programmers, and again, this is mostly due to users "spreading the word", thank you !

I hope you like this new version, and my first attempt at a manual of this size. I will of course be doing updates in the future, and adding to the manual. Keep in touch via the AMOS Club so that you can get hold of these updates, and if you have any ideas for new commands, or ways of improving the current ones, don't hesitate to call.

One last point, please don't pirate this product. It affects me, and all the people who work for, or with me directly. I can't afford to stay in the software business if people copy my work. If you see a pirated version of TOME anywhere, do something nasty to the disk or BBS that it is on.

Oh, and to my competitors, keep on trying to catch up boys !

Aaron Eathergil

This Manual is for all the TOME Version 3 users who asked "Where's the bloody manual ?",  
it is also dedicated to the following people:

Mike Oldfield

Piers Anthony and Terry Pratchett

Adam Fothergill and the Testing Team (Say hi to everyone in Newcastle !)  
Friends old and new.

# TOME Series IV

## Map Programming System For the Amiga Instruction Manual

Welcome to AMOS TOME Series IV !

TOME IV is the most powerful map editing system on any computer anywhere (TOME Series 3 is the second most powerful).

TOME is used to create huge game maps, which you can then use in your AMOS programs to give you huge background areas without using up much memory. TOME works with TILES, which are a fixed size, which are put together in a MAP. Each tile can be used as many times as you want in a map, and only 1 byte is used for every repeat of a tile. So you can get a 960 screen map in only 64K !

About 4 years of work has now gone into the TOME project since it was first seriously developed on the Atari ST. Thanks to users not pirating the software, we have been able to continuously upgrade TOME, with new routines that we have thought of, and ones that users requested. We have also been able to provide upgrades for users of older versions. As a TOME User, the only thing we ask of you is that you don't copy the software or manuals for anyone else. You should make a backup for your own use, but that's all !

---

### Installation

---

AMOS TOME Series IV is very easy to install on your AMOS System.

- 1) Make sure you are using a Back up of your AMOS disks and your TOME disk.
- 2) Load and Run the program TOME4Install.AMOS from your TOME Series IV disk.
- 3) Sit back and wait !
- 4) Once the program has finished, reboot AMOS, and TOME Series IV will be installed in your AMOS System, along with the latest version of the AMOS Club/Shuffle Extension (V2.6)

Once you have the TOME extension installed, you will have about 60 new commands that you can now use (These are detailed later in this manual). All the programs on the TOME Series IV disk use these commands, so make sure that they are installed before you try and run any of the TOME Series IV programs !

If you experience problems with the installation, check through this troubleshooting chart:

Problem	Cause	Action
Syntax Errors when entering a TOME command	Extension Not installed	Run the TOME4Install.Amos Program Check that TOME.Lib is in your AMOS_System Directory Check that the Default Configuration has :AMOS_System/TOME.Lib listed as extension 7.
Extension Not Present command when trying TOME demos or editor	Extension Not installed	As Above.
Extension Not Present command when attempting to compile a TOME program	Compiler hasn't been told that TOME is here !	Check that the RAMOS1_3.Env configuration has :AMOS_System/TOME.Lib listed as extension 7
AMOS Won't start up	Extension incorrectly installed	Make a new copy of AMOS Update it to at least V1.3 Run the installer again, and make sure that all Disk activity is finished before restarting AMOS.

Apart from the Demonstration programs, the only main program you will be using is called TOME.AMOS. This is the actual Map Editor, which you load into AMOS and Run. Basically, It's Big ! It has all the functions you will need to create and edit Maps for use in your AMOS or even machine code games.

**Copyright etc.** AMOS TOME is © Shadow Software 1991 and must not be duplicated in any form. Authorised users of TOME (i.e. those of you who bought it) can make a backup for your own use, but neither the original disk or the backup, or any of the files on the TOME disk or any part of the documentation may be lent, hired or duplicated in any way. When using TOME in any sort of program that is to be released either as P.D (including Shareware and related schemes) or Commercially (including Licenseware, program rental and other similar schemes) none of the TOME files may be included on the disk. Thus a self running TOME program must be compiled. You must not use TOME with RAMOS. This is to avoid piracy of the extension.

Other than that you can use TOME in any way you want in your programs, as long as you aren't in breach of our copyright we are quite happy. If you get your program into the best sellers lists give us a mention !

---

### Terminology

---

This being a technical manual, there are a few technical terms used. The following should help...

**TILES** are small fragments of a picture that can be used to make up a picture. All the tiles used in a map are the same size as each other, so that all the map needs to know, is what tile is to be placed in each position.

**MAPS** are large background screens made up from Tiles. A Map uses TILE CO-ORDINATES, so that each position on the map represents one tile.

TOME also has **BRIKS** (It is supposed to be spelt like that) which are groups of tiles. Briks can be pasted onto the screen like icons, or pasted into the map. This lets you change huge areas of your map instantly.

**TILE NUMBERS** are the actual number of the tile. i.e. the first one in the bank is Tile Number 0, the second one is number 1 and so on. In TOME Series IV you can have up to 255 tiles (An upgrade is being worked on which will allow you to use up to 511 tiles).

**TILE VALUES** are numbers that can be assigned to each tile. In TOME IV you can assign up to 8 values to each tile in 8 Tile Value Lists (Each list contains 1 value for each tile). These values can represent anything you want, but here are a few examples:-

**Whether or not the tile is solid**, 0-127 can represent non-solid, while 128-255 shows that the tile is solid.

**What tile it should change to if blown up**, the value can be set to the tile number of another tile, so that



when you hit the tile, it is easy to find out what tile to change it into.

**How many points you should get for hitting the tile,** when you hit the tile, multiply this tile value by 10 say, and add it to the score.

**Whether the tile has any special effects or not,** This can be combined with the Solid/Non-Solid value. e.g:

Non-Solid Tile with no effects	=	0
Non-Solid Tile with Ice on it	=	1
Non-Solid Tile with Bomb on it	=	2
Non-Solid Bonus Tile	=	3
Solid Tile with no effects	=	128
Solid Tile with Ice on it	=	129
Solid Tile with Bomb on it	=	130
Solid Bonus Tile	=	131

With values like this you can see if the tile is solid or not (Value>127) and which special effects the tile has (Value Mod 128). These values are examples, as you can assign any value (0-255) to each tile.

Tile Values are one of TOME's more powerful features, and although you may not need to use them straight away, you will probably find all sorts of uses for them !

---

### First Steps 1, "What is a map ?"

---

Put simply, a map is a grid of tiles, which is normally used to display huge background areas in games. If, for instance, you wanted a game that scrolled horizontally over 20 screens, you could do it the difficult way, and somehow draw 20 screens and cram them into memory which would take up 640K (each screen takes up 32K in 16 colours). O.k, so you could compact them, but even assuming that you can get each screen down to 6K (Very reasonable compaction), you are still looking at 120K, and you still have to decompact the screens (Very slow)! Now imagine designing the background as a grid. Each point on the grid can use one of 256 tiles selected from a set you have designed. Each point on the grid only takes up 1 byte of memory, and the only overhead you have is the size of the icon bank (not usually more than one screen).

You can design all your screens from these tiles, rather like a collage or jigsaw puzzle (depending on how you draw your tiles !). The end result. Even assuming you use 32K for your tiles, the 20 screen map can be done in 4.8K, so you would only be using **36.8K for the 20 screens !** (Even less if you are using the larger 32x32 pixel tiles). Now you can see why about 90% of games use map based backgrounds, and why AMOS TOME was

written.

The TOME Editor allows you to design these maps, and the TOME Extension commands you have recently added to your AMOS system give you commands specifically written to display and communicate with these maps in your program.

The only limits to the size of a TOME map are memory, and the maximum size of 32768x32768 tiles, which is about 4,473,924 screens using 16x16 pixel tiles, and just under 18 million screens using 32x32 pixel tiles. Unfortunately, this sort of map would take up 1 gigabyte (1024 Megabytes) of memory, but you could always work to CD Rom !

---

### First Steps 2, Loading & Displaying a Map

---

There are 5 types of file used for TOME, only 2 of which are vital, the other three are just useful.

We start with the **Tile Bank**. As far as AMOS is concerned, this is an Icon bank, where all the icons are the same size (e.g 16x16 or 32x32). This is loaded in exactly the same as a normal icon bank, i.e

Load "filename.abk"

which will load it into bank 2

You might want to change the size of tile that you want to use. Simply use the command:

Tile Size x,y

to change the size. X and Y can be 16 or 32, giving you 4 different sizes of tile, listed below with their pros & cons:

Size	Advantages	Disadvantages
16x16	Uses less memory for the tiles, single tile changes are rapid, <u>Best Collision detection. Best for general usage.</u>	Slightly slower scrolling
16x32	Reasonable memory use. Fast for scrolling vertically, <u>Reasonable collision detection</u>	Slow at scrolling horizontally
32x16	Reasonable memory use, Fast for scrolling horizontally, <u>Reasonable collision detection</u>	Slow at scrolling vertically
32x32	Fastest scrolling	High memory usage, Inaccurate collision detection



## 8

The Collision detection between the player and tiles can be dealt with in many ways to make it more accurate, we'll cover this later in the manual though.

You can find out how many tiles you have in the bank with the command:

```
Print Length(2)
```

You can also see what tiles you have available by using the command:

```
List Tile
```

Which will display them all on the screen.

I think that's enough theory for now, let's load in a tile bank and put some of what you've learned into practise. Go to direct mode and type:

```
Load "TOM24:Tiles/MazeManTiles.Abk"
```

Then type:

```
Curs off : Flash Off : Get Icon Palette : Cls 0
```

This will get the colour palette from the tiles and clear the screen (after getting rid of that @\$%&\$ text cursor).

Now we'll set the tile size:

```
Tile Size 16,16
```

and have a look at the tiles:

```
List Tile
```

You should now have a screen full of the tiles used in the Maze Man demo game !



The Maze Man Tiles

Now you have managed to load in the tiles, you can load in the map to go with them. Maps can be stored on disk in two different ways, either as an AMOS bank (with .abk on the end of the file name), or more normally as a .MAP file.

The easiest one to use is the AMOS bank format with the .Abk extender. It wastes a little memory on disk, and you need to remember what type of file it is, but it is very easy to load into AMOS. Try the following...

```
Load "Mazeman_map.abk",6
```

You have now loaded in the map. The ,6 at the end of the load instruction, tells AMOS to load it into bank 6, which is where the maps normally go.

If you want to use the .MAP type of file, loading is still quite simple (sort of!). Basically, you have to reserve bank 6 to the correct length and then use the BLOAD command to load in the map. Fortunately, TOME gives you a command to tell you the length of a map, the -Map Length(x,y). The X & Y parameters are the width and height of the map in tiles. So for the Mazeman map, which is 100x100 tiles, you would do:

```
L=Map Length(100,100)
```

```
Reserve as Data 6,L
```

```
Bload "Mazeman.Map",Start(6)
```

This achieves the same as the previous load command, but with this file type you save a little disk space and it is easier to spot that it is a map file. Note the use of ,Start(6) at the end of the Bload. This is because the Bload command requires the memory address to load to rather than a bank number.

With the map in memory, we can play around with it!

The **Map View** command tells TOME what area of the screen you want to use to show the map. We'll use a large area, say from 0,0 (The Top left) to 320,192 (nearly the bottom right of an NTSC screen). So we would use the command:

```
Map View 0,0 To 320,192
```

Which is logical enough! (Map\_View\_Demo.Amcs on the TOME disk lets you play around with the Map View command).

We can now display the map, using the **Map Do x,y** command. This command displays an area of the map in the area of the screen that you defined with Map View. The x,y co-ordinates are the point on the map (measured in tiles) that you want TOME to start drawing from. For instance, enter:

```
Map Do 0,0
```

and TOME will display the Top left corner of the map. Entering:

```
Map Do 1,0
```

will redisplay the map one tile in to the map:

```
Map Do 1,1
```

will move you down another tile, and so on. It is possible to use this to do a simple (if incredibly slow by TOME's standards) form of scrolling. See the Example on the TOME disk (Tomedemo2.Amcs).

Being able to scroll around a map is quite handy, but is not much use in a game if you can't check to see what the player is standing on, bumping into, about to be blown up by, and so on !

TOME has a comprehensive set of functions to allow you to check tiles in the map, so that you can find out what the player is standing on, where the nearest ultra splatto blitzem bonus is, or whatever.

The easiest one to understand is called `=Map Tile(x,y)` and returns the tile number at co-ordinate x,y on the map (x & y are of course measured in tiles).

Try the following..

```
Print Map Tile(0,0)
```

```
Print Map Tile(2,3)
```

```
Print Map Tile(105,94)
```

Oops ! the last one returned an error ! This is because the Map Tile function only allows you to check within the map itself. As this map is only 100x100 tiles in size, trying to check the tile at 105,94 is impossible, so TOME told us so !

"So hang on a minute !", you are now thinking to yourself, "how am I supposed to know what size the map is ?".

Good question ! Fortunately, there is a good answer. There are two functions in TOME called `=Map X` and `=Map Y`, these return the size of the map in tiles, logically, `=Map X` returns the width, and `=Map Y` the height. So if your game is going to load in maps of different sizes, you can use the Max and Min functions to make sure that your map co-ordinates don't go off the map !

```
TX=Max(0,Min(Map X-1,X))
```

```
TY=Max(0,Min(Map Y-1,Y))
```

```
Print Map Tile(TX,TY)
```

now whatever values you put into X & Y you can't force an error !

Alternatively, the `=Range` function serves the same purpose as the Max & Min commands. This function is supplied as part of the Shuffle extension, and is used like this:

```
TX=Range(X,0 To Map X-1)
```

Basically, it forces the value of X into the range of 0 to Map X-1 ( If it is less than 0 it makes it 0, if it is greater than Map X-1 it makes it Map X-1). This function can be used with any set of parameters, not just map co-ordinates.

Another, more powerful, function used to check tiles on the map is called `=Tile Val`, and uses the syntax

```
=Tile Val(x,y,list)
```

and if you've read the bit about tile values earlier in this manual, you'll know that they are most excellent, and totally bedacious when it comes to finding out what you are standing on.

`Tile Val` works in the same way as the `=Map Tile` function, in that it first gets the tile number at the co-ordinate X,Y on the map. It then looks up that particular tile number in the requested list and returns the tile value (a number between 0 and 255) that is listed there for it.

Before using the `tile val` function, you must make sure that the tile values have been loaded in. Like the maps, they can be stored on disk in two formats, either as `.Abk` which is easier to load, or as `.Map.Val`, which saves disk space. If it is stored on disk as a `.Abk` file, simply do:

```
Load "MyTileVals.Abk",8 : Rem Bank 8 is the Tile val bank
```

If it is a `.Map.Val` file, find out the length (between 256 and 2048 bytes in size, it will always be a multiple of 256 bytes). You now need to reserve bank 8 to this length in bytes, and `Blload` the file to the start of bank 8. e.g

```
Reserve As data 8,256
```

```
Blload "MazeMan.Map.Val",Start(8)
```

You can now use the `=Tile Val` function.

You can also change the map from within your program, using the `Map Plot` command, which works the same way as the normal `Plot` command, except that it changes a tile on the map instead of a pixel on the screen. The command uses the syntax:

```
Map Plot tile,x,y
```

and only changes the map itself, it doesn't change the screen until you re-display the map in some form. (See `Map Update`). Try the following:

```
Map Do 0,0
```

```
Map Plot 5,3,3
```

and you'll notice no change in the screen, but the map itself has changed. If you do

```
Print Map Tile(3,3)
```

you will see that tile number 5 is now at co-ordinate 3,3 in the map. If you now re-display the map with another `Map Do 0,0` command, you will see that the tile has changed.

## Scrolling

Scrolling is a bit of an art form, in that there is never a perfect way of scrolling a screen, merely a lot of different ways, each with its own advantages and disadvantages. I could write forever about the trade offs and differences between software scrolling (using screen copy) and hardware scrolling (using screen offset), but I won't! From the point of view of the beginner (and because of the memory saving, from the advanced user also), software scrolling is easier to use, and to understand. It is also more memory efficient, which is the whole point of TOME!

We can achieve scrolling by copying one section of the screen to another. For instance, if you wanted to scroll the screen left by a pixel, you would copy everything from 1 pixel in from the left, to the left of the screen, thus shifting it left.

Let's try an example:

```
Screen open 0,320,200,16,Lowres : Curs Off : Flash Off : CIs 0
```

```
Plot 319,100,1
```

```
Do
```

```
  Rem scroll 1 pixel left
```

```
  Screen Copy 0,1,0,320,200 To 0,0,0
```

```
Loop
```

Run this, and you will see the dot moving slowly to the left, leaving a trail behind it!

This trail is the "New" part of the scrolling area. Normally, we would replace this edge with some graphics, usually from a map. This is where the edge scrolling commands come in handy. Take a look at the *Edge\_Scrolling\_Demo* Amos program for an example of this.

## Using Map Handle

The Map Handle command is very powerful, and is specifically written for use in games which will scroll in 8 directions (i.e. all over the place).

To use Map Handle, you have to arrange your screens very carefully. In an ideal situation, you should have a Double Buffered Screen (say screen 0) and a single buffered, but hidden screen (say screen 1).

You then set your Map View command so that the second X and Y co-ordinates give you 1 tiles width and height more than the area you are going to see. So if you wanted to display from 0,0 to 256,160 and were using 16x16 pixel tiles, you would have to set map view as 0,0 To 272,176. You will also have to make sure that the hidden, single buffered screen can handle this area.



Before starting your main loop, use the command **Map Handle Init**, which resets the various internal registers used by the **Map Handle Command**.

At the start of your main loop, you will need to take your current map co-ordinates (in pixels), and turn each into two values, the number of tiles and the remainder in pixels. TOME has 4 commands to help you with this, **=Map Hx(px)** and **=Map Hy(py)** which return the number of tiles, and **=Map Fx(px)** and **=Map Fy(py)** which return the remainder.

The number of tiles is passed to the **Map Handle** command, along with the screen to display on. e.g (MX & MY are the maps pixel co-ordinates)

```

HX=Map Hx(MX)
HY=Map Hy(MY)
FX=Map Fx(MX)
FY=Map Fy(MY)
Screen 1
Map Handle 1,HX,HY

```

You would then switch to screen 0 (your double buffered screen), and display any bobs (remember to turn **Bob Update Off** before the main loop), and you then do the screen copy, which copies the hidden version of the map to the double buffered screen, handling the pixel offset as it does it:

```
Screen Copy 1,FX,FY,Width+FX,Height+FY To Logic(0),0,0
```

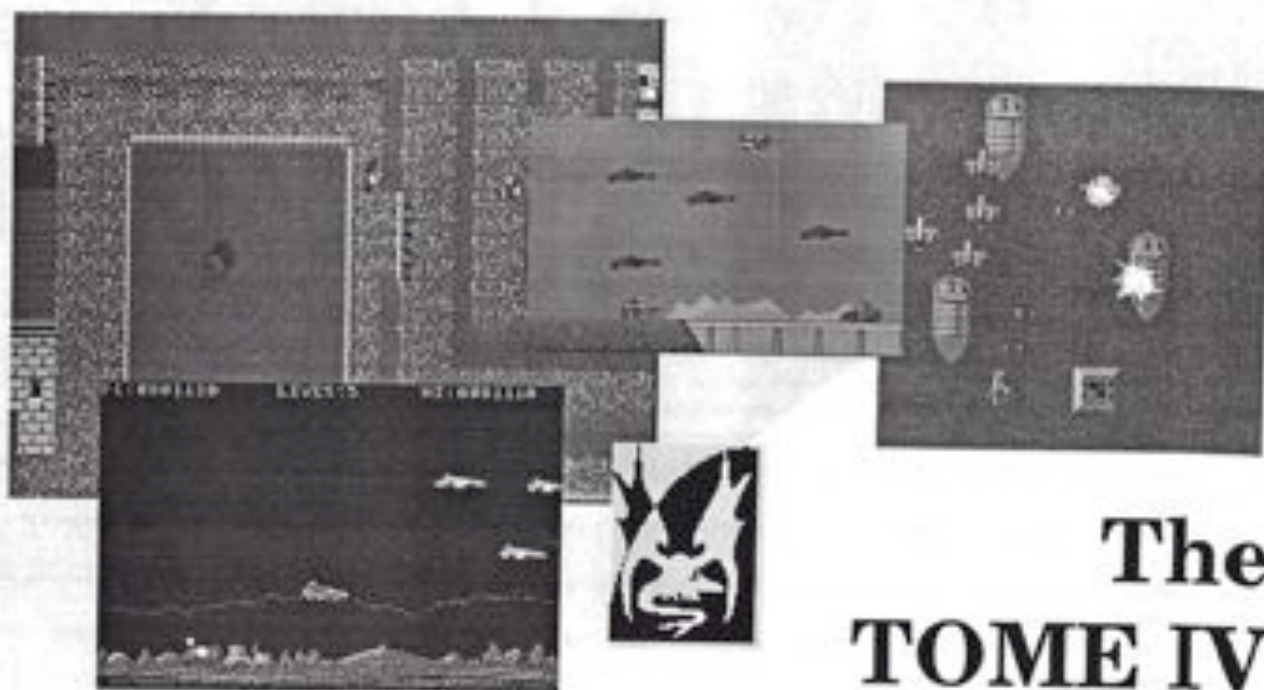
Width and Height in the above command are the Width and Height of the area you want to see.

To finish off, you then do a **Bob Draw** command (To update the bobs) and a **Screen Swap** (To flip the screen to the front so that you can see it).

To move around this map, you simply have to add and subtract from the MX and MY variables. Any time you want to jump to a particular position in the map, do a **Map Handle Init** first.

For an example of using **Map Handle** to scroll, see the **Map\_Handle\_Demo.Amos** program on the TOME disk.





# **The TOME IV Map Editor**

# The TOME Series IV

## Map Editor

The TOME Map Editor is the heart of the TOME System. It is used to design the maps you will use in your programs, create the tile banks required, and even create tile animations, tile value lists and bricks for use in your programs.

To run the Map Editor, you will need at least 1 megabyte of memory (it is one of the largest AMOS programs ever written). If you are running on a heavily overloaded A500 (You know, 4 disk drives, hard drive and lots of accessories running), you are going to be close to the limits on memory, and the editor will automatically economise to try and run on your machine. If it does run out of memory, switch off and disconnect all but 1 disk drive, and make sure that no accessories are running at the same time as AMOS, that or get more memory. This shouldn't be a problem on most machines, only on 1 meg machines that are really overloaded!

Loading the Map Editor is very simple. First make sure that you are using AMOS 1.3 or above, and that you have installed the TOME and Shuffle Extensions. If you haven't, refer to the installation instructions at the start of this manual.

Now select LOAD from the AMOS file menu, and select the program TOME.Amos from your TOME Series IV disk.

After a while, the editor will have loaded in. You can now press F1, or select RUN from the menu, and the editor will kick itself into life!

If you are one of those people who is pushing a poor defenceless A500 to its limits of memory, the editor will automatically save 94K of memory by:

- 1) Shutting down workbench
- 2) Switching to NTSC mode (You can switch back to PAL from the Niceness menu if you think you have the memory).
- 3) Drop the packed icon screens out of memory, so that they are accessed from disk instead.
- 4) Use 2 colour icon screens instead of 8 colour. Not quite so ergonomic, but memory saving.

You don't lose any functionality if TOME does this, it just means that when using the Tile Maker and MaPLe utilities there will be a very short delay while TOME loads in the icon screens from disk.

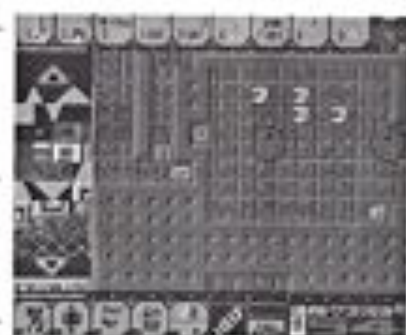
Once TOME is running, you should have a screen looking similar to **Diag 1**. TOME works with a very simple system of Icons, similar to other programs on the Amiga. You have two rows of icons on the screen, the Menu Selector Icons (at the top of the screen) and the Command Icons (at the bottom of the screen).

Clicking on one of the nine Menu Selector Icons will select one of the nine menus of Command Icons, which will appear in the bottom row.

Menu Selector Icons, These choose which Command Menu you want to use

The Tile Selector Palette, this is where the tiles you will be using to create the map are selected from.

Command Menu Icons, these activate the various TOME commands



Scroller Controls, you can use these, a joystick or the cursor keys to move around the map.

The Map Display window, the map you are working on is displayed here.

A Window Dragger, you can drag the windows up and down with this handle.

Snooze Corner, clicking here will put TOME to sleep while you multi-task another program.

The Help ! Button, Activate Help ! and TOME will tell you what you are pointing at !

**Diag 1. The TOME Editor**


Clicking on one of the 5 Command Icons displayed will activate that command. On some of the Command menus, the Command Icons will be split (Like the icons in the Disk Menu for instance). These Icons have 2 or even 4 separate functions on them. The 2 button icons are split horizontally, so that clicking on the top part will activate one command, while clicking on the bottom part will activate another. Other icons (such as on the Pine Drawing menu) are split into 4. Clicking on each quarter will activate a different command.



If at any time you are unsure of an icon or button on the screen, you can look for it in this manual, or click on the HELP button (to the right of the command icons at the bottom of the screen) or press the HELP key on your keyboard.

By activating the Help ! system, any icon or button you point to on either menu screen will be explained by TOME in a window in the centre of the screen. You can still use all the functions of TOME while it is doing this ! To de-activate the Help ! system, click on Help again, and select Cancel from the alert box.

#### General Controls

On each menu bar, there is a grabber handle button, which looks like this: : Dragging the window by this handle will let you move the menu bars to anywhere on the screen vertically, so that you can see what is on the map window under them. If you want to see the whole map window, simply press the Space Bar, or click on the FULL SCREEN Button on the bottom menu (next to the Help Button). This will flip the Map Window to the front, allowing you to work on all of it. You can also use the Left & Right Amiga keys to scroll the window left and right, thus allowing it to use the area taken up by the Tile Selector Palette. This can also be achieved by holding down the Left Alt key, and dragging the Map Window with the mouse.

Most TOME Functions are accessed by clicking with the left mouse button. The Right mouse button is allocated to a "useful" function for each command menu. For instance, clicking the right mouse button when in the normal Map Editing menu will take you to the Tile Selector Screen, so that you can quickly pick a tile from your entire list. Here are the Right Button "Useful" functions for each menu.

Menu	Function
1) General Options	Jump to Map Edit Menu 1
2) Disk Options	Select File types to use.
3) Utilities	Jump to Map Edit Menu 1
4) Map Edit 1	Select Tile
5) Map Edit 2	Select Tile
6) Brick Menu	Select Brick
7) Bank Manager	Fine Edit Current Tile
8) Fine Editor	Pick Colour
9) Locator control	Skip to next stored location

You will probably have noticed the numbers on each of the menu selector icons, and the letters Q,W,E,R and T on

the Command Icons. Basically, these are a keyboard shortcut you can use. Pressing a key from 1-9 will select the associated Menu, and pressing the Q,W,E,R & T keys activates a command from that menu. e.g 1Q selects the Niceness control from the General Options Menu.

There are also keyboard shortcuts for most of the functions in TOME, these are listed in the Editor Keyboard Shortcuts on page 32.

---

### Getting Around

---

To move around the map area, you have a pretty good choice of options. You can use the scroller control in the right side of the main menu bar. Clicking on any of the directional arrows on this control will move the map by one tile (as you edit the map in tiles, it scrolls around a tile at a time). If you have a joystick plugged into your Amiga, then you can also scroll around using it. By pressing the Fire button, you can skip 10 tiles at a time. If the cursor keys are in "Move" mode (The default) then you can use them to scroll around also (Shift activates the skip 10 tiles function).

---

### Locator Marks

---

If you want to skip back to a particular point on the map, then you can store its location in one of the 10 Locator Marks. Holding down Shift and pressing one of the 10 function keys will memorise the current position on that key. Pressing the same function key (without Shift) will bring the map back to that point instantly. These 10 locator marks can also be played around with in the Locator Menu.

When you start up TOME will load the locators with 10 locations across the map going from top left to bottom right in a diagonal line.

---

### Drawing

---

Drawing a map couldn't be more simple! Clicking on any tile in the tile selector on the left of the screen will select that tile, and clicking anywhere in the map area will place the currently selected tile onto the map at that position. As the map always works in terms of tiles, the tile will automatically snap to the nearest tile position (rather like pegging it to a board).

If you want to see more of the map area, you can press the Space Bar (or the Full Screen button), which will hide the menus. You can also use the A and J keys, which will scroll the palette off the left, giving you more of the map area in the screen. Pressing the Left Alt key and dragging the map area with the mouse gives you the same control.

In TOME you won't always be just placing single tiles. The editor has several (about 30) different modes that it can use, ranging from simple placement of the current tile (called Draw mode), through to complex evaluation of

surrounding tiles for wall placement (Maze drawing mode). Normally you will be using one of the simple modes such as draw, box, cut and paste, however at any time you are not sure what mode you are in, you can either click on the Draw button in the 1st Edit Menu, Press "4Q" or simply press "D", all of which will put you back into draw mode.

There are two other modes that you can use, called "Brik Draw" and "Random Draw". These are highly useful, and work along with the normal modes (So you can be using Box mode using Brik Draw for instance). Both are simple to explain (hopefully) and easy to use.

**Brik Draw** allows you to draw using a previously defined Brik (Cut out using the Brik functions) as a sort of fill pattern. This means that if you have a particular complex pattern of tiles for your dungeon floors, or you want your spaceship walls to be made up from a complex pattern of computer banks, then you simply design the pattern, cut it out as a Brik and switch to Brik Draw mode (The last option in the Brik Menu or "6T"). Now anything you draw will automatically be drawn using the Brik as a pattern.

**Random Draw** is basically the same, except that it will use a random tile from the current user defined tile palette. So if you want your floor to be a random mix of concrete, mouldy bits and cracked bits, you select the User tile palette (By clicking on the Paet Palette button at the bottom of the tile selector, so that it changes to User Palette) and selecting the Grab to User option ("4T") which allows you to select a tile from on the map and click it into the user tile palette.

When you have selected your tiles for the random spread you require, click on the Random Draw mode button (The top half of the far right one on Menu 5). You simply select how large a random spread you want to use and Ok it, now when you draw on the map, the tiles used will be picked randomly from the range you specified. Both Random Draw and Brik Draw can be used with the following modes: Draw, Box, Circle, Flood Fill, Clear Map, Scatter and Map resize.

Now that you know how to get around your map, and plot tiles to it, you might want to find out what each of the Menu options does ! Remember that you can switch the Help ! mode on at any point, to tell you what each icon represents.



## Menu 1, General Options (Right Button goes to Edit Menu 1)

This menu contains generally useful functions, such as the Niceness Controls, Map resizing and Grid controls.



### Option 1: Niceness Control

This menu gives you complete control over the look and feel of the TOME Editor, and has several options to make your map designing life a little easier. Not everyone will want to use TOME in exactly the same way as we do, so there are options in the niceness control for:

**PAL or NTSC operation:** (TOME automatically starts up using whichever screen mode your computer is using anyway, but this option lets PAL users test their maps in NTSC mode).

**Cursor Chase mode:** Normally the co-ordinate display in the bottom screen shows the tile co-ordinates of the top left of the viewed map area. However, if you switch Chase co-ords on, it will show the tile co-ordinates of the tile that the mouse cursor is pointing at. Highly useful if you need to put a tile at a particular co-ordinate.

**Keyboard Control Mode:** The Arrow keys on your keyboard can be set to scroll around the map (Move, the default), or to move the cursor around the map area itself. In this case the Shift key acts like the left mouse button, so that you can move around and plot tiles using only the keyboard.

**Icon and Tile colours:** Not only can you change the RGB settings of the colours used in your tiles, but you can also change the colours TOME uses for its icons. Thus if you don't like the look used (One of our testers said it reminded him of the Alien film sets) you can completely customise it !

**Load & Save User Tile Palette:** The tile palette to the left of the screen normally shows the Preset palette, or rather all the tiles, in the order they appear in the tile bank. The User palette contains tiles that you place into it, in any order you like, thus allowing you to have your most used tiles at hand without scrolling through the palette. The load and save functions allow you to save this palette list to disk.

**File Formats:** Each of the file types saved out by the TOME editor has its own format. However, you might want to save the file out in the more simple AMOS memory bank format (.Abk), so that you can use the straight



**Load command in AMOS.** This option allows you to select which way the files will be saved and loaded.

**Save Niceness Settings:** Once you are happy with your re-designed editor you can save the settings to disk. These will then be loaded automatically when you restart TOME.

**Reset to Defaults:** Just in case you aren't very good at ergonomics, or you want to have a laugh at what it looked like before you customised it, this will restore all the settings to their default.



### Option 2 : Grid Options

Sometimes you might need to see the boundaries between the tiles, either on the map area itself or on the tile selector screens. The following options allow you to put overlaying grids on the map display and the selector screens, and even to put the tile numbers on the tile selector screen.

**Map Overlay Grid:** This switches the overlay grid on for the map display area, so that you can see exactly where you are placing tiles.

**Tile Screen Overlay Grid:** This activates a similar grid for the selector screen and tile selector to the left of the map. This is so that you can see which tile you are picking up.

**Tile Numbers:** This displays the numbers of the tiles when you use the tile selector screen.

**Flip Screen Grid:** If you are doing a flip screen game rather than a scrolling one, then you might want to save all your screens as a large map, and simply jump around the map to each screen. By activating the Flip Screen grid, you can mark out your map into areas the size of your screen (The co-ordinates for width and height are in tiles).



### Option 3 : Map Size

Put very simply, this option allows you to change the size of the map in tiles. You have a choice of initialising the map to the current tile/fill pattern. If you leave this initialise option off then the current map will be resized.



### Option 4 : Credits

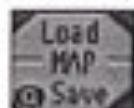
Probably the most important bit of the program this ! The programming was by Aaron Fothergill (myself), the graphics were all done by Adam Fothergill, and it was tested by amongst others, (deep breath) Len Tucker, Dion Peeke, Peter Hickman and Sandra Sharkey. Thanks also to Norm Allen in Australia and Dave Lazerek in America.





## Menu 2, Disk Options (Right Button Selects File Types)

This menu is where you'll do all your loading and saving of TOME files. Note that all the buttons in this menu are split into 2 parts. The Top part of the button LOADs the file and the bottom part SAVEs. Just in case you are about to accidentally save over an existing file, the Save option will check with you first!



### Option 1 : Load/Save Map

This button loads and saves your TOME maps in either .Map or .Abk format. When loading up your map and tiles at the start of a session, always load the tiles first, as TOME Checks the map when it is loaded to make sure that it doesn't use any tiles that aren't available.



### Option 2 : Load/Save Tiles

This button loads and saves Tile banks. These are normal AMOS Icon banks, which can be created in the Tile Maker built into the TOME editor, or Sprite X. Tiles are always loaded and saved in .Abk format.



### Option 3 : Load/Save Briks

This button loads and saves TOME Briks, which can either be in .Brk or .Abk format. The brk bank must be one that is created with the currently loaded tiles, or you will get unpredictable results.



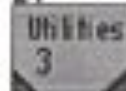
### Option 4 : Load/Save Tile Values

This button loads and saves Tile value lists. These are normally saved and loaded along with the map when you use the Map Load/Save function, but this gives you the option to edit a tile value list without affecting the map.



### Option 5 : Load/Save Tile Anims

This button loads and saves TOME Tile animation banks, either in .Anm or .Abk format. These are edited within the Anim Editor (see the Utilities Menu). When loaded all anims are switched on.



## Menu 3, Utilities (Right button goes to Edit Menu 1)

This menu contains 5 utilities for use on your map. These include the Tile Maker, Anim Controller and Tile Valuer as well as the MaPLe language. As they are quite complex each has its own section later in this manual.



### Option 1 : Picture to MapConverter

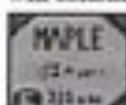
This function allows you to design your map from a zoomed out (1 pixel per tile) perspective in an art package, and then convert it into a map using the current set of tiles. See Page 43 for details.

This is not a function for creating tiles from an IFF screen, you use the Tile Maker function for that.



### Option 2 : Animation Controller and Tile Swap controller

As with the file buttons, this one is split into two. The Anim controller (top half) allows you to define tile animations for use in your map. The Tile Swap controller allows you to define pairs of tiles which will constantly swap with each other. See page 39 for full details of these 2 utilities.



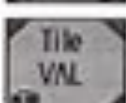
### Option 3 : MaPLe Editor

MaPLe (Map Programing Language) is a custom written language specifically for working with TOME maps. It allows you to do powerful processing jobs on the map. Full details are in the section on MaPLe on page 41.



### Option 4 : The Tile Maker

The tile maker is used to create tiles for use in TOME by cutting them out from IFF Screens. A full description of the functions of the Tile Maker are on page 36.



### Option 5 : The Tile Valuer

As mentioned earlier in the manual in the section on programming with TOME, tile values are incredibly useful. This utility allows you to assign up to 8 values to each tile. For full details of what the Tile Valuer's functions are see page 45 and for information on what Tile Values are and how to use them see page 9.



## Menu 4, Edit Menu 1 (Right Button Selects Tile)

This menu contains 4 basic drawing modes, and the Grab to User palette function. As this menu is one you will be using a lot, several of the other menus switch to this one on hitting the right mouse button.



### Option 1 : Draw Mode ( Selectable with the "D" key)

Draw mode is the default mode in TOME. In draw mode, you simply place single tiles onto the map. Draw mode supports both Brick Fill and Random Tile modes.



### Option 2 : Bar Mode (Selectable with the "S" key)

In Bar mode, you select the Top left and Bottom Right of a box, which is then drawn using tiles. As with Draw mode, both Brick Fill & Random Tile are supported.



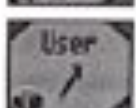
### Option 3 : Disc Mode (Selectable with the "C" key)

Disc mode works exactly the same way as Bar mode, except that it draws a disc with the tiles.



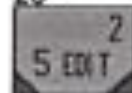
### Option 4 : Flood Fill Mode (Selectable with the "F" key)

When you click on the map with Flood Fill Mode, the map area visible on the screen will be flood filled with the current Tile/ Brick fill pattern.



### Option 5 : Grab tile to User Palette

Normally in TOME, you will use the preset palette, where all the tiles are shown in the order that they appear in the tile bank. However, you may want to use a particular group of tiles, which are spread out across the bank. This is where the User Palette comes in. Click on the PSet Palette button below the tile selector and the user palette will appear. Selecting Grab Tile to User Palette now allows you to select files from within the map area or the Tile selector screen (by pressing Right mouse button) and click them into a slot in the user tile palette. Once you select a drawing mode, you can use the UTP in exactly the same way as the normal tile selector.



## Menu 5, Edit Menu 2 (Right Mouse Button selects tile)

This menu contains some of the more powerful drawing functions in TOME.



### Option 1 : Maze Draw Mode (Selectable with the "M" key)

This mode is normally used for drawing plan view maps of games involving walls and floors, such as Gauntlet and Puffy's Quest, where the walls are made up of several tiles, including corners and junctions. Basically, Maze mode automatically decides which piece of wall should be plotted to make the wall correct at the bit you are drawing. The Maze mode selector screen allows you to decide which tile is to be used for each of the 13 wall sections (2 Straights, 4 Corners, 4 Ends, 4 Junctions and the Crossover). If you are using the Maze Man tiles, the Maze mode settings are ready to go using the wall tiles in that tile bank.



### Option 2 : Hard Scroll Map

There will be more than one time when you will find out that you have drawn the map one tile too far up, down, left or right. The Hard Scroll function allows you to move the tiles on the one map one tile in any direction.



### Option 3 : Clear Map

This function clears the map to the current Tile/ Brick fill pattern. It will of course check to make sure that you are really sure you want to do this.



### Option 4 : Automatic Map Draw

The Auto Map Designer is again designed to work with plan view dungeon type games, although I've no doubt it will be put to other uses (roads and railways for instance). Basically it will draw an entire map for you from scratch ! All you have to do is either set it going with its defaults, or change the various settings, such as the amount of corridors, rooms, the floor tile or pattern to use and the occurrence of the various objects to scatter. Then sit back and watch ! The function will take some time, but it uses Maze Mode to calculate the walls (or roads/tracks), it will then scatter the scatter objects around the resultant maze.





### Option 5 : Random Draw & Scatter

The Random Draw option selects Random Draw Mode and allows you to define the range of tiles from the USER Tile Palette that will be used. Basically, when Random Draw is in operation, one tile is picked at random from the range of tiles specified, from the User Tile Palette, every time a tile is placed. This is very useful when you want to clear a map to say, a grass tile, but you want to have variations of that tile used. To choose the tiles Random Draw will use, select the User Tile Palette (By clicking on the Feet Palette button), select Grab Tile to User Palette (Menu 4) and place the tiles you want to use in the first few positions of the User Palette. Then select Random Draw and set the number of different tiles you want to use. You can then select the drawing mode you want to use, and draw using tiles picked at random from this range.

The Scatter function randomly places the current tile within the currently viewed area of map, allowing you to scatter objects within this area.



### Menu 6, Brik Options (Right Buttons Selects Brik)

This menu gives you TOME's cut and paste options. Any bricks created with these functions can be saved and used in your AMOS Programs, or you can use them with the Brik Fill Mode.



#### Option 1 : Cut Brik from Map (selectable with the "B" key)

This allows you to cut a brik from the current map. Once it is cut (it doesn't change the map) TOME will automatically go into Brik Paste Mode.



#### Option 2 : Cut Brik From Tile Picture

If you have designed a large, multiple tile object when designing the tiles, you can cut it out as a brik with this option. Also, if you press the "B" key when in the Tile Selector Screen, it will allow you to Cut a brik from the picture. As with Cut Brik from Map, TOME will go into Paste Brik Mode.



#### Option 3 : Paste Brik (Selectable with the "P" Key)

This selects paste brik mode, allowing you to paste the current brik onto the map.





#### **Option 4 : Pick Brik** (Also selectable with Right button)

This allows you to select which brik to use. The Brik Selector also allows you to delete briks.



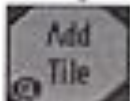
#### **Option 5 : Brik Fill Mode** (also selectable with the "O" key)

When Brik Fill mode is active, any drawing will be done using the current brik as a fill pattern. This is useful when you need to cover an area with a repeating pattern of tiles.



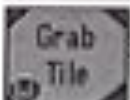
### **Menu 7, Tile Bank Editor** (Right Button Fine Edits Current Tile)

This menu contains functions to arrange and edit the tiles used in the map, including the ability to add, delete and flip the tiles used.



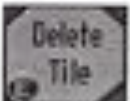
#### **Option 1 : Add Tile to Bank**

This function adds a blank tile to the end of the tile bank. It will not allow you to exceed TOME's limit of 256 tiles however.



#### **Option 2 : Grab Current Tile from Screen**

If you have designed a tile using the fine draw functions (See the next menu), Grab Tile allows you to grab it from the map view area, replacing the current tile.



#### **Option 3 : Delete Tile from Bank**

If you find you don't need a particular tile, and want to save memory and/or make space for another tile, the Delete option allows you to delete a tile from the bank, and remove all occurrences of it on the map. Clicking on Delete Tile and O.K'ing the subsequent alert box will activate Delete mode. Clicking twice on any tile in the tile selector will delete that tile. To get out of delete mode either select any other drawing mode or click on Delete Tile and select Cancel from the Alert box.



#### Option 4 : Flip Tile

This option lets you flip the current tile horizontally and vertically. Remember that this function actually modifies the current tile, so to make flipped versions of a tile, copy them with the Fine Editor (Option 5) which also has Flip functions.



#### Option 5 : Fine Edit Tile

Fine edit zooms the current tile up to many times its normal size and allows you to edit it using various drawing functions (draw, line, box, bar, circle, disc, fill etc), before putting it back into the bank overwriting the old version, or adding it to the end of the bank.



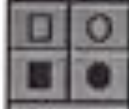
#### Menu 8, Fine Draw Menu (Right Button selects drawing colour)

Fine Draw gives you basic art package features, which can be used to design tiles in rough form, if you happen to forget to put them in originally. None of these functions affect the map in any way.



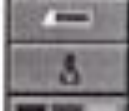
#### Option 1 : Fine Draw, Flood Fill, Line & Airbrush

Pretty self explanatory, these 4 functions allow you to draw, fill and spray with the current colour.



#### Option 2 : Fine Box, Bar, Circle & Disc

4 more drawing options.



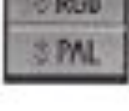
#### Option 3 : Fine Cut & Paste

Not to be confused with Brick Out & Paste, these functions cut and paste graphics on the current screen.



#### Option 4 : Select Colour & Fill Pattern

Allows you to select the colour & Fill pattern you want for drawing.

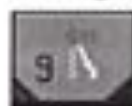


#### Option 5 : Colour Controller and Palette Selector

The Colour Controller is exactly the same as the one in the Niceness options and allows you to edit the RGB Colour values of your tile bank. It has 2 extra functions over the TOME V3 colour control,

in that it has Range and Tint control. By selecting a range of colours with the left (1st colour) and Right (Last colour) mouse buttons and clicking on Range, the colours will be calculated to be a spread of colours going from the 1st to the last. To tint a range of colours, simply select the range with the left and right mouse buttons and select Tint. Clicking on any colour will now tint the range one shade towards that colour. To switch Tint off, simply click on the Tint button again.

The Palette Selector allows you to choose from 10 different palettes for your tiles. This is useful when you want to change the look of the game between levels, without having to load in new tiles.



## Menu 9, Locator Controls (Right Button steps through locations)

The locator controls give you rapid access to any part of the map by letting you jump to any of the 10 stored locations. The locator marks can also be stored and recalled with the Function keys.



### Option 1 : Store Location

Clicking on this button will allow you to store the current map location in any one of the ten locator marks. This can then be recalled with a function key, or the Recall Location button.



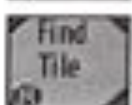
### Option 2 : Recall Location

This option allows you to recall one of the 10 stored map locations.



### Option 3 : Goto Location

By clicking on this button and entering an X,Y tile co-ordinate, you can jump to any location on the map.



### Option 4 : Find Tile

Clicking on this button will search through the map for the currently selected tile.



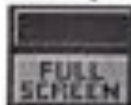
### Options 5 : Find and Replace Tile

This option works like the Find Tile function except that you have to click on a tile in the map first. This tile will then be searched for and replaced by the currently selected Tile/ Brick Fill.



### The Help ! Option

For those of you who are prone to forgetting what the icons do, the Help ! function is at hand. By clicking on Help (or pressing the Help key) and pressing the O.K option you can switch Help ! on (You switch it off by doing the same and selecting Cancel). Once it is active, Help ! will tell you what any button or icon you are pointing to does. You can still work all the buttons and controls when Help ! is active.



### The Co-ordinates display and Full Screen Button

The co-ordinates display shows you what section of the map you are currently looking at, or the co-ordinates of the tile you are pointing at if you are in cursor chase mode. By clicking on the Full Screen button (Or pressing the Space Bar) you can flip the menu bars behind the map view and edit the map using the whole screen. You can return by either hitting the space bar or pressing the "Return to Menu" button above the tile selector.



### The Scroller Controller

This control is used to scroll around the map using the mouse (Assuming you are not using the joystick or keyboard to do this). The button in the centre of the controller gives you a zoomed out overview of the map, which you can save to disk as a screenshot if you wish.

# Keyboard Controls

Many of the TOME Editor's functions can be replicated with keyboard presses:

<u>Key</u>	<u>Function</u>
1-9 Keys (not Keypad)	: Select Menu
Q,W,E,R,T	: Select function from current menu
<	: Grab Tile from screen to nearest tile
>	: Grab Tile from screen
V	: Get tile value if in direct tile val mode, and/or select direct tile value mode
D	: Select Draw mode
S	: Select Bar mode
C	: Select Circle mode
F	: Select Fill mode
U	: Select Grab to User palette mode
B	: Select Cut Brik mode
P	: Select Paste Brik mode
H	: Select Fine Draw mode
J	: Select Fine Box mode
K	: Select Fine Bar mode
L	: Select Fine Circle mode
:	: Select Fine Line mode
[	: Select Fine Cut mode
]	: Select Fine Paste mode
#	: Select Fine Disc mode
Y	: Select Search and Replace mode
O	: Switch Brik Fill mode on/off
X	: Switch Random Draw on/off, when switching on, press a number key next to select the range.



<u>Key</u>	<u>Function</u>
<b>M</b>	: Switch Maze mode on/off
<b>G</b>	: Switch map overlay grid on/off
<b>? or /</b>	: Switch cursor chase on/off
<b>Shift+ F1-F10</b>	: Record current position
<b>F1-F10</b>	: Jump to stored position
<b>Space</b>	: Switch between full screen and menus
<b>N</b>	: Switch between NTSC and PAL mode
<b>.</b>	: Switch Anims on/off
<b>Return</b>	: Overview map
<b>A</b>	: Go to Animation Controller
<b>Z</b>	: Go to Tile Swap Controller
<b>Help</b>	: Request Help / mode

When in key plot mode, the cursor keys control the cursor around the viewed area of the map and either shift key can be used to plot.

The keys on the keypad can be used to select tiles. The tile selected is based upon the tile shown in the top left of the tile selector to the left of the map area. The keys are:

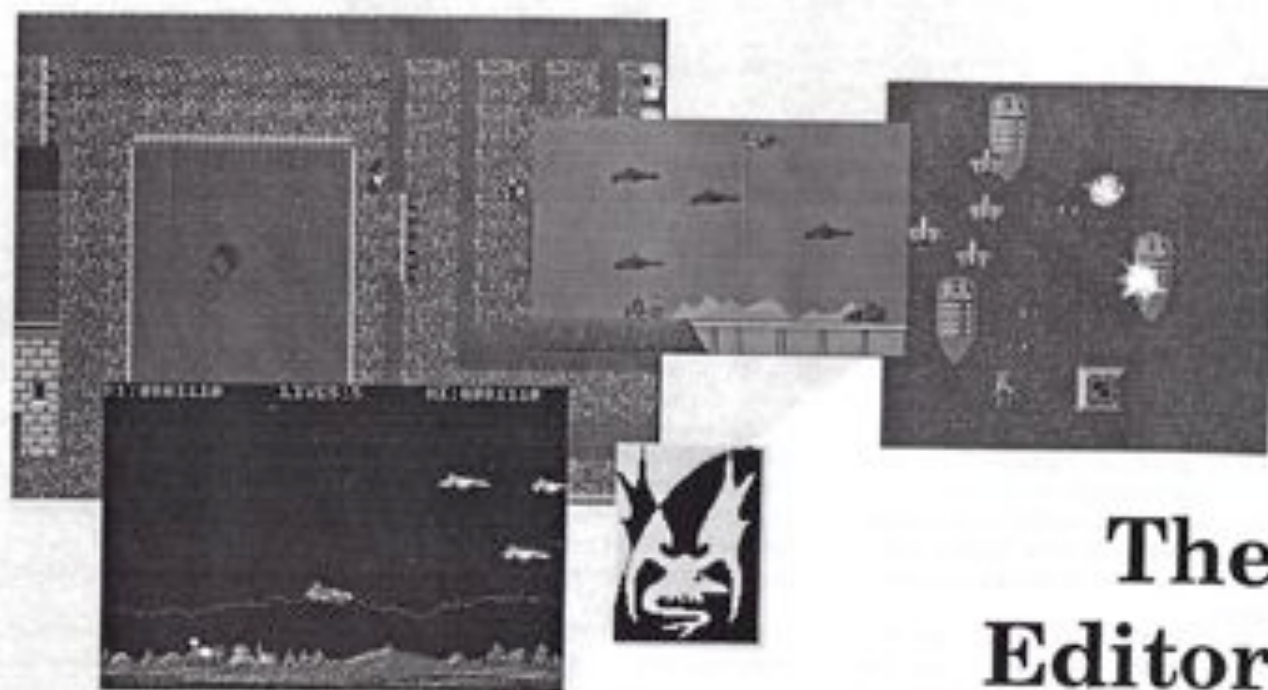
<b>[</b> +0	<b>]</b> +1	<b>/</b> +2	<b>*</b> +3
<b>7</b> +4	<b>8</b> +5	<b>9</b> +6	<b>-</b> +7
<b>4</b> +8	<b>5</b> +9	<b>6</b> +10	<b>+</b> +11

The **8** and **\*** (full stop) keys allow you to move the tile selector through all the tiles.

When in key move mode the cursor keys simply scroll you around the map, the same as the joystick. Holding down either Shift key will make the movements 10 times greater.

The **A** & **Δ** keys allow you to physically move the map viewing area, moving the tile selector off the screen, so that you can see more of the map. Similarly, holding down the left Alt key while dragging the map view area with the mouse will move the screen.





# The Editor Utilities

# The Tile Maker

## Tile Grabbing Utility

Before you design your TOME Maps, you need tiles. Tiles are the small pieces that your map will be made up from and are stored in memory as AMOS Icons. In TOME, the tiles are always the same size as each other, thus the map can be updated and checked with great speed. As explained earlier in the manual, there are 4 different possible sizes of tile. They are: 16x16, 16x32, 32x16 and 32x32.

Tiles are normally drawn as a screen in an art package such as DPaint. This involves making up a screen, starting from the top left corner and working across, then down a line of tiles and so on. For 32x32 pixel tiles for instance, they would be stored on the screen in the following order:-

```
0 - 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9
10 - 11 - 12 - 13 - 14 - 15 - 16 - 17 - 18 - 19
etc.---
```

Depending upon how you like to draw your graphics, you can if you like, put a 1 pixel gap between each tile.

Other graphics artists prefer to draw the tiles as a large block, without the gap.

It is best to design your tiles on a 320 pixel wide lowres screen, as this is the size the Tile Maker was designed to handle. You can make the screen as high as you want, but always remember that TOME can only use 256 tiles.

The Tile Maker is the utility (now built into the TOME Editor) that takes these tile screens and turns them into actual tiles. It has several options, which can save you memory, and make life easier. These will also be detailed here.

When designing your tiles, you should bear in mind the fact that you will need to use them to make up a screen, and to get the most out of them, you need to design them as tiles, rather than designing a screen and cutting it up into tiles. Because of the sizing of the tiles, and the way they are placed, it is also better to design your objects that the tiles will make up with the size of the tiles in mind. It isn't very efficient for instance to design a building block that is 24 pixels wide when using 16x16 pixel tiles, as this will waste half a tile in width and will not be able to connect on at least one side with similar blocks. By making the block 16 or 32 pixels wide, you will be able to connect them together on the map.

Once you have designed your tile screen, you will need to get to the Tile Maker. Select the Utilities Menu (Menu 3) in the TOME Editor and click on the Tile Maker Button (Option 4). This will start up the Tile Maker utility... You should now have a mostly blank screen, with the following Control Menu at the top..



This control menu has all the Tile Maker controls and gives you the ability to:

- Load and Save Tile banks
- Load IFF Pictures for conversion
- Clear the Tile Bank
- View the IFF Picture currently Loaded
- Change the Tile Size
- Set Tile Miser and Line Gap modes.

If you are creating an entirely new tile bank, the first thing you will want to do is clear all the current tiles. You do this by clicking on the **CLEAR TILES** button. The Tile Maker will make sure you really want to delete all of the tiles currently stored in memory, and if it is O.K. by you, will clear them.

You can now decide what tile size you want to use and pick it by clicking on one of the 4 size buttons in the top left. These can only be selected when the tile bank is cleared (so that you can't mix tile sizes).

Now click on the **LOAD PIC** button and select your tile screen that you saved to disk as an IFF file from your paint package. The Tile Maker will also accept packed AMOS screens in memory banks as .Abk files.

Once your picture has loaded in, you can view it (to make sure you loaded the right one) with the **VIEW PIC** button.

If you put a pixel gap between your tiles, click on the **LINE GAP** Button, so that it is set to ON, otherwise, set it to OFF.

If you want the Tile Maker to automatically eliminate any duplicate tiles (Which waste memory, and can cause you problems when programming) Switch the **TILE TIDY** Option on. When activated, this function scans through all the tiles and eliminates any tiles which are duplicated in the tile bank. As a 32x32 pixel tile in 16



colours uses 512 bytes, it only needs the tile miser to eliminate 2 duplicate tiles to save you 1k of memory ! Also, if you have duplicate tiles and accidentally use both of them on the map, you can have problems detecting which tile your player is standing on, as although the tiles look the same, they will have different tile numbers !

Once you have decided on these settings for your tiles, click on the GO ! button. This will activate the Tile grabber and grab all the tiles from the loaded screen. If the Tile Tidy option has been activated, it will automatically scan and eliminate duplicated tiles after the grab.

If there are more than 256 tile in memory after grabbing and Tile Tidying, the Tile Maker will warn you and then eliminate tiles 257 upwards.

If you want to add some new tiles to the current set, then just do the above sequence, without clearing the tiles or setting the tile size. In this way, you will be able to add tiles on, until you run out of memory, or reach the 256 tile limit.

The Load and Save tile buttons are in the Tile Maker only as a convenience, and it is better to do the loading and saving from the file menu in the main editor, which you can return to by hitting the bottom right "Return to Editor" button.

### Tips for designing Tiles

When designing your tiles, try and make them as versatile as possible. If a tile can be used in more than one place in the map it will be much more useful to you, and will save you memory.

The fewer colours you use, the faster your map will update, the more memory you will save and the more colours you will have to concentrate on your bobs.

When assigning colours for your tiles, make sure that there will be enough colours available to draw your bobs, or that the colours you have used can be used for the bobs also.

Try and avoid duplicating tiles when you draw them. Although the Tile Tidy option will eliminate any duplicates, it will change the order that your tiles are stored, possibly making it a little difficult when trying to grab bricks from the selector screen.

Always make sure that the tiles start in the top left corner of the screen as this is where the Tile Maker starts grabbing from. If you offset the tiles by any amount, you will get strange results.

# The Animation Controller

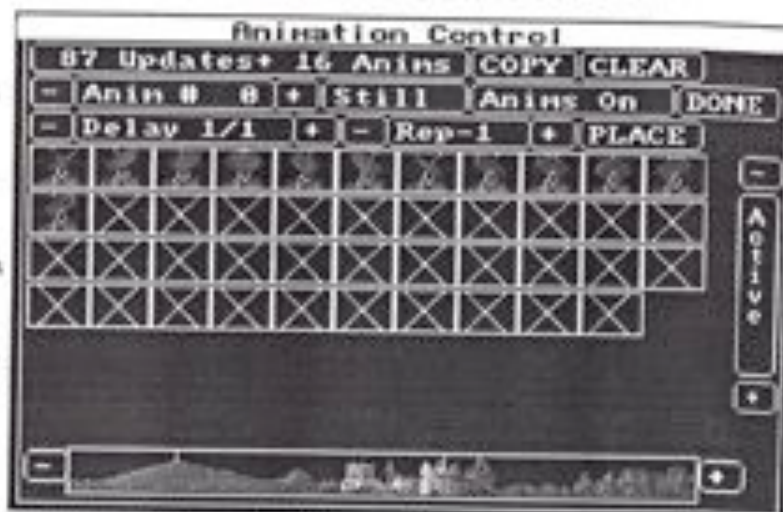
The Animation Control utility in TOME lets you design tile animations for use in your programs. These animations are then saved out as an anim bank, which can be loaded into your program, and used with the **Map Anim Bank** and **Map Anim On** commands.

The Anim Controller is very simple to use. The first button at the top left of the screen allows you to set the number of updates, and the number of anims available. The number of updates must always be at least as many as the number of anims, or not all the anims will be able to be displayed at once. The number of updates and anims is changed with a normal dialog box.

The animation number that you are editing is shown below the updates/anims box, and can be changed with the up/down buttons on either side of it. The anim number displayed in this control is the number you would be using as the Anim number if you were using the **Map Anim** command.

To the right of this control is a button which will show as either Still or Moving. This button is used to switch the anim between being a still anim (which cycles between 2-44 tiles on the spot), or a moving anim, which can be programmed to move around the map in steps of 1 tile.

To the right of the Still/Moving button is the Anims On/Off switch. It is sometimes handy to switch the anims off, so that you can line them up on the map (if you were using several anims in conjunction with one another for instance). If you just want to freeze one anim, the Active / Frozen button is to the right of the actual anim display, down the right hand side of the screen.



On the bottom row of buttons, you have two controls for the delay between animation frames and the number of times that the animation will repeat. When set to -1 the animation will repeat indefinitely. These two controls are exactly the same as the Delay and Repeats parameters in the **Map Anim** command.

To the right of these controls is the Place button. Clicking on Place will take you back to the main editor, ready to place the tile anim that you were editing. Simply click on the point in the map where you want the anim to start. If the animations are switched on, then you will see it start immediately, otherwise it will start as soon as you switch the anims on, either with the switch in the Animation Controller, or by pressing the ` key (above Tab). The co-ordinates you placed the animation at will be stored in that animation's sequence, so that when you save the anim bank, it can be loaded into your program with the anims in position.

The Copy button at the top of the screen is so that you can duplicate an animation. Simply click on Copy, O.K the alert box and select another animation. Clicking anywhere in any one of that animation's display boxes (where the tiles are shown) will copy the previously selected animation to it.

Creating the actual animations is very easy. A tile selector is shown at the bottom of the screen. Simply select the tiles you want in your animation sequence, and click them into place in the animation display with the Left mouse button. Clicking in the display with the Right mouse button will delete that frame. If you are using larger tiles, you can use the up/down control on the right of the screen, to scroll through the frames.

When you are designing a moving anim, it is slightly different. Instead of 44 tile boxes, you are given 14 groups of 3 boxes. In each group, you can put a left / right movement, an up / down movement and a tile. The tiles are placed as normal and clicking in either of the other two boxes will bring up a dialog box so that you can set the amount that that frame will jump in that direction.

Once you have designed your animation sequences, you can save them from the Disk menu as either .Abk or .Anm files (.Abk is a lot more practical to use). They can then be loaded into your program. Simply put the following code in at the start of your program.

```
U=Number of Updates : A=Number of Anims
Load "Myanimbank.Abk",9
Map Anim Bank 9,U,A
Map Anim On
```

The anims can then be controlled with the normal map anim commands, see the commands Map An Freeze, Map An Unfreeze, Map An Move, =Map An At(), =Map An Point() in the TOME Commands section.

# MaPLe

(Map Programming Language)

The MaPLe language is a simple interpreted language specifically written for working with maps. It is quite easy to use, and is intended to be used for those "Let the Amiga do this while I get a cuppa" type tasks.

An MPL program is constructed in the MaPLe editor built into the TOME Map Editor. The MPL editor shows you all 18 MPL commands at the top of the screen, and a series of programming lines (normally they start off blank) below them. At the bottom of the screen is a tile selector.

An MPL program is constructed by clicking on the commands you want, and clicking them into place into the listing. If a command requires a tile as a parameter, you simply select and click the tile into the parameter column after the command. If the command requires co-ordinates, or directions (such as the Move command), clicking on the parameter column brings up a dialog box to change the parameters.

Where commands use Tile parameters, you can place up to 10 tiles in each parameter box. In the case of some commands (like If Tile or Find) MPL will check against all of the tiles (to see if the tile being checked is one of them) and in the case of the Plot command, a tile will be picked from the box at random from the ones there. MaPLe is best suited for complex search and replaces, but can be used for all sorts of things that might not be in the TOME editor.

---

## The MaPLe Command Set

---

### Plotting/Moving commands

<b>FIND (Tile list)</b>	: Search the entire map for the first occurrence of any of the tiles in the parameter list. When one is found the co-ordinates are passed to MPL's tile cursor.
<b>MOVE (+X+Y)</b>	: Move the MPL tile cursor X Tiles Left/Right & Y Tiles Up/Down.
<b>GOTO (Co-ordinate)</b>	: Move the MPL tile cursor to the X/Y co-ordinate shown.
<b>PLOT (Tile list)</b>	: Plot one of the tiles (picked at random) from the tile list at the current tile cursor co-ordinate.
<b>BRIK PLOT (Brick)</b>	: Plot a brick at the current MPL tile cursor position.



### Looping/condition commands

<b>GLOBAL</b>	: The MPL code between the Global & EndGlobal commands is performed on every tile co-ordinate on the map. Very useful for searches.
<b>ENDGLOBAL</b>	: See <b>Global</b>
<b>IF TILE (Tile List)</b>	: Compares the tile at the current tile co-ordinate with those in the parameter list, if it isn't there, then the code jumps to the corresponding <b>Endif</b> command.
<b>ENDIF</b>	: See <b>If Tile</b> . You must always match each <b>If Tile</b> with an <b>Endif</b> command.
<b>REPEAT (Counter)</b>	: Basically similar to the Repeat command in AMOS, except that the Until command is slightly different, and that the number of repeats is normally entered with the Repeat command. Repeats of 1-99 can be entered with the up/down arrows in the dialog box. If you go below 0, you can set a random number of repeats, and if you go over 99 you can set the number of repeats to the Width (Map X) or Height (Map Y) of the map.
<b>UNTIL X (X co-ord)</b>	: basically this checks the X cursor co-ordinate against the parameter. If it is not equal, the program loops back to the previous <b>REPEAT</b> command.
<b>UNTIL Y (Y co-ord)</b>	: The same as <b>Until X</b> except that it compares the Y co-ordinate.
<b>UNTIL OFF</b>	: examines both X & Y cursor co-ordinates. If they are still on the map the program hops back to the matching <b>REPEAT</b> command.
<b>ENDREP</b>	: This is for doing simple counted loops, i.e. repeat 10 times.

### Mode Commands

<b>MAZE ON</b>	: Switches on Maze Drawing Mode. This works exactly the same as Maze Mode in the TOME Editor.
<b>MAZE OFF</b>	: Switches Maze Drawing Mode off.
<b>B FILL ON</b>	: Switches on Brick Fill Mode. This works exactly the same as the Brick Fill Mode in the TOME Editor.
<b>B FILL OFF</b>	: Switches off Brick Fill Mode.

### An example MaPLe program.

Say for instance you had a lot of tiles placed around the map, that were the bottom part of a tree made up of 1 tiles placed one above the other. You can use MPL to search for the bottom part, move up 1 position and place the top part, automatically, over the whole map.



Command	Parameter
GLOBAL	
IF TILE	(Lower Tree Tile)
MOVE	X=0 : Y=-1 ( Move up one place)
PLOT	(Upper Tree Tile)
END IF	
END GLOBAL	

Press the Run button and sit back. MPL will go through the entire map for you.

MaPLe is probably destined to be one of those features that is not used to its full potential in TOME, because it is really only for those situations that would require deep thought or lots of repetitive moving and plotting, but at least you have the reassurance that it is there if you need it!

If you want to experiment with MaPLe, there are a few .MPL demo programs on the TOME disk (in the Maple directory).

Possibly for the next update of TOME (4.3 or 4.4) we'll have the MPL commands as an actual AMOS extension, so that you can actually use them in your AMOS programs!

## The Picture to Map Convertor

### Iff Picture to Map Data conversion utility

This particular utility is designed to be used to help create large maps (Outdoor wilderness maps for instance) or particularly complex shaped maps (Circular space stations etc.) by converting a drawing of the map done in an art package to a TOME Map. Each pixel in the drawing represents one tile, with each colour being mapped to a different group of tiles.

If for instance you wanted to do a wilderness map 200x200 tiles in size, like they use in many of the popular dungeon games, you could draw a map of your wilderness as an IFF screen, using an area of 200x200 pixels. If you draw this map in 4 colours, you could have 4 different types of terrain, each of which can have a group of tiles assigned to it. For instance Grass, Water, Rocks and Trees.

Because of the way the converter works, you can use a completely different palette for this screen from the one you used for your tiles, so you can make the grass light green, the water blue, the rocks grey and the trees dark green. Now you draw your map in rough form and save it to disk as an IFF file.

The converter will handle any IFF screen in 2-64 colours in low or medium resolution, although it can only handle areas up to 320 pixels wide and 256 (200 in NTSC) pixels high.

Before you select the converter, make sure that your map is set to the correct size, using the map re-size control in the General Options menu in required.

When selected from the utilities menu in TOME, the Pic-Map converter will put up a file selector so that you can choose what IFF picture file to load. Once chosen, it will load the file in and put up its settings page.

The settings page will show you the palette of colours used in the screen, and a tile selector. Basically, you just click on tiles in the selector and place them with the left button into the boxes to the right of the colours that you want the tile to be used for. In our example above, you could put several different grass tiles in the box by the Light Green colour. If you want to use the current brick as a brick fill for any colour, simply click on that colour's tile box with the right mouse button.

Because the IFF screen can use different colours from those used for your Tiles, the Converter has a button which allows you to change between the palettes of the Tiles (Icon) and the Picture (Pict). When set to Icon, you will be able to see your tiles in their normal colours, when selected to Pict, you can see the colours of the IFF screen (You should set to PICT to see what colours are used and swap to ICON to choose the tiles). If you want view the picture simply click on the VIEW button.

Once you click on the O.K button you will be shown the screen, and (if the screen area is different from the size of the map) required to select which area of the screen to convert, by positioning a box which represents the map size.

The conversion is relatively simple. For each pixel on the map, TOME picks one of the tiles that you assigned to that colour and places it on the map. Where you assigned a brick, TOME will place the equivalent tile from that brick, using brick fill mode.

As you will no doubt find, this converter is incredibly handy when you need to design a large map in a hurry!

# The Tile Valuer

## Tile value assignment utility

Although it doesn't look it, the Tile Valuer is one of the more powerful sections of TOME. By assigning up to 8 different values to each tile, you can make your programming life a lot easier.

Tile values themselves are explained in the introductory chapter on page 5, if you haven't read it yet, I suggest you do so!

As a refresher though, the tile values are arranged in lists. Each list contains one value for each tile. The values can be in the range of 0-255. Up to 8 lists can be stored in memory at one time, so each tile can have up to 8 values assigned to it.

The Tile valuer controls are very simple. Basically you are shown a screenfull of tiles, with one of their values to the right of them. At the bottom of the screen are the 3 control buttons that let you skip through the set of tiles in pages, change the current value and change the tile list. With each of these controls, clicking with just the left button moves the amount up or down by 1. If you click the Val: Control with the right button however, you can change the amount up or down 10 at a time.

The **Page:** Control jumps through the pages of tiles available so that you can edit all the tiles.

The **Val:** Control changes the current value. To the right of the control you can see the equivalent tile for that value (the value is the tile number), so that if you are assigning tile number to tile values (for instance, what tile it will turn into when destroyed) you can easily pick tiles.

Clicking on the tiles in the display will assign the current value to any tiles clicked.

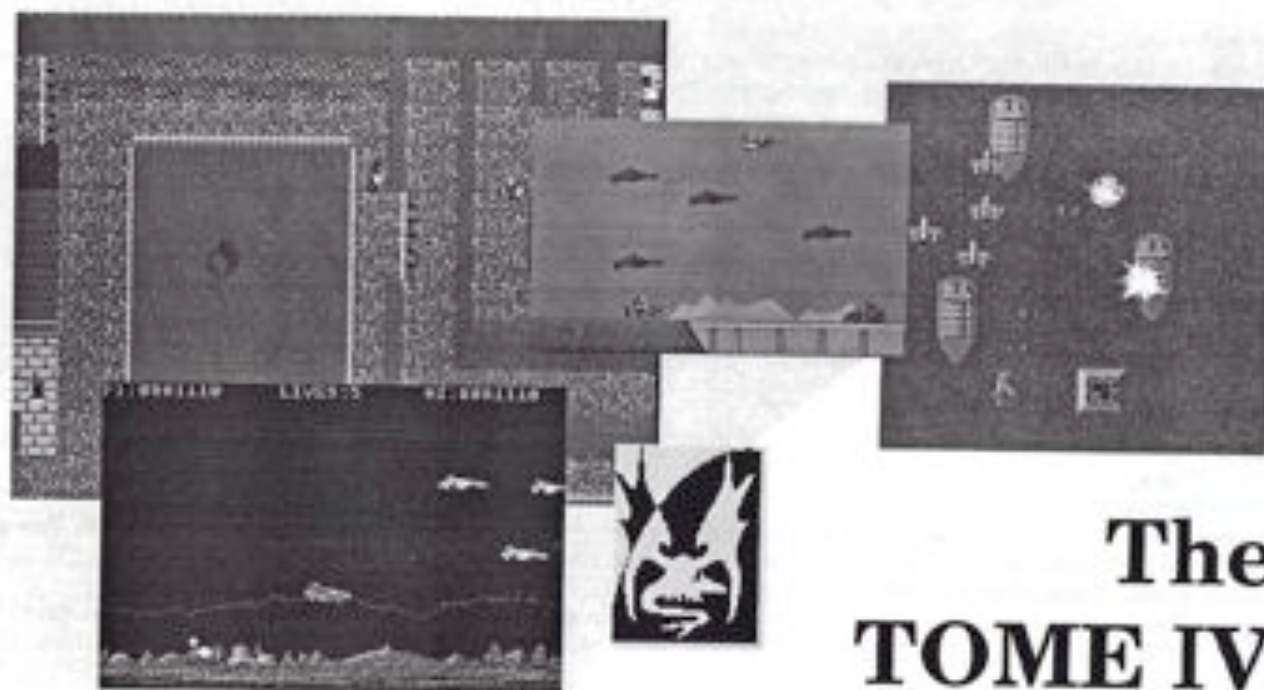
The **List:** Control selects which list you are editing from 0-7, these directly correspond with the List number in the `Tile Val()` function in AMOS.

To the right of these controls are 5 options buttons:-

**Values/Tiles** chooses which way the values are displayed. When set to values, the actual number values themselves are displayed. If set to tiles, they are displayed as the equivalent tiles (again, useful if you are assigning tile numbers).

The **Clear** button wipes the values in the current list to 0 or the tile numbers of the tiles.





# The TOME IV Commands



## TOME Series IV Commands

### Map Do x,y

Draws an area of the map, onto the current screen in the selected map area (see **MAP VIEW**). The top left of this area starts at tile co-ordinates x,y on the map. Thus changing X or Y by 1 would move the view around the map by 1 tile. This can be used for a very simple form of scrolling.

Because Map Do redraws the entire viewing area, it is normally only used at the start of a scrolling routine, and then **Map Left/Right/Top/Bottom** or **Map Handle** are used.

### Map View x1,y1 to x2,y2

Defines the area of the screen to be used for displaying maps. All co-ordinates are in pixels. Changing Map view will not automatically re-display the map, you will have to use the **Map Do** command for that. Map View also doesn't clip the area to be displayed, as it only informs TOME where it can place tiles. The **Clip** command can be used in combination though. See the **MAP\_VIEW\_Demo.Amos** file for an example of this in use.

### Map Left x,y

### Map Right x,y

### Map Top x,y

### Map Bottom x,y

These 4 commands work in the same way as **Map Do**, however they only re-draw 1 edge of the map area. This is useful for high speed scrolling, as using these, you are only re-drawing the edge that has just scrolled on. See the **Edge\_Scroll\_Demo.Amos** file for an example.

Generally, the idea with these commands, is to scroll the screen by one tile, and then use them to re-draw the now vacant strip of tiles that will have scrolled onto the screen.

### Tile Size x,y

Sets the size of the tiles (icons) to be used. x & y can be either 16 or 32 (pixels). The tile size would normally need to be set to the size of the tiles you are using, however, you can set the tile size to different values to space the tiles out, or contract them together (The TOME Editor uses this for its overview mode).

When using the **List Tile** command, Tile Size can be used to counteract the extra pixel line space that List Tile

puts between the tiles. For instance, if the tiles were 16x16 pixels, List Tile would display them at 17 pixel intervals. So doing a Tile Size 15,15 beforehand will place all the tiles together.

#### **Map Plot t,x,y**

Changes the tile at co-ordinates x,y on the map to tile number t (0-255). This does not change the picture on the screen, just the map. However, if you are using the **Map Update** command, the plot will be noted for the next update and the tile re-displayed automatically the next time you do a Map Update.

#### **=Map Tile(x,y)**

Returns the tile number at co-ordinate x,y on the map. This can be used for a rudimentary form of collision detection (checking to see if a particular tile is where the player is), although **Tile Val** is normally used for this, being a lot more powerful.

#### **=Tile Val(x,y,n)**

Returns the tile value of the tile at co-ordinates x,y using the values in tile val list n. n can range from 0-7. The original Series 3 TOME editors only allowed you to edit lists 0-3. The Series IV editor can handle all 8. Basically Tile Val looks at the tile number at X,Y. It then looks up the value in list n for this tile number and returns it. This allows you to assign values to each tile (such as solid, non-solid, slippery, explosive etc.) and check them easily. See the **Tile\_Val\_Demo.Amos** file for an example.

#### **=Map X**

#### **=Map Y**

These return the X & Y size of the map in tiles. These are useful when you want to make sure you are not going off the edge of the map. Fortunately Columbus didn't have these commands, or we wouldn't have America!

#### **Map Paste x,y,px1,py1 to px2,py2**

This command is actually a combination of **Map View** and **Map Do**. Basically, the px & py co-ordinates (pixel) are passed to the Map View routine, and the Map is done using tile co-ordinates x,y. Note that this command does not permanently effect the Map View area.

**Map Bank b**

Normally, the map is stored in Bank 6, however, you can use Map Bank to change the bank number that TOME will use for the map, thus enabling you to use multiple maps.

**Map Brik n,x,y**

This works in the same way as **Map Plot**, but places Brik number n at co-ordinates x,y on the map. As with Map Plot, this does not change the screen, just the map. The Map Update system does not automatically handle Briks (yet), so until TOME Series 5 is released (possibly in March 93) you will have to do a **Paste Brik** command.

**Paste Brik n,x,y**

This command pastes Brik number n to the screen at co-ordinates x,y (pixels). As with **Map Paste**, this command does not change the map.

**=Briks**

Returns the number of briks stored in the current Brik Bank. If there is no brik bank in memory you will get a Bank not reserved error.

**=Brik X(n)****=Brik Y(n)**

Returns the size of Brik number n (in tiles). As with =Briks, having no brik bank in memory when these commands are used will give you a Bank Not Reserved error.

**Brik Bank b**

Changes the Brik Bank number to b. Normally bank 7 is used for TOME Briks, however, you might want to use bank 7 for something else !

**=X Tile(px)**

**=Y Tile(py)**

Translates pixel co-ordinates px & py into tile co-ordinates, based on the current Map View area and tile size. You can use these functions to find out a player's position in the map, although normally you would use the **=Map Pos X(px)** and **=Map Pos Y(py)** commands.

**=Map Pos X(px)**

**=Map Pos Y(py)**

As with **=X Tile(px)** and **=Y Tile(py)**, these commands convert pixel co-ordinates into tile co-ordinates. However, **=Map Pos X/Y** converts the co-ordinates as if they were pixel offsets from the top left of the map, so that you can use the resultant tile co-ordinates with commands such as **Map Plot** and **=Tile Val**. e.g.

Rem MX is current pixel map position,

Rem X is players bob position in screen.

TX=Map Pos X(MX+X)

Rem MY & Y are the same for the Y co-ord.

TY=Map Pos Y(MY+Y)

Rem now get the tile value at that location

TV=Tile Val(TX, TY, 0)

Rem player is standing on tile value TV

#### **List Tile**

Displays the tile bank on the current screen, within the current TOME window (set with Map View). Dead useful when you need to know a tile number. This command automatically puts a space of 1 pixel between each tile. If you don't want this space, simply reduce the tile size by 1, e.g

Tile Size 15,15 : List Tile

You can use this command in place of the old Tile Paster program, simply reduce the tile size by 1, **List Tile** and save the resultant screen to disk using the **Save IFF** command. **List Tile** uses the Map View area though, so make sure Map View is set to a large enough area to display the whole set of tiles.

**Map Update On****Map Update Off**

Switches the Map update system on/off. When switched on, the Map update system will automatically record any Map Plot's, so that when you do a Map Update X,Y any of the changed tiles that are within the screen area will be pasted to the current screen.

**Map Update x,y**

Performs a Map Update. This is basically the same as doing a Map Do x,y but it will only paste tiles that have been Map Plotted since the last update. e.g

```
If BX<>BX0 or BY<>BY0 Gosub NICEMAP : Endif
```

```
Map Update BX,BY
```

The system is flag driven, so if there weren't any tiles plotted in the previous frame, then the routine won't do anything. Basically, Map Update is easy to use, and it's incredibly fast!

Normally you would do a map update just after handling the scrolling using either the NiceMap: subroutine or the Map Handle command.

**Map Anim n , x , y , r , d , anim\$**

Sets up tile animation n at co-ords x,y. r is the number of repeats (-1 = infinite), d is the number of updates between each animation (1 is normal). The animation string is made up of Chr\$'s, 1 for each tile to mutate between, and can be from 0-44 characters in length ("=anim off) e.g

```
Map Anim 0,3,4,-1,1,Chr$(4)+Chr$(7)+Chr$(7)+Chr$(24)
```

You can also make the anims move around the map by making the anim number negative and coding the move and anims in blocks of 3 characters. of Chr\$(DX)+Chr\$(DY)+Chr\$(Anim). Obviously, you can't store as many anims, but you can move them around!

The maximum length for a moving anim is 14 frames. To do negatives, it is basically a case of adding the negative value to 256, i.e -1 would be 255, -2=254 etc. or using the Mkb\$(v) function in the Shuffle extension as this converts a 1 byte value in the range of -128 to 127.

The anims are updated every Map Update.

As an alternative to whacking in huge great strings AMAL style, you can use the **Animation Controller** in the TOME editor to design the Anims, and save them as a bank, which can be loaded (or Blood'ed) into your



program, Pointed out to TOME with the Map Anim Bank command and then activated with the **Map Anim On** command.

#### **Map Anim On**

Activates the Map Anim.

#### **Map Anim Off**

Activates the Map Anim... Not!

#### **Map An Freeze n**

This command temporarily freezes animation number *n*

#### **Map An Unfreeze n**

This commands thaws it out again.

#### **=Map An Point(n)**

Map An Point(*n*) returns the frame or step that animation *n* is currently displaying.

#### **Map An Move n,x,y**

This very simple command allows you to change the position of anim *n* on the map to map co-ordinates *x,y*.

#### **=Map An At(x,y)**

This function will return the map anim that is currently at map co-ordinates *x,y*. If no anim is present at this position then the function will return a zero.

#### **Map Anim Bank bank, n,a**

This reserves a bank for the map update list (if you are using map update) and the map anim list (if you are using tile anims). The bank should be reserved to *a* (The max number of updates) \*8+ *a* (The max number of anims) \*64 + 4. The following function works this out for you...

**=Map Ab Length(n,a)**

Returns the correct length required for a bank of **n** updates and **a** anims.

an example of the above 2 commands would be.

```
Reserve as work 9,Map Ab Length(32,16)
```

```
Map Update Bank 9,32,16
```

When using Map anims, you will need at least 1 update for each anim (otherwise the anims will not automatically update the screen). Remember to also include enough updates for any **Map Plot's** you will be doing each frame.

**= Map Length( x , y )**

Returns the file length of a TOME map **x \* y** tiles in size. (basically **x\*y+4**). e.g

```
L=Map Length(100,100)
```

```
Reserve as work 6,L
```

```
Bload "mymap.map",start(6)
```

**=Map Scan X(t,x1,y1 to x2,y2, mode)****=Map Scan Y(t,x1,y1 to x2,y2, mode)**

Returns X & Y co-ordinates of the tile or tile value that you want to scan for.

If mode is set to zero, then the command will search through the current map from **x1,y1** to **x2,y2** for tile number **t**. If mode is greater than zero, then TOME will do the same search, but for tile value **t**, from list **mode-1**.

An example of use would be...

```
Rem Search for the first occurence of Tile number 3
```

```
TX=Map Scan X(3,0,0 To Map X-1,Map Y-1,0)
```

```
If TX<-1
```

```
TY=Map Scan Y(3,0,0 To Map X-1,Map Y-1,0)
```

```
Endif
```

**=Map HX(x)****=Map HY(y)**

Returns the result of  $x \div \text{tile size}$  and  $y \div \text{tile size}$  (for use with the **Nicemap** routine or the **Map Handle** command).

**=Map FX(x)****=Map FY(y)**

Returns the result of  $x \bmod \text{tile size}$  and  $y \bmod \text{tile size}$  (for use with the **Nicemap** routine or the **Map Handle** command).

**Tile Type Bank b**

Changes the bank number used for Tile Values (normally set to 8).

**=Tme Credit\$**

Returns a string containing the Credits for the TOME extension. This can be used if you are a reviewer for a magazine and don't know who to credit !

**=Tme Ver\$**

Returns a string containing the TOME Version number. This is handy if you spot anything unusual (i.e. a bug), so that we know what version of the TOME extension you are using.

**=Map Base**

Returns the base address of TOME's data table, useful for advanced programming and a couple of tricks !

**=Map Check**

Scans through the map and makes sure that the tiles used in the map are available in the tile bank. If they are not, it replaces them with tile number 0. This function returns the number of tiles it changed. If you are loading a map when you aren't sure whether it is supposed to go with the tiles you have loaded or not, it is a good idea to do Map Check, as it will stop any chance of TOME displaying tiles that don't exist.

**Map Handle screen,x,y**

Basically, this is the Nicemap routine, with x & y being the MX & MY variables normally used.  
e.g

```
Map Handle 1,MX,MY
```

instead of

```
HX=MX/Xsize
HY=MY/Ysize
FX=MX mod Xsize
FY=MY mod Ysize
If HX<>HXO or HY<>HYO Gosub NICEMAP : Endif
HXO=HX : HYO=HY
etc.
```

Obviously, this will speed up TOME map scrolling that was previously using the Nicemap routine quite considerably !

**Map Handle Init**

This command is used to initialise the internal variables used by Map Handle. It should be done before you first use Map Handle, and any time where you will be jumping by more than one tile, or resetting the map.

**Tiny Map x,y,size**

The tiny map command is a simple version of the **Map Do** command, that works by taking the 1st tile value for each tile to be displayed, dividing it by 16 and displaying this result as a tile from the Tiny tile Bank (normally bank 10).

This is handy when you want to do things like infra red scans and dungeon overviews etc, as each different tile in the Tiny Map represents a particular range of 16 tile values. See the **Tiny\_Map\_Demo.Amos** file for an example.

Parameter wise, Tiny Map is the same as Map Do, except that it has an extra **Size** parameter, which is the size of the tiles to be used. So using **Size=1** would mean that Tiny Map would display a very small map using 1x1 pixel tiles from the Tiny tile bank.

#### **Tiny Bank n**

This command tells TOME where the tiles for the Tiny Map command are stored. Normally this will be bank 10, so that if you are using **Ctext** (or **Ctext 2**) you can define the Tiny Tiles as the first 16 icons in your font bank. A Tiny Tile bank is very simple, you just need 16 icons, set to the size that you intend to use Tiny Map with. Each tile will represent 16 Tile Values i.e. Tile 0 is values 0-15, Tile 1 is Values 16-31 etc.

#### **Map Fall n**

The Map fall command will scan the entire map, using the second tile value list. Depending upon the value of each tile on the Map and the surrounding tiles, Map Fall will make the tiles fall downwards, replacing them with tile number **n**. Surprisingly enough, this command is very useful when you are writing Boulder Dash and Repton style games!

The following values are used in the second tile list to show whether or not a tile will fall :-

- 0: Tile is not solid, falling tiles will go right through!
- 1: Tile is solid, will not fall, and tiles will not roll off it.
- 2: Tile is solid, will not fall, but tiles will roll off it.
- 3: Tile is solid, falls but tiles will not roll off it.
- 4: Tile is solid, falls and tiles will roll off it.

#### **Map Swap Tile t1,t2**

This command will search the map for all occurrences of the tiles **t1** and **t2** and swap them over. Great if you want water to ripple, simply have 2 different tiles and Tile Swap them every frame or so.

Because this command scans the entire map for occurrences of the tiles, it is not very practical for large maps, unless you don't need them to update very rapidly.

The Tiles changed by Map Swap Tile are passed to the **Map Update** system, so as long as you have enough updates available, they will be automatically updated on the screen, as if you had used **Map Plot**.



**Tile Tag Set tile,n**

The tile tag system is a very simple way of monitoring the map for occurrences of particular tiles. Useful for things such as enemy generators (like in Gauntlet), alarms and so on. Basically, Tile Tag allows you to warn TOME to watch out for up to 8 tiles. If these tiles are plotted on the screen with Map Do/Left/Right/Top/Bottom, Map Update or Map Handle, then the Tile Tag variable will have a bit set for the relevant tile, and its co-ordinates will be put into the «Tile Tag X and Y variable for that tile.

The Tile Tag Set command is very simple, you do Tile Tag Set tile,t which makes TOME remember that tile as tag number t and whenever that tile is plotted bit t-1 (0-7) will be set in the Tile Tag variable.

The Tile tag number can be in the range of 1-8 and the tile number can be in the range of 0-255 (0 switches off that particular tag. Obviously, this means that you cannot use tile number 0 with this system.

**«Tile Tag**

This returns a bitmap of the 8 tile tags and clears itself. This bitmap represents all 8 tags by 1 bit each and is used to check whether any tag or tags occurred in the last frame. e.g

Tile Tag Set 4,1 : Rem watch out for Tile 4 as tag 1

.....

....

Map Handle 1,xx,yy etc

...

....

TT=Tile Tag

If TT

    If TT and 1

        Rem Tile 4 was plotted this frame !!!

    Endif

Endif

Obviously, this system is quite handy!

**«Tile Tag X(n)**

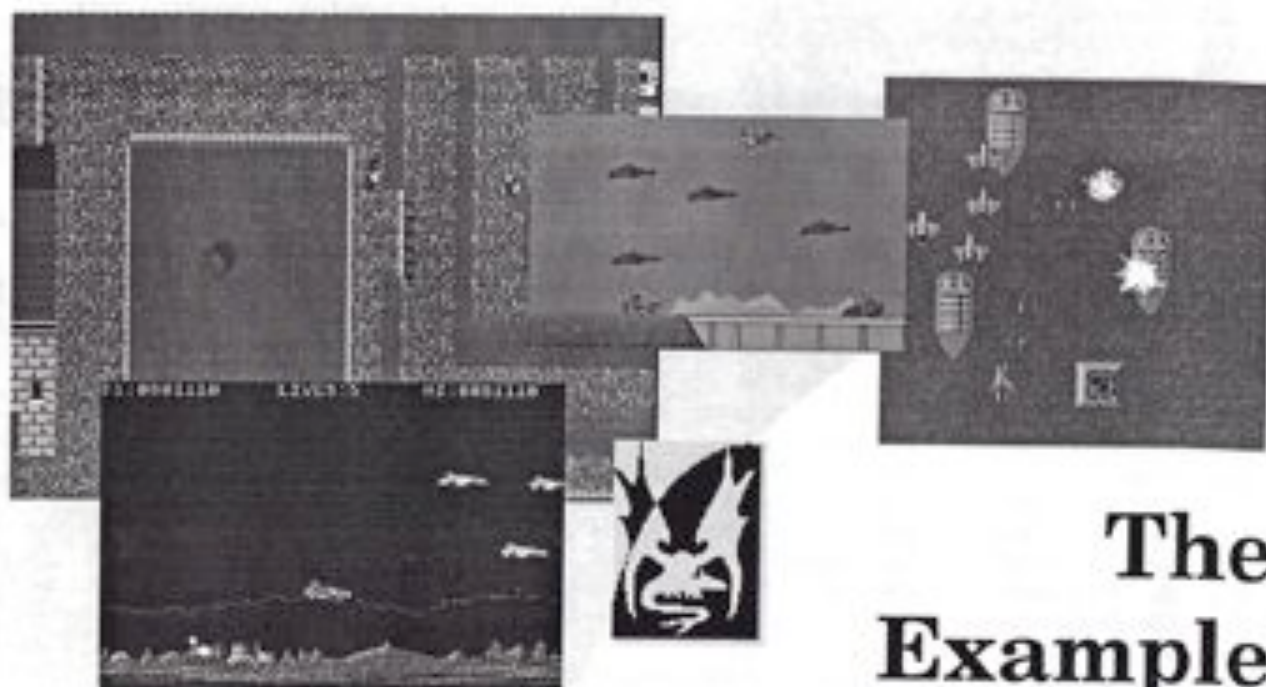
Returns the X co-ordinate that the tile tagged with tag number n was last spotted at.

**Map Set Zone zone,x1,y1 To x2,y2**

This command works almost exactly the same as the **Set Zone** command, except that the *x* & *y* parameters are in tiles, and the zones apply to the map. The **zone** number can range from 1-the number of zones reserved.

**=Map Zone(x,y)**

This function returns a zero if the map co-ordinate *x,y* is not in a map zone, or the zone number that the co-ordinate is in.



# The Example Programs

## The Programs

So that you can see how to use the TOME Commands in your programs, we have supplied a few (!) examples on the disk. While the graphics are © Shadow Software, feel free to use the code from these examples in your own programs. All the commands are demonstrated in the various examples, some more than others. The following list of example programs shows you which commands are demonstrated by them.

Program	Technique demonstrated	Commands, routines demonstrated
Tomedemo1.Amos	Loading and Displaying Maps	Map View, Map Do, map loading
Tomedemo2.Amos	Simple Movement around a map	Map View, Map Do, map loading
Map_View_Demo.Amos	adjustment of map viewing area	Map View, Map Do
Map_Paste_Demo.Amos	Multiple Map Display	Map Paste
Edge_Scroll_Demo.Amos	High Speed Scrolling	Map Left/Right/Top/Bottom
Map_Handle_Demo.Amos	Fine Scrolling	Map Handle
Tile_Size_Demo.Amos	Sneaky tricks with Tile Size	Tile Size
Tile_Val_Demo.Amos	Using Tile Val in a platform game	Tile Val, Map Pos X, Map Pos Y
Tile_Tag_Demo.Amos	Plotting and Spotting Tiles	Tile Tag, Map Plot, Map Update
Map_Fall_Demo.Amos	Rockfalling with Map Fall	Map Fall, Map Plot, Map Update
Map_Bank_Demo.Amos	Multiple Levels	Map Bank, Map Handle
Briks_Demo.Amos	Using TOME Briks in a game	All Map Brik commands
Map_Anim_Demo.Amos	Using Map Anim commands	All Map Anim commands
Map_Base_Demo.Amos	Full list of TOME data Table	Map Base, For advanced users
Tiny_Map_Demo.Amos	Displaying "Infra Red" Maps	Tiny Map, Tiny Bank, Map Check
Map_Info_Demo.Amos	Getting info from the map	Map Check, Map X, Map Y, Tile Count
Swap_Tile_Demo.Amos	Simple Swap animations	Map Swap Tile, Map Update
Scanning_Demo.Amos	Using Map Scan X/Y	Map Scan X, Map Scan Y
Map_Zones_Demo.Amos	Using the Map Zone commands	All Map Zone commands
Game	Plot	
Star_Gun.Amos	Multi mission horizontal shoot and bomb em up.	
Maze_Man.Amos	8 way scrolling game. Collect all the disks, and shoot or avoid the zombies.	
Velcro_Grub.Amos	Blast everything. 1 or 2 players, horizontal band parallax scroller	
Vert_Scroll.Amos	Simple Vertical blast-em-up	

### Star Gun

This is a fairly simple shoot and bomb em up game, based on several old 8 bit classics. Basically, you are given 5 missions, in which you must bomb certain targets. There are several enemies trying to destroy you, but you are armed with both bombs and lasers. When above the level of the mountains in the background, you can fire your lasers, but as soon as you go below this level, you will only be able to drop bombs, of which you can carry 8 at a time.

If you shoot down an enemy ship, it will release a supply pod, which you can collect to get another 8 bombs. If you complete the first 5 missions, you have to do them again, but with more and different enemy ships!

## Make Me

This is the improved version of the Maze Man game that was supplied in TOME V3. In this version you have to collect the disks that are spread about the maze, avoiding the Zombies that will appear from the graves.

One major difference between this, and normal scrolling maze games is that I have added momentum to the player's movement!

## Velcro Group

A Very simple horizontally scrolling shoot-em-up for 1 or 2 players. Basically shoot everything!

## Vertigo

Another very simple shoot-em-up. This time the basis for a vertical shoot-em-up game.

Obviously, as TOME Series IV is developed, programs may be modified, and more added to the disk. So send in your TOME Registration card straight away, so that you can get these later updates.

For the latest up to date information on new features and updates for AMOS TOME, subscribe to the AMOS Club Newsletter. Subscription costs £12 (£15 overseas) and gets you 6 information packed newsletters, access to the AMOS Helpline and special offers on AMOS Software (including software that is only available to AMOS Club members). Send Cheques or Postal orders made out to "The AMOS Club" to

If Subscribing from overseas, please note that we can only accept cheques made out on a London bank, in Pounds sterling (or GBP). Eurocheques or U.S. Postal Money Orders in Pounds can also be accepted. Cheques made out in foreign currencies cannot be accepted.



## Other AMOS Products

Shadow Software have been quite busy over the last couple of years, developing a range of products in conjunction with the AMOS Club, to make AMOS more powerful, and to make programming easier.

### TOME Goodies Disk 1:

This add on for TOME includes 3 mini games, all written using AMOS TOME:

The Dungeon: A mini 3D dungeon adventure, showing how you can write Dungeon type games with AMOS TOME.

Green Flag: A simple Isometric scrolling game, demonstrating the use of Isometric maps.

Magic Forest II: The full version of this popular platform game as source code, so you can see how it was written. The TOME Goodies Disk 1 costs £5.00 (£7.00 overseas)

### TOME Goodies Disk 2:

At the time of writing this manual, the second TOME Goodies Disk is still being finished. It will contain:

Ninja Cowboy: A full Scrolling Beat and Shoot-em up game.

Rock Crash: Hunt for the fuel bearing ore to recover your spaceship. A full Boulderdash type game.

Bandits 1: Straightforward vertical shoot-em-up.

The TOME Goodies Disk 2 will cost £5.00 (£7.00 overseas). If all goes well, it will be released mid November 1992, although it is possible that the games may change.

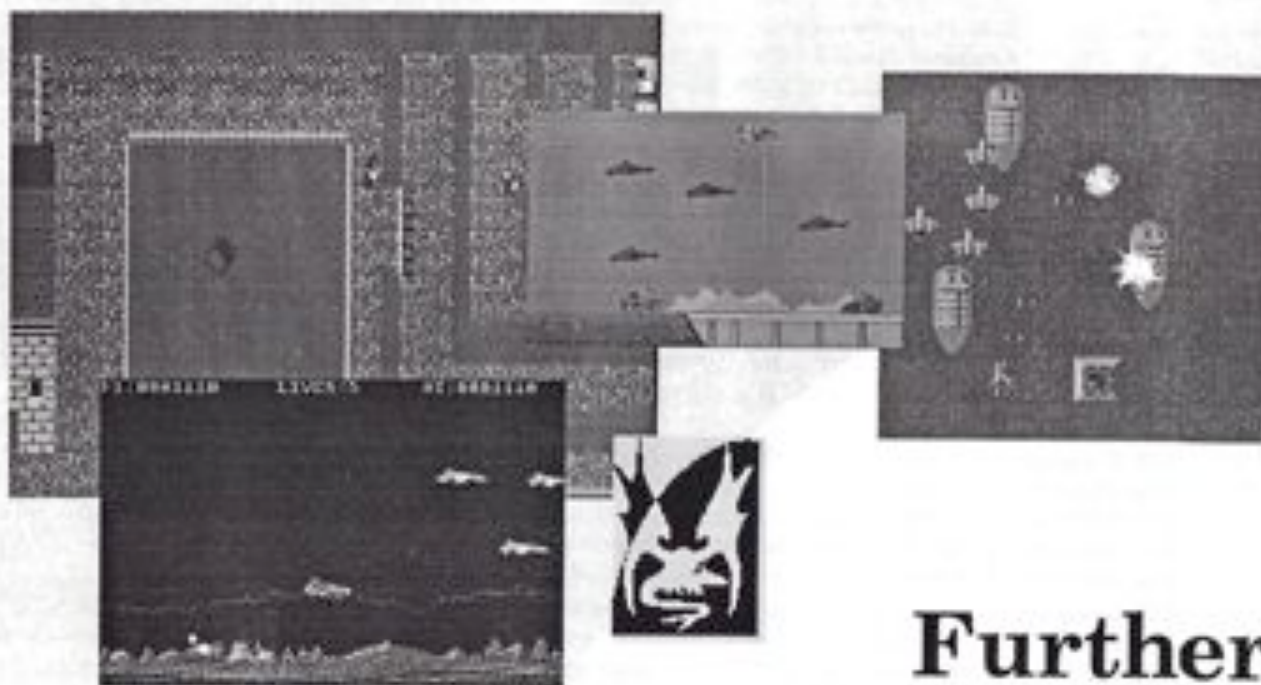
### AMOS CText V2.5

This latest version of the Colour Text extension for AMOS gives you 22 commands to handle icon bank based fonts, which can be kerned, re-coloured, word wrapped (pixel accurate) and overlaid onto graphics.

Ctext V2.5 is only available to AMOS Club U.K members, and costs £11 (£10 if Ctext 1.32 owned). See the AMOS Newsletter for details.

### AMOS Sprite X V2.0

This is the latest version of the powerful image editor. It can edit sprites, bobs, icons, screens and even workbench icons. Its grabbing utility is so powerful that it can automatically work out where all the sprites on a screen are, grab them accurately, and even put hotspots on them ! See the AMOS Newsletter for Details



## Further Information

## The History of TOME

TOME Series IV is now the result of around three and a half years development work. The first proper version of TOME was developed back in 1989 specifically for the game YOMO, written in STOS on the ST and released as part of Mandarin Software's "Games Galore" package. TOME V1.0 was a short (768 byte) machine code bank, and a 1 screen editor, where only about a third of the screen could be used to display the map. TOME Version 1 was never released, except for about 4, which were sent out as temporary packages to those that had put in very early orders for TOME V2.0.

V1.0 is a rarity, as even I've only got one copy, and that's somewhere at the bottom of a disk box !

TOME V2.0 was the first production version of the system, being released on the ST for STOS. It was on 2 disks (360K disks were the standard on the ST at the time), had an 18 command extension and an editor more along the lines of the current one, in that it had a main menu, which selected sub menus. Several amazing features, such as the help mode, user tile palette, random draw and block (STOS TOME's equivalent of a Brick) fill were also introduced in this version. To date, TOME V2.0 (and the slightly later V2.1 which introduced the edge scrolling commands) have been used on several projects, not just on the ST. Despite being blown away by TOME Series IV, TOME 2.1 is the most powerful map editor on the ST still, and still ranks among the top 10 most powerful map editors available.

When AMOS was released we started work on the Amiga. After quickly writing TAME (The Temporary AMOS Map Editor) to go with the AMOS package, a program which I'm not incredibly proud of, as we were told it had to be finished in two weeks, had to work in 512K and had to use menus (as AMOS didn't have a menus demo at the time). As menus are not something you would want to use with a map editor, TAME didn't come out too well, and we got started on TOME Version 3.0 before everyone lynched us !

TOME V3.0 was the first AMOS add on to be released, on March 8th (my birthday) 1991, 3 months before the AMOS Compiler ! Version 3.0 had 27 commands, full niceness controls (2.0 had limited control) and thanks to the power of AMOS it had multiple screens in the editor, making the user's life even easier. Because of a low budget, the manual was written as a hyper text program on the actual TOME disk. Despite problems with the installation program on the first 30 disks released (Which was immediately rectified by sending out new disks) TOME V3.0 took off, and soon all the major AMOS programmers were using it. With the release of the AMOS Compiler, TOME V3.1 was released (with upgrades being sent to all the V3.0 users) and TOME had grown to 36 commands. The TOME Goodies Disk 1 was also released around this time. In early 1992, TOME V3.2 was

released, with the Tile Maker program being built into the actual editor, and a couple of minor tunings to the extension and the rest of the system.

Development on TOME Series IV itself actually started before TOME V3.2 was released. As comments and ideas came in from TOME users, we noted them down and worked out ways of incorporating them into the new super version. Ideas such as animating tiles, easy updating when plotting tiles, map zones and tile tagging were all suggested for the extension. The editor got a complete revamp too. To make the icons more legible, Adam sat down and redesigned them all in medium, rather than low, resolution. This enabled me to add more controls in the editor, and to re-arrange the menus slightly. To compliment the Tile Maker, Pic to Map Converter and Tile Valuer already in the editor, I added the Animation Controller and MaPLe, TOME's own programming language. To give the user a bit more power (you mean 64 commands more isn't enough !), I put the Shuffle V2.6 extension on the TOME disk, bringing the total number of commands up over the 110 mark ! The one bit that was most requested however, was this 80 page manual that has just taken me the last 4 months to write.

Once this package is finished, I'll be sitting down and writing 3 new games for the TOME Goodies Disk 2, which will show off TOME Series IV's new commands to their limits. I've told everyone they will be finished by mid November, so I've got one and a half months to do it !

I'm now considering what I'll be putting in TOME Series 5. I've already got a version of TOME IV running using 511 tiles instead of 256. This will become available in 1993 to registered users of TOME IV as an extra developers version. For TOME V I'm going to let you use 8192 tiles, which can be flipped vertically and horizontally. I'm also developing MaPLe as a separate extension, so that you can actually use it within your programs ! Don't expect instant miracles however, I have a few other projects to finish first, and I would be hoping to finish Series V towards the end of '93 or probably some time into 1994, so get your requests in now !

I hope you enjoy using TOME, and I hope to see one of your games on the shelf (or preferably being taken off the shelf and being sold to a very happy customer) the next time I go down to my local computer shop.

Aaron Fothergill (October 1992)

A Big Thank you again to all of the following...

Adam Fothergill, Len and Anne Tucker, Sandra Sharkey, Peter Hickman, Rod Pascoe, Phil South, Ben Ashley, Stephen Hill, Norm Allen, Dave Lazerek, Hendrik Heimer, Bob Baker, Robert Brady and Jason Tucker. Thanks also to Mike Oldfield for playing the most excellent music ever, and a brilliant concert in Edinburgh Castle !

### Technical Help

If you experience any problems in using AMOS TOME, contact us direct on [REDACTED] between 2pm and 7pm U.K Time. This is the AMOS Club helpline, so please state either your AMOS Club membership number, or that you are a TOME Series IV user. We will be glad to help.

If you are overseas, and bought your AMOS TOME from a local distributor then contact them first, as they will normally be able to help you. If not, then they will put you in touch with us. (This stops you having to ring long distance, or wait for a couple of weeks for a letter to get through)

### The Shuffle Extension

This extension has been included in the TOME Series IV System for you to use. You will find an example of each of the commands in the `Shuffle_Demo.Amos` program on the TOME IV disk. If you would like a full printed manual for this extension, send £5.00 (£7.00 overseas) to Shadow Software, at the address below, or contact your local dealer for details.

Shadow Software, and The AMOS Club  
(Help and new product information)

[REDACTED]

Allen Computer Supplies  
(Australian TOME and AMOS Club Distributor)

[REDACTED]

David Lazarek  
(USA TOME Distributor)

[REDACTED]

[REDACTED]



## TOME Bank/File Formats

Just because TOME comes with an excellent extension for AMOS doesn't mean that you have to use it for only AMOS software. In fact TOME has been used to create maps for all sorts of programs!

Just for those of you who will need to know, here are the formats of the various TOME data files and banks, so that you can incorporate them into your routines. (A tip for the chap doing the rather large map on the Cray II, try creating it as several 32768x32768 maps and using the Cray's file handler to fix them together).

Most of the data formats are very simple, this means that they work faster and are more memory efficient.

### Map Bank

This format is very simple, first we have 2 words for the Width and Height of the map (MapX.w and MapY.w), then we have the data for the map itself, starting at the top left of the map, and working along the top row, 1 byte for each tile, then going to the next row and so on. e.g

```
MapX.w      5
MapY.w      3
data        0 1 2 3 4
            5 6 7 8 9
            A B C D E
```

### Tile Value Bank

These banks are 256 bytes per value list (from 1-8), and are very simply blocks of 256 bytes, 1 byte per tile number, with each list following the previous.

```
e.g
List1       256 bytes
List2       256 bytes
etc
```

### Brik Bank

The brik bank is quite complex by TOME's standards!

First we have the number of briks in the bank as a word. Briks.W

Then we have a lists of Longwords, each of which is an offset to the data area of a particular brik. e.g

```
Brik1Off.L
Brik2Off.L
etc...
```

At the end of these is the first Brik data area, which is almost exactly the same as the map bank format, i.e 1 word each for the Width and Height (BrikX.w and BrikY.w) and then the tiles used in the brik stored as single bytes.

#### Animation/Update Bank

This bank stores data for the animations, as well as temporary data for the Update system.

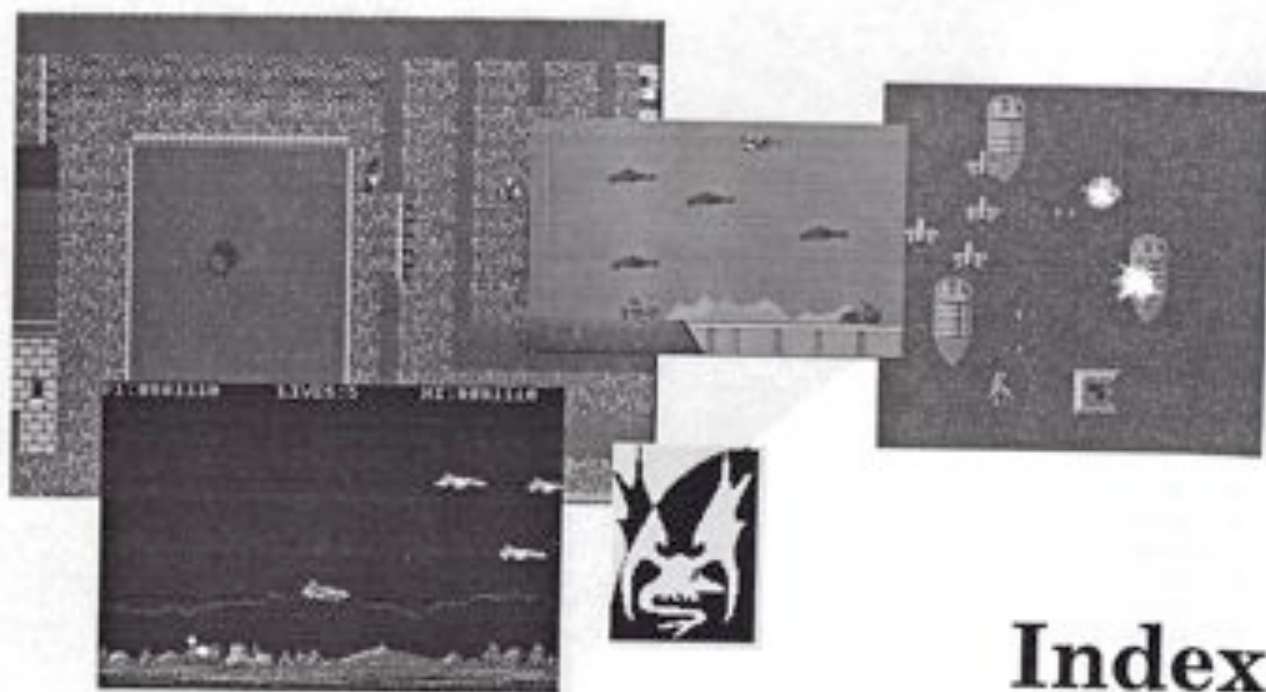
The bank starts with the offset to the animations as a longword, and then 4 words for each reserved update. These are:

Tile.w	Tile to place
X.w	X co-ordinate to place it at
Y.w	Y co-ordinate to place it at
Dummy.w	spare word (its faster to store as 8 byte blocks than 6 byte blocks)
After these blocks follows the Animation data, for each animation the following data is stored:	
ANX.w	X co-ord of animation
ANY.w	Y co-ord of animation
ANRL.w	number of repeats
ANS.w	Delay between updates
ANC.w	Counter
ANL.w	Length of animation string
ANP.w	pointer to position in string
ANBASE.l	Offset into map data of X,Y co-ord (pre calculated for speed)
ANMOVE.w	Animation Still/Moving flag
ANSTR.b	44 byte string for animation data

#### Map Zone Bank

This is again a very simple format. It is made up of 1 word to show the number of zones reserved, and then 4 words for each zone, being:

X1.w	Top Left X co-ord
Y1.w	Top Left Y co-ord
X2.w	Bottom Right X co-ord
Y2.w	Bottom Right Y co-ord.



## Index

# Index

## A

AMOS Club Newsletter 63

## B

Bank/File Formats 69

Briks 5

## C

Copyright 5

## E

### Editor

Add Tile to Bank 28

Animation Controller 24, 39

Automatic Map Draw 26

Bar Mode 25

Brik Draw 20

Brik Fill Mode 28

Clear Map 26

Co-ordinates display 31

Colour Controller and Palette Selector 29

Command Icons 17

Credits 22

Cursor Chase mode 21

### Editor (Cont.)

Cut Brik from Map 27

Cut Brik From Tile Picture 27

Delete Tile from Bank 28

Disc Mode 25

Disk Options 23

Draw Mode 25

Drawing 19

File Formats 21

Find and Replace Tile 31

Find Tile 30

Fine Box, Bar, Circle & Disc 29

Fine Cut & Paste 29

Fine Draw, Flood Fill, Line & Airbrush 29

Fine Edit Tile 29

Flip Screen Grid 22

Flip Tile 29

Flood Fill Mode 25

Full Screen Button 18, 31

General Controls 18

Getting Around 19

Goto Location 30

Grab Current Tile from Screen 28

Grab tile to User Palette 25

Grid Options 22

Hard Scroll Map 26

Help! 18, 31

Help! Button 17

Keyboard Control Mode 21

Keyboard Controls 32

## Editor (Cont.)

- Load/Save Bricks 23
- Load/Save Map 23
- Load/Save Tile Anim 23
- Load/Save Tile Values 23
- Load/Save Tiles 23
- Locator Marks 19
- Map Display window 17
- Map Overlay Grid 22
- Map Size 22
- MaPLe 41
- MaPLe Command Set 41
- MaPLe Editor 24
- Maze Draw Mode 26
- Menu 1, General Options 21
- Menu 3, Utilities 24
- Menu 4, Edit Menu 1 25
- Menu 5, Edit Menu 2 26
- Menu 6, Brick Options 27
- Menu 7, Tile Bank Editor 28
- Menu 8, Fine Draw Menu 29
- Menu 9, Locator Controls 30
- Menu Selector Icons 17
- Niceness Control 21
- Overview map 33
- PAL or NTSC operation 21
- Paste Brick 27
- Pick Brick 28
- Picture to Map Converter 24,43

## Editor (Cont.)

- Random Draw 20, 27
- Recall Location 30
- Right Button "Useful" functions 18
- Scatter 27
- Scroller Controller 17, 31
- Select Colour & Fill Pattern 29
- Snooze Corner 17
- Space Bar 19
- Store Location 30
- The Editor Utilities 35
- The Menus 21
- Tile Maker 24, 36
- Tile Tidy 37
- Tips for designing Tiles 38
- Tile Screen Overlay Grid 22
- Tile Selector Palette 17
- Tile Swap controller 24
- Tile Valuer 24, 45
- TOME IV Commands 47
- Window Dragger 17
- Example Programs 61
- Bricks\_Demo.Amos 62
- Edge\_Scroll\_Demo.Amos 62
- Map\_Anim\_Demo.Amos 62
- Map\_Bank\_Demo.Amos 62
- Map\_Base\_Demo.Amos 62
- Map\_Fall\_Demo.Amos 62
- Map\_Handle\_Demo.Amos 62
- Map\_Info\_Demo.Amos 62



## Example Programs (Cont.)

Map_Paste_Demo.Amos	62
Map_View_Demo.Amos	62
Map_Zones_Demo.Amos	62
Maze_Man.Amos	62
Scanning_Demo.Amos	62
Star_Gun.Amos	62
Swap_Tile_Demo.Amos	62
Tile_Size_Demo.Amos	62
Tile_Tag_Demo.Amos	62
Tile_Val_Demo.Amos	62
Tiny_Map_Demo.Amos	62
Tomedemo1.Amos	62
Tomedemo2.Amos	62
Velcro_Grub.Amos	62
Vert_Scroll.Amos	62

**F**

Foreword	1
Further Information	65

**H**

History of TOME	66
-----------------	----

**I**

Installation	3
Troubleshooting	4

**L**

Loading & Displaying a Map	7
----------------------------	---

**M**

Map Do	9
Map Editor	16
Map Length	9
Map Plot	11
Map Tile	10
Map View	9
Map X	10
Map Y	10

**O**

Other AMOS Products	64
AMOS CText V2.5	64
AMOS Sprite X V2.0	64
TOME Goodies Disk 1	64
TOME Goodies Disk 2	64

**R**

Range	10
-------	----

**S**

Scrolling	12
Shuffle Extension	68

## T

Technical Help 68  
 Terminology 5  
   Tile Numbers 5  
   Tile Values 5  
   Tiles 5  
   What is a map? 6  
 Tile Bank 7  
 Tile Numbers 5  
 Tile Size 7  
 Tile Val 11  
 Tile Values 5  
 TOME Commands 48  
   = Map Length 54  
   = Brik X 50  
   = Brik Y 50  
   = Briks 50  
   = Map Ab Length 54  
   = Map An At 53  
   = Map An Point 53  
   = Map Base 55  
   = Map Check 55  
   = Map FX 55  
   = Map FY 55  
   = Map HX 55  
   = Map HY 55  
   = Map Pos X 51  
   = Map Pos Y 51  
   = Map Scan X 54

## TOME Commands (Cont.)

=Map Scan Y 54  
 =Map Tile 49  
 =Map X 49  
 =Map Y 49  
 =Map Zb Length 59  
 =Map Zone 60  
 =Tile Count 59  
 =Tile Tag 58  
 =Tile Tag X 58  
 =Tile Val 49  
 =Time Credit\$ 55  
 =Time Ver\$ 55  
 =X Tile 51  
 =Y Tile 51  
 Brik Bank 50  
 List Tile 51  
 Map An Freeze 53  
 Map An Move 53  
 Map An Unfreeze 53  
 Map Anim 52  
 Map Anim Bank 53  
 Map Anim Off 53  
 Map Anim On 53  
 Map Bank 50  
 Map Bottom 48  
 Map Brik 50  
 Map Do 48  
 Map Fall 57  
 Map Handle 56

## TOME Commands (Cont.)

Map Handle Init	56
Map Left	48
Map Paste	49
Map Plot	49
Map Right	48
Map Set Zone	60
Map Swap Tile	57
Map Top	48
Map Update	52
Map Update Off	52
Map Update On	52
Map View	48
Map Zone Bank	59
Paste Brik	50
Tile Size	48
Tile Tag Set	58
Tile Tags Off	59
Tile Tags On	59
Tile Type Bank	55
Tiny Bank	57
Tiny Map	56
TOME Commands Quick Reference	78

**U**

Using Map Handle	12
------------------	----



# TOME COMMANDS QUICK REFERENCE

Tile Size x,y	=Y Tile(py)	=Map HX(x)
Map View x1,y1 to x2,y2	=Map Pos X(px)	=Map HY(y)
Map Do x,y	=Map Pos Y(py)	=Map FX(x)
Map Left x,y	List Tile	=Map FY(y)
Map Right x,y	Map Update On	Tile Type Bank b
Map Top x,y	Map Update Off	=Tme Credit\$
Map Bottom x,y	Map Update x,y	=Tme Ver\$
Map Paste x,y,px1,py1 to px2,py2	Map Anim n , x , y , r , d , anim\$	=Map Base
Map Handle x,y	Map Anim On	=Map Check
Map Handle Init	Map Anim Off	Tiny Map x,y,z
Map Plot t,x,y	Map Anim Bank bank, n,a	Tiny Bank n
=Map Tile(x,y)	Map An Freeze n	Map Fall tile
=Tile Val(x,y,n)	Map An Unfreeze n	Map Swap Tile t1,t2
=Map X	=Map An Point(n)	Tile Tag Set tile,n
=Map Y	Map An Move n,x,y	=Tile Tag
Map Bank b	=Map An At(x,y)	=Tile Tag X(n)
Map Brik n,x,y	=Map Ab Length(n,a)	=Tile Tag Y(n)
Paste Brik n,x,y	= Map Length( x , y )	Tile Tags On
=Briks	=Map Scan X(t,x1,y1 to x2,y2, mode)	Tile Tags Off
=Brik X(n)	=Map Scan Y(t,x1,y1 to x2,y2, mode)	Map Zone Bank bank,n
=Brik Y(n)	=Tile Count(t)	=Map Zb Length(n)
Brik Bank b		Map Set Zone z,x1,y1 To x2,y2
=X Tile(px)		=Map Zone(x,y)