AMOS professional Compiler

User Guide

europress SOFTWARE

AMOS professional Compiler

by

François Lionet

Programming and design	François Lionet
Shell design and programming	Jean-Baptiste Bolcato
Examples by	Jean-Baptiste/Syntex/François
User Guide	Stephen Hill
Product Manager	Richard Vanner

Write to Europress Software for help with defective discs or other initial problems: Customer Services, Europress Software Ltd, Europa House, Adlington Park, Macclesfield, SK10 4NP.

Copyright conditions

Once a program has been compiled using the AMOS Professional Compiler, it cannot be distributed unless the following conditions are met:

Public Domain/Shareware/Licenceware

If the compiled program is to be released in one of the above categories, the software must state to the user that the software was written in AMOS. The AMOS sprite (supplied on the AMOS Pro Compiler Extras disk) must be used for this purpose.

We have supplied a simple procedure which is ideal for meeting the above condition. Just call it at the start of your program. The procedure can be found on the AMOS Pro Compiler Extras disc and is called "AMOS_Copyright.AMOS".

Commercial releases

A number of AMOS programmers have written to let us know that certain software publishers are not keen to release products written using what they call 'game creators'. To get around this problem, we have decided to change the copyright condition for commercial developers.

Compiled programs that are to be used in a full commercial release do not have to carry the AMOS logo in either packaging or software, providing you let us know in writing that your AMOS game is to be released commercially, and that you send a copy of the finished program on or before its release date.

Europress Software reserve the right to publish the fact that the program was written in AMOS, two months after its release.

Foreword

Welcome to the AMOS Pro Compiler!

In this box you will find a magic utility that will enhance the overall speed of your programs by up to two or three times!

Not only will this utility make your programs run like the wind, it will also entirely recode them so that it is virtually impossible to see the original AMOS source code. You can now turn your programming creations into Public Domain, Shareware or even Commercial software products.

We have also crammed a bunch of examples on to the disc, including a game, utilities and demos. These demonstrate the benefits of compiling your AMOS programs. Just run them through your Editor first, then compile them and see the difference in speed! Then have fun exploring the code: we have inserted as many comments as possible.

AMOS Pro users will also be pleased to read that this box also includes a new version of AMOS Pro (V 2.0). It offers major enhancements, so check out the extra on-line help information in 2.0 for full details of what's new.

Both AMOS Pro and the Compiler have had a considerable rewrite. The resulting software is now much easier for us to modify and amend. The obvious advantage is for you, the user. AMOS Pro and its Compiler can now evolve at a faster rate. You may see in the future new instructions, new modes and other add-ons. AMOS Pro is a fast moving product and we want to have it moving even faster, at the speed of our beloved machine, the Amiga!

Now insert the AMOS Pro Compiler disc into your drive, fasten your seat belts, and get ready for speed!

François

Contents

1: Quick Start

Congratulations on buying the new AMOS Professional Compiler. It's the indispensable companion for all AMOS, Easy AMOS, and AMOS Professional users. Amongst it's exciting new features, you'll find:

- An intelligent memory management system which automatically makes the best use of all the available resources on your machine.
- The ability to create smaller and more optimised compiled files. You can now pack dozens of compiled programs onto a single 3.5" inch disk!
- A fantastic new Shell utility which brings the Compiler to life! All features can be accessed with a simple, easy to use, control panel. Help is available at any time, using a powerful interactive help system. So you'll never be more than a second away from assistance!
- A new batch mode which can compile an entire directories worth of AMOS programs in a single operation!
- The BOOT_DISK_MAKER utility, which effortlessly installs any compiled program onto its own start-up disk, ready for immediate use!

Once you've discovered the AMOS Pro Compiler, you'll never look back. It opens up a whole new dimension in AMOS programming.

AMOS Pro Users read this

If you're an AMOS Professional user, you're able to reap the maximum benefits from the new Compiler. The Shell is available using a simple option from the "User" menu and you're able to compile and run your programs directly from an editor window.

Before you can access these features you'll need to upgrade to the latest AMOS Pro 2 system! This is an essential first step because the Compiler won't work with any of the previous versions of AMOS Pro (including 1.12).

You can find more information on the installation procedure in Chapter 2. It's a simple matter of booting up the updater and inserting your working copies of the AMOS Professional master disks when requested.

After you've updated to version 2, you'll need to install the actual Compiler. This will require you to boot up a copy of the "AMOSPro_Compiler:" disk and follow the on-screen prompts. The installer saves the Compiler Shell onto your AMOS Pro start-up disk and creates new options in the "User" menu. From then on, the Compiler will be available directly from the editor, whenever you load AMOS Professional! See Chapter 3 for the exact installation procedure.

You'll then be ready to get into some serious compiling. If you're a beginner, you should jump immediately to the Compiler tutorial in Chapter 4. Expert users may prefer to study chapters six and seven, where you'll find a complete explanation of the various compilation features.

You should also check out the "Compiler_Examples" folder on the "AMOSPro_Extras:" disk. This contains a host of exciting demo programs which can be used for your compilation experiments. Happy compiling!

Easy AMOS and AMOS Compiler Users

If you've yet to upgrade to AMOS Professional, you'll find the AMOS Pro Compiler invaluable, as it will provide a massive boost to your existing capabilities.

The new package can compile programs written in Easy AMOS, AMOS or AMOS Professional. And it produces results which are vastly superior to the original AMOS Compiler!

You can access the Compiler from either the Workbench or by simply booting the "AMOSPro_Compiler:" disk from the internal drive. In general, you'll be able to call up exactly the same options as dedicated AMOS Pro users. The only limitation, is that you won't be able compile or test programs from your AMOS editor.

Although this is an obvious drawback, it does provide the maximum amount of memory for the Compiler to work. So the compilation will be fast, even on the smallest Amiga.

Before using the Compiler, you'll need to install it carefully on your current system. So jump to the installation section in Chapter 3 for instructions on how to do this. The basic procedure is easy enough, you simply boot-up a copy of the "AMOSPro_Compiler:" disk and follow the on-screen instructions. A little later, you'll be up and running.

Once you've installed the package, you can call up the Shell by just booting your new copy of the "AMOSPro_Compiler:" disk or by clicking on the "COMPILER_SHELL" icon from the Workbench.

You can then progress to either the tutorial in Chapter 4, or to a thorough explanation of the various Compiler features in Chapter 6.

We've supplied several demo programs to accompany this package. These can be found in the "Compiler_Examples" folder on the "AMOSPro_Extras:" disk.

If you're an Easy AMOS user, don't be surprised if a few of the demos won't run from the Editor. In order to generate the best possible screen effects, we've occasionally used features which are only available from the complete AMOS system. But never fear - you'll still be able to compile and run these programs from the Workbench with no problems. Just copy the examples onto a fresh disk and follow the tutorial in Chapter 4. Ok? You can tell if a compiler example is Easy AMOS compatible by checking it's filename has "_EZ.AMOS" at the end.

2: Updating AMOS Pro to Version 2.x

If you're an AMOS Professional user you'll need to upgrade immediately to version 2.x using the updater utility on the AMOS Professional Updater disk. The new version is the latest available, this is why we're labelling it with a ".x" instead of .1, .2 etc.

Easy AMOS and AMOS users should proceed directly to the Compiler installation section in Chapter 3.

AMOS Professional version 2.x is a major enhancement over the original AMOS Professional package. It integrates the AMOS Pro Compiler with the Editor, adds many additional features and fixes a number of problems reported in earlier versions.

How to Update

- Gather together working copies of your six AMOS Professional disks. **Don`t** use the original masters - these should be kept in a safe place, just in case the update fails! The updater will also update an AMOS Professional stored on a hard disk.

- Insert the AMOS Professional Updater disk into the internal drive and boot up your Amiga. If you have a hard disk then boot your system first and then insert the updater disk.

- When the Workbench screen appears double click on the "AMOSPro_Update" disk icon to display its contents to the screen.

- You can now either:

  1 Double click on the "About_AMOS_Pro" icon for more information about the updating process and the files which are affected by the update. To view more of the text file click to the bottom of the screen. Stopping the display from scrolling is done by clicking with the right mouse button. To quit the text reader simply press the Escape key.

  2 Click twice on the "AMOS Pro_Update" icon to install AMOS Pro 2. The updater will load and run. Just follow the on-screen prompts, inserting the various system disks as and when they are requested for.

- If the update fails you'll need to restart the update process on new disks.

- Contact Europress Software if the procedure completely fails - this will probably be due to one of the AMOS Pro master discs or the Updater disc being corrupted.

3: Installing the Compiler

The AMOS Pro Compiler automatically configures itself to make the best possible use of the available system resources.

The installation process is really easy, and providing you follow the instructions, you should be up and running in a matter of minutes. However, since this procedure makes permanent changes to the original system files, you should always install the Compiler on a **copy** of the supplied disks.

Before we get stuck into the main installation section, we'll say a few words about the way the Compiler will be set up for the various types of users.

Easy AMOS and AMOS Users

The AMOS Pro Compiler disk will give excellent results on your system. The only snag being the inability to compile or run your programs from your AMOS Editor.

The installer sets up the Shell permanently on the "AMOSPro_Compiler" disk.

You can call up the Compiler in three ways:

1. Boot the newly installed "AMOSPro_Compiler:" disk from the internal drive. This makes the maximum use of the available memory and is recommended if you don't have a hard disk.

2. Click on the "COMPILER_SHELL" icon from the Workbench.

3. Call the APCmp command from the CLI. Strictly for CLI enthusiasts!

AMOS Pro Users

AMOS Pro users will be able to access all the compilation features straight from the Editor. The installation program will take your updated AMOS Pro system disk and add three new menu items to the "User" menu: "Compiler Shell", "Compile" and "Set Compiler".

You can also run the Compiler from AMOS Pro's Direct mode, using the new, improved COMPILE instruction.

AMOS Pro can compile your programs into CLI, Workbench or ".AMOS" formats. ".AMOS" programs are special, as they can be loaded into a window and run directly from the editor. You can even compile your favourite AMOS Pro accessories in this way! Fancy a faster Object Editor or an instant help system? It's easy with the AMOS Pro Compiler.

If you'd prefer to run the Compiler from outside AMOS Professional you can also use the same options as AMOS or EASY AMOS users. So you can either:

1. Boot the newly installed "AMOSPro_Compiler:" disk from the internal drive. This consumes the minimum amount of memory and could be quite useful if you're compiling very large programs.

2. Click on the "COMPILER_SHELL" icon from the Workbench.

3. Enter the APCmp command from the CLI. Perfect if you're compiling Ascii programs from a text editor.

Using a Work disk

The "AMOSPro_Compiler:" disk is crammed full of exciting features and requires an entire 3.5 inch disk to itself. So if you don't have a hard drive, you'll need to store all your compiled programs on a separate work disk.

In practice, it's a good idea to use several disks for this purpose, and switch between them at regular intervals. This avoids the risk of losing vital programs due to a single, unexpected disk error. After all, blank disks are cheap, but your own work is priceless!

Different systems

When you install the Compiler the system will ask you a few simple questions about your configuration. The installation program takes the following factors into consideration:

- **Your type of AMOS:** If you're an AMOS Pro user, a copy of the Compiler Shell will be saved onto your AMOS Pro boot disk. You'll now be able to call it straight from the Editor without having to swap disks.

- **Are you installing onto a hard disk?:** AMOS Pro works quite differently on floppy and hard disk based systems.

Once the installer has this information, it can configure the Compiler to make the best use out of your resources. The precise results will depend on your actual system. Here's a quick run down of the various possibilities:

Single floppy systems (no hard disk)

If you've just a single floppy, the Shell will automatically read in the various Compiler libraries and copy them into a super fast ram disk.

You'll now be able to remove the Compiler disk from the internal drive and compile your AMOS programs directly from your Work disk. Since the AMOS Pro Compiler generates much smaller files than the original, your disk space will naturally stretch a lot further.

Twin drive floppy systems (no hard disk)

The stand-alone Compiler will work as with a single floppy. It will automatically copy all the necessary library files into the ram disk before use. You'll now be able to place your work disk into any drive and Compile away.

If you're running the Compiler from the AMOS Pro Editor, things will be slightly different. The Shell will only copy the libraries into the Ram disk if you've more than 1 Meg of memory. If not, the Shell will read the library files directly from the Compiler disk during compilation. So you'll need to keep the Compiler disk in one drive and the work disk in another. This will free up the maximum space for your AMOS programs, but it will also significantly slow down the compilation process. If you're compiling a batch of programs, you may find it quicker to leave AMOS Professional and call the Compiler from the Workbench instead.

Hard disk users

If you've a hard disk, the AMOS Pro Compiler will perform exceptionally well. You'll get superb results with either AMOS, EASY AMOS or AMOS Pro. All the various Compiler files will be safely installed in their own "Compiler" folder and you'll be able to access the Shell at any time from the Workbench.

The Compiler libraries can be kept on the hard disk or loaded into ram disk as required. Because of the speed of your hard disk, there's not a great deal of difference! Most operations will be practically instant!

How to install

AMOS Pro users should immediately check that they've upgraded to AMOS Pro 2. This is an essential first step, as the Compiler won't work with anything else.

If you don't have a hard disk, you can now simply boot up a copy of your original Compiler disk and follow the simple on screen instructions. A few minutes later, the Compiler will be installed!

Hard disk users should run the separate "COMPILER_INSTALLER" program from the "AMOSPro_Compiler:" disk. After you've provided your configuration details the entire package will be copied into a new "Compiler" folder alongside your other AMOSPro folders, ready for immediate use.

4: Compiler tutorial

The AMOS Pro compiler provides you with everything you need to create professional programs on the Amiga.

We'll show you how it works with a simple tutorial. Our mission, will be to load up the Compiler Shell and compile one of the demos from the AMOS Pro Extras disk. For an encore, we'll take our compiled program and install it onto its very own boot disk.

The AMOS Pro Compiler has been carefully designed to give excellent results on anything from a humble A500+ to the very latest A4000 systems. There's obviously no way we can cater for every possible configuration in the space of a few pages, so we'll keep this tutorial fairy general. When we refer to the AMOS Editor we mean the AMOS Editor you are using – the Easy AMOS one, the AMOS Editor or AMOS Professional Editor.

If you've extra memory or a hard disk you'll often be able to perform many of these activities using a faster, more efficient approach. However, once you've familiarised yourself with the basic techniques, you'll be in the perfect position to read up on these additional features.

1 Preparing for the tutorial

Before you proceed any further, you'll need to do a little preparation. The AMOS Pro Compiler can't be run straight out of the box. It needs to be installed carefully on your current system.

- If you're an AMOS Pro user, you'll need to immediately update your copy of AMOS Pro to the latest version 2.x system. This applies even if you've already upgraded to AMOS Pro 1.12! As always, you're recommended to perform all upgrades on a copy of your master disks. It's a vital safeguard against mistakes.

- You should now install the compiler onto your Amiga. Detailed instructions can be found in Chapter 3. This procedure should be performed by all users, as it sets up the compiler for your particular system.

- After you've successfully installed the compiler, you should prepare a couple of fresh 3.5" disks for the tutorial.

  The first disk should be formatted and labelled as "WORK_DISK". It will be used to hold a working copy of your compiled programs.

  The second disk will be required by the "BOOT_DISK_MAKER" utility to generate a bootable copy of the compiled example program. See later.

Now we've created our disks, let's get stuck into our tutorial.

2 Selecting a program to Compile

In order to keep things simple, we'll perform all operations from the Workbench. Start off by inserting your normal "Workbench" disk into the internal drive and rebooting your Amiga.

- When the Workbench screen appears, place your copy of AMOS, EASY AMOS or AMOS Pro into an available drive, and select the disk icon to display its contents on the screen. You should then load your AMOS editor by double clicking on the appropriate icon.

- Insert a copy of the AMOS Pro Extras disk and call up the Load option from your editor's main menu. From the file list in the file selector click on the "Compiler_Examples" folder to list the available demo programs. The example we are currently interested in is called "FRACTAL3_EZ.AMOS". So select it with the mouse and load it into the editor.

- We'll now run this program on the screen. It's an ingenious fractal generator, created by Jean-Baptiste Bolcato. Just hit F1, and watch it go! While you're at it, don't forget to jot down the timings on a scrap of paper. This will come in handy later, as you'll be able to compare the speed of the interpreted version to the compiled version.
  Once you've had a play, insert your "WORK DISK" and save the Fractal program as a new file to this disk. Use the SAVE As option from the menu.

- You should now leave AMOS and return to the Workbench. It's time to run the compiler!

3 Running the Compiler

- Place the newly installed compiler disk into any drive and open up the directory window as before. A large icon labelled Compiler_Shell will be displayed on the screen. This forms the gateway to a simple control panel which handles all aspects of the compilation process. It can be loaded by clicking twice on its icon. Here's a quick run through of the various buttons:

  Source chooses the files you wish to compile.

  Destination enters where the compiled program will be saved once compiled. It's currently set permanently to DISK, but AMOS Pro users will also be able to compile a program directly into a new Editor Window.

  Type chooses the type of the final program. Depending on your selection the compiled program can be run from either the CLI, the Workbench or the AMOS Pro editor.

  Compile begins the compilation process and prompts you for the source and destination files.

  Help activates a powerful on-line help system. When it's clicked on you can view information on any item by simply clicking on a help active zone with the mouse. You can turn the help off by just reselecting the Help icon again.

  And finally, there's the **Flag** icon which allows you to quit the Compiler Shell.

- For the moment, the SOURCE and DESTINATION should be set to "Disk" and the TYPE should be "W.B" Exec". If the buttons are currently displaying something else, you may need to click on them a few times to set them correctly. Well, that's the control panel. Let's put it through its paces.

4 Compiling the program

- First, click on Compile to begin the compilation. You'll immediately be presented with a standard file selector. This should be used to choose your AMOS source program. Insert the new "WORK_DISK" disk, click on Get Dir and select Fractal3_EZ.AMOS when the directory is displayed.

- You'll now be asked to enter a destination file for your compiled program. The name can be typed in either from the selector or generated automatically by the system. Choose "OK" to use the default system which will create a file called Fractal3_EZ.

- Your program will now be compiled in front of your eyes. As the compilation progresses, a horizontal bar will expand across the screen. If there are no problems you'll then be presented with a small dialogue box containing a few statistics. Click on "Continue" to return to the main compiler screen.

5 Running the compiled program from the Workbench

- Now that we've compiled our example we'll test it from the Workbench. So we'll wave a fond farewell to the control panel and exit by selecting the Flag icon to the left of the screen.

- After a few seconds the Workbench will reappear. Open the "WORK_DISK" disk on the screen and click on the newly created "FRACTAL3_EZ" icon. The compiled program will be executed straightaway. As you can see, it's much faster than the interpreted version!

- If you're a CLI enthusiast, you can also run the same program from the command line. Replace your Workbench disk and display its contents on the screen. Now click on the "Shell" icon to bring up the CLI Shell.

- Insert "WORK DISK" and select it as the current drive. If you've only a single drive use:

```
1.SYS>cd DF0:
```

Alternatively, if your example disk is available from the second drive use a line such as:

```
1.SYS>cd Df1:
```

You can now run your compiled program by simply typing its name at the prompt:

```
1.Empty>Fractal3_EZ
```

- Finally, we'll round off this section by installing the fractal program onto its own individual boot disk.

6 Creating a boot-disk

In the original AMOS Compiler creating a boot-disk was a real chore, as you had to copy all the library files by hand. So you'll be pleased to hear that we've automated the procedure completely. Here's the procedure:

- Insert a copy of the AMOS Professional Extras disk into any drive and select its icon with the mouse. When the file window appears click twice on the "BOOT_DISK_MAKER" icon.

- You'll now be asked to enter the name of the compiled program to be installed on the disk. If you've been reading the previous tutorial, you should already have this ready. Place the "WORK DISK" in an available drive, and choose "FRACTAL3_EZ" at the file selector. This file will now be loaded straight into memory ready for the installation.

- You'll then be prompted for a blank disk for use as your boot disk. So insert it into any drive and click on "OK". Make sure it doesn't contain anything useful, as it's destined for immediate formatting.

  After the boot disk has been prepared you'll be asked for the "AMOSPro_Compiler:" disk. The program will now copy the various system files onto your new disk, and set-up the required start-up sequence in the "S:" directory.

  If you have two drives, you'll be able to keep both the boot disk and the compiler disk

in separate drives. If not, you may have to do a bit of disk swapping.
Once the process is complete, you'll be returned directly to the Workbench.

The new disk can now be booted straight from the internal drive.

Note: If you want to generate really professional results, its best to start off your compiled program with a totally clean display. This can be accomplished by selecting the "NODEFAULT" and "WB" options before compilation. Check out the "Compiled Program Setup" menu from Chapter 6.

And that's it! You've just compiled your first program. The next step is to have a bash at compiling your own AMOS programs.

Expect to be amazed!

5: The APCmp Utility

What is APCmp?

Well, the AMOS Pro Compiler is really two separate programs:

1 The Compiler Shell provides a fancy control panel which can be accessed from either the AMOS Editor or the Workbench. Amazingly, it was written entirely in AMOS Pro! All those attractive icons and powerful menu options were generated using the built-in AMOS Interface commands! So if you've yet to upgrade to AMOS Professional, it's another compelling reason to take the plunge.

2 The actual Compiler's core code is held in a small module called APCmp (**A**mos **P**rofessional **Comp**iler). It's a 68000 program which converts your AMOS or Ascii files into ultra fast machine code. APCmp is normally loaded invisibly by the Shell, but it can also be called directly from the CLI.

APCmp is an extremely clever program which automatically configures itself to your available memory. It doesn't matter a jot if you've got a simple A500 or the latest A4000 dream machine, you'll still get terrific results from this package!

Tokenising

This is a technical term for the process of converting a list of Basic instructions into a form which can be executed directly by an AMOS Editor.

Generally the AMOS Editor tokenises your programs as you are typing them. You can see the effect whenever you move to a new line. As AMOS reads through your text, any variables and instructions are picked out and displayed in their own distinctive style.

So:

```
print "This is a test":a=b+5:gosub code
```

becomes:

```
Print "This is a test" : A=B+5 : Gosub CODE
```

The AMOS Pro Compiler can handle either tokenised AMOS programs or raw Ascii text. It works by checking the first few characters of your intended source programs and seeing if they make sense. If it recognises an AMOS program it can compile it immediately. But if it encounters an Ascii program it will load up a tokeniser from "Compiler.Lib" and automatically generate an appropriate ".AMOS" file for use with the Compiler.

The tokenisation is usually performed directly onto the disk. So the resulting ".AMOS" program file will be saved in the same folder as the original Ascii source file. If you want to speed things up though, you can store all temporary files into a ram disk. This can be selected using either the TEMP directive from the command line or by changing the default Compiler options from the Shell. The line you'll need is:

```
APCmp File.Asc TEMP="Ram:"
```

The destination file is generated directly from the source name. The Compiler removes any characters after the dot and just adds ".AMOS" to the result. So "File.Asc" would be saved as "File.AMOS"

Once the file has been successfully tokenised the new ".AMOS" file is tested and entered into the Compiler. The temporary ".AMOS" file is then deleted from disk automatically.

If an error occurs during the tokenisation you'll get an appropriate message on the screen:

```
Line too long at line XXX
```

Tokenising files only

The AMOS Pro Compiler can also be used as a simple tokeniser.

```
APCmp Myprog.Asc TOKEN
```

The TOKEN option generates a new ".AMOS" file called "Myprog.AMOS" which can be subsequently loaded directly into your AMOS Editor. The Compiler then stops in its tracks.

This type of tokenisation is much faster than the MERGE Ascii command from the Editor. It's therefore ideal for converting large numbers of source files into ".AMOS" format.

Testing

The AMOS Pro Compiler only works with tested programs. So if it encounters a file which hasn't been checked, it will need to test it first. This applies to both native ".AMOS" programs and Ascii files which have been tokenised by the Compiler.

If it does require testing then APCmp will load a series of routines from "Compiler.Lib" and check the program automatically. The program will then be entered straight into memory and tested just as if you'd tested it from the Editor.

If an error occurs it will be reported as normal using the error messages from the "S:AMOS Pro_Interpreter_Config" file. The Compiler will then abort the compilation.

The AMOS.Library

AMOS Pro users must have a copy of this library installed in the "LIBS:" folder of their current start up disk. It contains all those vital screen display routines which make AMOS programs so special.

The original AMOS Compiler saved a separate copy of these routines with each compiled program. This resulted in a 45K increase in the final program size of compiled programs.

APCmp avoids this overhead by sharing a single library file between all your compiled programs. The only snag is that you'll need to keep a permanent copy of the "AMOS.Library" on your boot disk.

However, if you want to create truly independent programs, you can force the Compiler to revert to the original scheme. Just add the INCLIB directive to the end of your command line like so:

```
APCmp MyProg.AMOS INCLIB
```

If you're using the Compiler Shell, you can achieve the same effect by selecting the "Include AMOS.Library in compiled program?" option from the "Compiled Program Setup" menu. Each program will now include its own individual copy of the AMOS.Library.

The Compiler Configuration

The Compiler stores its various configuration settings in the "S:" folder of your current boot disk. Unlike the old AMOS system, these files are not in text format and can't be modified with a standard text editor. They should only be changed using the dedicated "Setup" option from the Compiler Shell.

Internal buffers

In the original AMOS Compiler, it was occasionally necessary to increase the Compiler buffers in order to compile very large AMOS programs. This process is now completely automatic.

6: The Compiler Shell

Welcome to the new, improved, AMOS Professional Compiler Shell. This provides you with total control over the compilation process using a series of simple, easy to use screen options. The AMOS Pro Compiler is packed with exciting new features. These include:

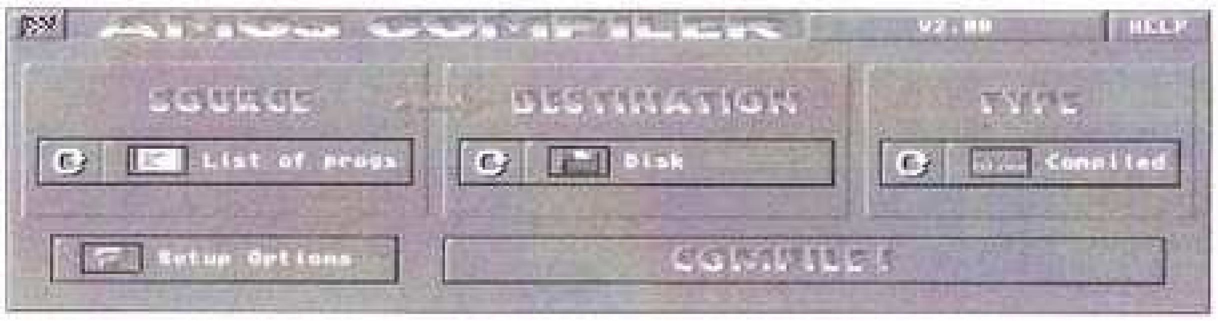
- Integrated on-line help. So if you get stuck, help is only a key-press away.
- AMOS Professional users can compile a program directly from the currently active editor window.
- Automatic testing. (No more "program not tested" errors!)
- A new batch mode which can compile several programs in a single operation.
- Entertainment! You can now play your own selection of music and watch your favourite IFF animation files while your programs are compiled.

The Shell can be called up in two main ways:

1 If you're an AMOS or EASY AMOS user, you can only load the Compiler by either clicking on the "Compiler_Shell" icon from the Workbench or by booting directly into the Shell by inserting the AMOS Pro Compiler disk into the internal drive and starting up your Amiga.

2 If you're an AMOS Professional user, you're able to enter the Shell as above or from within AMOS Pro's editor by using a powerful new "Compiler Shell" option from the "User" menu.

Either way, you'll be presented with the following screen:



Those of you who have used the old Compiler might be thinking that this new Shell looks like a simple update to the original. But don't be misled by the apparent similarities. Underneath the surface, it's been completely transformed! Many options are totally new and others have been changed beyond recognition.

Take the "SOURCE" and "DESTINATION" icons for example. These buttons originally controlled the memory system and provided you with the choice between either loading the entire program before compilation, or reading the code piecemeal off the disk. However, with the introduction of the AMOS Pro Compiler, all this memory is allocated automatically. So the effects of these buttons are totally different. The "SOURCE" icon just enters the location of your AMOS source programs and the "DESTINATION" button sets the position of the resulting code. They've nothing to do with memory at all!

The AMOS Pro Help system

The Shell includes a built-in help system which supplies you with practical advice about all aspects of the Compiler Shell.

Help can be accessed at any time using the "Help" icon at the top right of the control panel. When it's selected, the mouse pointer will change shape to show that you've entered help mode. You can return to normal by choosing the "Help" button again.

Once you've entered this help mode you can get instant details about any button or menu item by just selecting it with the left mouse button. As it's moved over the screen the pointer will animate when it enters an area which has Help available for viewing.

The help will be presented in its own attractive text window. This screen should be familiar to all AMOS Pro users, as it's identical to the standard AMOS Pro help window. However, if you're a die hard Easy AMOS or AMOS 1.3x fan, you may appreciate a few simple directions:

- Menu items are displayed in REVERSE text colours. These "hypertext" options can be selected with the mouse to move directly to a new area of help.
- The "X" icon at the top left corner is an EXIT button. This quits help and returns you to the main Compiler Shell. You can now leave help mode with an additional click on the "Help" icon.
- If the current text is too large to fit in a single window, you can scroll through the information by either using the "scroll bar" or pressing the UP and DOWN arrow keys from the keyboard.
- "Prev.Page" moves you back to the most recently displayed help page.
- "Main Menu" jumps to a central help menu. Be sure to check out the "Latest News!" option. This contains all the latest info on the AMOS Pro Compiler.
- "Print" outputs the current help page onto your printer. Check that it's connected first!

Note: The help system consumes around 150K of memory. So if you're running the package from inside AMOS Pro you may have to close a few windows before use. This will normally free up enough space for the system to work.

Using the Compiler

To control the Compiler simply click on the large screen icons with the left mouse button. Here's a list of the available options:

SOURCE

Chooses the program to be compiled. The possibilities include:

Current Prog: compiles a program from the current editor window. This feature is only available if the Shell has been called up from the AMOS Pro "User" menu. Since it requires a hefty amount of memory it won't be selectable if you're running short of space.

Disk: reads your program from the disk in either ".AMOS" or "Ascii" format. You'll be prompted for the filename when you call the Compiler.

List of progs: compiles a batch of programs in one smooth operation. See "Compiling several programs" later in this chapter.

DESTINATION

Enters a destination for the compiled program. The two states in can be in are:

Editor Window: can only be used if the "Current Prog" option has been selected as the Source. It places the compiled program into its own individual editor window ready for immediate running.

Disk: is the default. It compiles your program into a brand new file on the disk.

TYPE
Selects the type of the compiled program. There are three possible settings:

W.B exec: These compiled programs can be run by just clicking on their icons from the Workbench screen. The Compiler will automatically save an appropriate icon to accompany your program. You can redraw it using the IconEd utility from the "Tools" folder of your Workbench disk.

CLI.Exec: Programs compiled in this way can be executed from the command line by simply typing their names. Apart from not having a ".info" icon file, these programs are identical to the Workbench versions.

AMOS Compiled: You can load and run these programs from AMOS Professional, just like a normal Basic program. But they can't be loaded and run from AMOS 1.3x or Easy AMOS.

COMPILE
This icon compiles your programs using the current settings.

If a program is being compiled from disk you'll be requested to locate the file you want compiling using a file requester which pops up as soon as you click on this icon. If the resulting compiled file is to be saved onto disk, then you'll also be prompted for the destination file with another file selector.

You can choose a default destination name by simply clicking on the file selector's "OK" button. An appropriate filename will be automatically generated for your program, using a variation of the original source name. "AMOS Compiled" programs will have "_C" inserted before the ".AMOS" bit. So "Test.AMOS" will be saved as "Test_C.AMOS".

CLI or Workbench programs will have the ".AMOS" part removed. "Test.AMOS" would now be compiled as just "Test".

Warning! The automatic naming system will only work if you've already saved your source program on the disk. If you're compiling from an unnamed program window, you'll need to enter a specific destination file into the file selector.

Also note that if you're compiling a list of files, the filenames will be computed automatically by AMOS Pro. The compiled programs will then be saved in the current directory, alongside their original source files.

After you've entered the filename(s), the Compiler can get to work. You can see how the compilation is progressing from the bar in the centre of the screen. As the program is compiled, this will slowly expand to the right. If that seems rather boring, you can also play an IFF animation or listen to a little music as your program is compiling. See the section on "Compiler Shell Setup" later in this chapter for more details.

If there's a problem, the Compiler will normally display the error in a small dialogue box. However, if you've compiled your program from an AMOS Pro window, you'll be returned straight to the editor with the cursor positioned over the offending line! Either way, the compilation can be aborted by pressing Control-C.

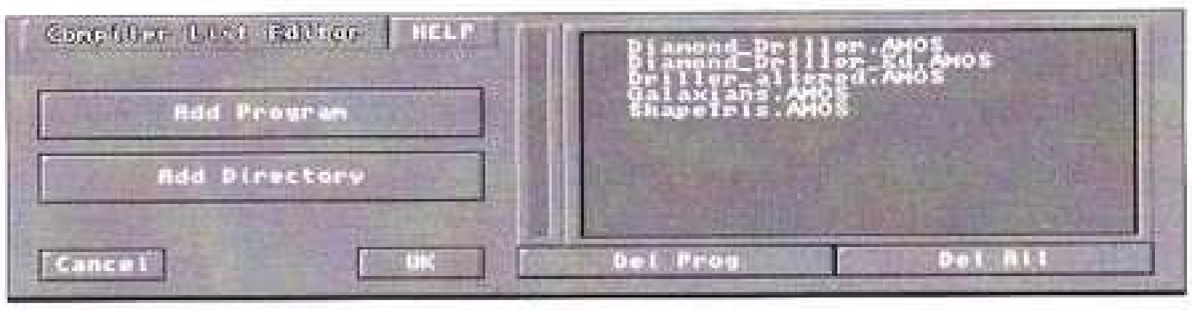
Once your program has been successfully compiled you'll be presented with an information window containing a few statistics. You can return to the control panel by selecting "OK".

"Quit" exits the Compiler Shell and returns you to either the Workbench or the AMOS Pro Editor as appropriate.

Compiling a list of programs

The Shell is just not limited to compiling a single program, it can actually compile a whole batch of programs at a time. You can access this feature by choosing the "List of progs" option as the "SOURCE".

When you click on "COMPILE", you'll now be presented with the Program List Editor like so:



```
Diamond_Driller.AMOS
Diamond_Driller_Ed.AMOS
Driller_altered.AMOS
Galaxians.AMOS
ShapeTris.AMOS
```

HELP
This toggles the help system on and off. If it's highlighted you can get help on any part of the list editor by just clicking on the various screen options.

Add Program
This button appends a single file to the compilation window. It brings up a standard AMOS Pro file selector which is used to add a new program into the list.

Add Directory
Searches a directory for ".AMOS" files and copies their names directly into the compilation list. As before, you'll be presented with a file selector for selecting the desired directory. To actually select a folder move to the required directory and double click on any AMOS file within the folder - all the ".AMOS" files will be selected for compilation and their names will be listed in the compilation list window.

Del Prog
Deletes a program from the compilation list. Just select a program in the list window and press the "Del Prog" button. If the required file is not on display you can scan through the list using the large scroll bar to the left of the list window.

Del All
Removes all the entries from the compilation list leaving you with an empty window.

Cancel
Ceases the compilation and returns you to the main control panel.

Ok
Begins compiling the selected programs. The Compiler will now read each program in turn and save the compiled file to the disk using the default filename. If there's an error you will be presented with a requester that allows you to stop the whole compilation batch or continue. Any programs that create errors will be left uncompiled.

You can abort from the compilation run at any time by simply hitting the "Esc" key from the keyboard.

The Compiler Configuration

"Setup Options" forms the gateway to a large configuration menu. This allows you to fine tune the Compiler to your own particular needs.

Whenever the Compiler is run, the current settings are automatically read from the "AMOSPro_Compiler_Config" file. This file is first looked for by the Compiler in the current directory, if this search fails the Compiler looks to the path "S/". Another failure will have the

Compiler looking into the "S:" folder for the file. The configurations can be modified using a series of menus accessable from the Options menu.

The Options menu

Compiled Program setup: changes the way your programs will be compiled onto the disk.

Compiler Shell setup: alters the operation of the Compiler Shell.

Compiler System setup: sets the default command line options and saves the positions of the various system files. These should only be changed if you're an expert user, as a single mistake could stop your Compiler from functioning properly.

Load Config: loads the selected compiler configuration file from the disk.

Save as Default: saves a new "AMOSPro_Compiler_Config" file into the "S:" directory of your current boot disk. If your boot disk is not presently available, you'll be asked to insert it immediately. This option also saves the contents of the "SOURCE", "DESTINATION" and "TYPE" buttons on the control panel. So you can tailor the initial Compiler Shell settings to precisely your own requirements.

Save Config: stores the Compiler settings in a separate file on the disk. You can use it to create different configuration files for different jobs.

Cancel: cancels any changes you've made to the configuration and returns you to the main control panel.

Use: applies the new configuration for the duration of the current session.

Compiled Program Setup

Screen 1

Include error messages?: Includes a copy of the standard AMOS error messages along with your compiled program. "Yes" loads the messages from the "AMOSPro_Editor.Config" file in the "APSystem" folder. Whenever an error occurs in your compiled programs an appropriate message will be displayed to the screen.

"No" turns off the error messages completely. So if a problem occurs in your compiled program it will jump straight back to the Workbench, CLI or AMOS Pro.

Create default screen: Normally, a standard AMOS screen will be created for your compiled program.

"No" starts your program with a totally clean display. It's equivalent to using the NODEF option from the command line. Note your program must open a new screen before performing any output (print, fade, bob and so on). If you forget, your program will abort immediately with a "Screen not opened" error.

"Yes" spontaneously generates a default screen for your compiled program.

Send AMOS TO BACK upon booting?: "Yes" completely hides your current display, just as if you had included an AMOS TO BACK instruction at the start of your program. The screen can be flicked into view using an AMOS TO FRONT command. Easy AMOS users don't have an AMOS TO BACK command, you'll have to save your program as an Ascii file, edit in the extra command from a text editor and then Compile the Ascii listing.

CLI Program to run in the background?: This option is only important if you are compiling your program as type "CLI".
"Yes" forces the compiled program to detach itself from the CLI and run as a separate process under the Amiga's multi-tasking system. It's the same as executing your CLI program with the AmigaDos RUN command.

Next panel: Jumps to screen 2 of the "Compiled Program Setup" menu.

Cancel: replaces your changes to the previous settings.

Ok: returns to the main "Setup" menu with the settings setup as you have directed.

Screen 2

Long forward jumps (option LONG for VERY long programs)?: Forces the Compiler to use 68000 JMP instructions rather than BRAnch commands in the compiled program. It's needed to allow program structures such as IF..ELSE..ENDIF and WHILE..WEND to jump across 32k boundaries.
"Yes" should be selected if you get a "Program structure too long, compile with LONG option" message during compilation. When you recompile, everything will be fine.

Include AMOS.Library in compiled program?: This option will only take effect if you're compiling a program in CLI or Workbench format.
"Yes" instructs the Compiler to save a fresh copy of the "AMOS.Library" as part of the compiled program. As a result the compiled program will be 50K larger than normal, but it will be totally independent of the AMOS system.
"No" (default). Uses shared libraries. The "AMOS.Library" will now be loaded automatically from the "LIBS:" folder whenever your compiled program is run.

Next Panel: returns you to Screen 1 of the "Compiled Program Setup" menu.

Compiler Shell Setup

Screen 1

Copy all libraries onto RAM disk: If you've got an expanded machine, you can speed things up by copying all the compiler libraries onto the super fast RAM disk. This feature should only be used if you've plenty of memory since the libraries consume around 145k of RAM.
"Yes" instructs the Compiler Shell to copy the compiler libraries into a RAM disk during the initialisation process. If the RAM disk is not available the option will be completely ignored.
"No" deactivates the RAM disk copying.

Leave libraries on RAM disk upon exiting?: "Yes" stores the compiler libraries permanently on the RAM disk. So they'll remain in place for the duration of the current session.
"No" forces the Shell to delete the entire contents of the RAM disk when you leave, returning about 145k of memory to your system. The next time you run the Shell, it will be forced to copy all the compiler libraries back onto the RAM disk.

Keep APCmp in memory upon exiting?: "No" removes the Compiler before leaving. So when you restart the Shell, APCmp will be loaded into memory again.
"Yes" keeps the Compiler in memory for the duration of the current session.

Squash compiled program?: "Yes" will tell the Compiler to compress your compiled programs automatically before they are saved onto the disk. These files will be restored to their original size whenever they are loaded from the CLI or Workbench.

Note: This option has no effect on programs compiled as type ".AMOS".

Next panel: Jumps to Screen 2 of the "AMOS Shell Setup" menu.

Cancel: resets any changes to their previous settings.

Ok: returns to the main "Setup" menu.

Screen 2

This panel offers a number of neat animation and sound options which can be played automatically during compilation. These provide you with a little light entertainment while the Compiler is going through its paces.

However, since they consume a great deal of ram they should only be utilised if you've memory to burn. Despite the warning though, these options are still worth a look, even on the smallest Amiga. They may be a bit silly, but they're a lot of fun!

Play IFF animation when Compiling: "Yes" plays an IFF animation while your program is being compiled. This file must be an IFF animation module in mode 5 (DPAINT) format. You select the anim file using a file selector which can be called up by clicking on the small disk icon just below the Yes/No option button.

Note: If you want to use this feature, you'll need enough memory to hold the following data:

- Your animation file (FAST RAM).
- A new screen to play the animation (CHIP RAM).

So we recommend you have at least 2 megabytes of memory for this feature.

Play Tracker module when compiling: With this option, you can play a piece of music as your program compiles. Your music can be either a tracker module, an AMOS music file, an IFF sample, or a MED format tune. (The MED option will require the "MED.Library" in your "LIBS:" folder.)

Once you've activated this feature, you can change the music file to be played using the small disk icon. This will present you with a file selector which is used to choose your new music.

Warn with bell sound: "Yes" rings a bell when the compilation is complete. If you've previously assigned a MED or TRACKER file with the Music option, this feature will be ignored.

Animated buttons when under pointer: "Yes" adds an attractive animation effect when the pointer is moved over an option. Since it only consumes around 12K of fast memory it can be used on even the smallest Amiga. "No" leaves the icons static.

Next panel: returns you to Screen 1 of the "Compiler Shell Setup" menu.

Compiler system setup

Displays the "Compiler System Setup" menu. It's strictly for advanced users. So beware! This option can seriously muck up your system!

Compiler System Setup panel (Expert Users only)

CLI Compiler messages: changes the messages which are displayed when you are compiling your program from the CLI.

Default command line: sets the default command line. See Chapter 8 for a detailed list of the various options.

Compiler system files: enters the positions of the various library files used by the Compiler.

Compiler error messages: edits the error messages used by the Compiler.

These options all work in the same general way. You select an item to be changed from a standard list window and edit it on the screen using a simple dialogue box. When you're satisfied, "Ok" will enter the changes into memory, and "Cancel" will return you to the default version.

For more in-depth information use the Compiler Shell Help system - each system file's purpose is explained there.

7: Compiling from within AMOS Professional 2.x

Although the AMOS Pro Compiler works well with AMOS or EASY AMOS, it's sensational from AMOS Professional. So if you're an AMOS Professional user, you're in for a real treat.

Compiling is a cinch. You simply select the "Compile" option from the "User" menu and watch your program compile in front of your eyes. Development is effortless, even on the smallest system. And if you've a hard disk, you'll be amazed at the sheer speed of this package. Most programs will compile in a matter of seconds!

Calling the Compiler Shell from the User menu

Here's a list of the available menu options:

Compiler Shell	Runs the Compiler Shell and lets you compile your programs using a range of powerful options.
Compile	Compiles and tests the program in the currently selected window. It then places the compiled program into its very own window ready for running.
Set Compiler	Calls the Compiler Shell and enters the main setup menu. When you leave you'll be returned straight back to the AMOS Pro Editor.

Compiling from the current AMOS Pro window to another

If you've enough memory you'll be able to compile the current program directly from the "User" menu. AMOS Pro will automatically place the resulting compiled program in its own editor window. You'll now be able to run it straight from the editor, just like any other program!

Unfortunately, this feature makes heavy demands on your system resources. So if you've less than a couple of megabytes you could well run out of memory. However, there's no need to despair. You'll still be able to quickly compile even the largest programs from either the Shell or from AMOS Pro's Direct mode.

Here's how to compile from window to window:

1. Select the window containing the source program to be compiled by clicking on it and making it active.
2. Choose the "Compile" option from the user menu.
3. The Compiler will now test your program. If there's an error a requester will be displayed with the following options:

   Return to AMOS Professional Editor: This will return you to the current program. The cursor will be helpfully positioned over the offending line.

   Continue: Will return you to the Shell program. If you wish, you can now select a new program to compile.

4. After your program has been tested, it will be compiled into a temporary file on the disk. As a default, all temporary files are stored on the Ram disk, but this can be easily changed if you're short of memory.
5. Once it's been successfully compiled a new editor window will be opened up for the

resulting program. If you've compiled this program before, any previous version will be automatically replaced.
6. The temporary file will now be erased and the compiled program will be renamed so that its pathname matches that of the original source program.
7. Finally, you'll be returned to the editor with your compiled program active and ready to run.

Compiling from direct mode

The COMPILE command lets you call up all the features of the Compiler straight from Direct mode using a fraction of the normal memory.

COMPILE *(Compile a program from AMOS Professional)*

COMPILE command_string$

If the Compiler has not been previously installed, it will be immediately loaded from the "APSystem" directory. You will be asked to insert an appropriate disk if it's not available.

command_string$ contains the name of the program to be compiled along with an optional list of Compiler directives.

If it's set to "", AMOS Pro will automatically compile the current program from the editor, using the default Compiler options. The results will be saved straight onto the disk using a variation of the original filename. Example:

> Suppose you are presently editing the file "Df1:My_Program.AMOS". You could compile this file by simply entering direct mode and typing:
>
> **AMOS>Compile ""**
>
> The Compiler would now save the results in the file "Df1:My_Program"

command_string$ uses the same directives as the APCmp utility discussed in Chapter 8.

Here's a quick run-down of the more important options:

```
Compile "Program.AMOS [TO destination] [TYPE=0/1/2/3] [NODEF]
         [NOLIB/INCLIB]"
```

As you can see, the options have changed dramatically since the original AMOS Compiler. We've updated them to conform to the standard format used by AmigaDos 2.0. Another point to keep in mind is that the old "-D" options have been completely removed. They're no longer needed because the Compiler automatically makes the best use of your available memory.

Program.AMOS is the name of an AMOS program you wish to compile.

TO enters the name of an optional destination file. The default is simply the original name of your program without the ".AMOS" extension.

If your filename includes spaces you'll need to surround it with a pair of single quotes. For example:

```
Compile "'test me.amos'"
```

TYPE chooses the type of your compiled program.

TYPE=0	Creates a Workbench program with its own icon (default).

TYPE=1	Produces a normal CLI program.
TYPE=2	Generates a CLI program which runs as a detached task.
TYPE=3	Defines a ".AMOS" program. These can only be run from AMOS Pro.

NODEF stops the compiled program from creating a default screen for your graphics. If it's included, your program will start off with a totally clean display, so that you can set up your screens invisibly in the background. You will however, need to define a new screen before using any of the text, graphics, or bob commands.

NOLIB saves the compiled program using the new shared library system. Instead of including its own copy of all the AMOS system libraries each program now loads a single "AMOS.Library" file from the LIBS: directory.

But since your programs are reliant on the "AMOS.Library", they're not really independent of the AMOS System. If this is a problem, you can call up the INCLIB directive to store the libraries as part of the compiled program. This generates fully stand alone programs, just like the original AMOS Compiler. Examples:

```
AMOS> Compile "Prog.AMOS"
```

Compiles "Prog.AMOS" and stores the new version in a file called "Prog". This file can now be executed from either the CLI or Workbench as required.

```
AMOS> Compile "Prog.AMOS TYPE=3"
```

Compiles your program in ".AMOS" format. This file can be loaded straight into any AMOSPro editor window and run just like a normal program.

```
AMOS> Compile "Prog.AMOS TO Prog"
```

Creates a compiled program called "Prog" along with an appropriate Workbench icon.

See Chapter 8 for full details of the available Compiler options.

The Compiler extension commands

The new Compiler extension adds several additional instructions to the AMOS Professional system.

RUN *(Run a compiled program)*

RUN "program name"

Actually, this isn't a new instruction at all! It's just an extended version of the original AMOS RUN command.

You can use this feature to split your program into several parts which can be loaded from the disk when they are required. This technique is widely used in commercial games. Each level in the game is now assigned to a separate program on the disk and loaded separately, allowing the full game to run in low memory configurations.

RUN can be freely used in any of your compiled programs. There are only a couple of restrictions.

1. RUN can only be called from a CLI or Workbench program. So you can't chain a group of ".AMOS" files in this way.

2. If you've a limited amount of RAM, you might encounter problems due to memory fragmentation. If so, you'll be delighted to discover that we've included a powerful new memory saving feature along with the AMOS Pro Compiler:

If you get an "Out of memory" error, just place the "-Mem" statement at the start of the COMMAND LINE$, before the final RUN. This will save the maximum amount of space for your compiled program. Example:

```
Command Line$="-Mem" : Run "Myprog"
```

Whenever the RUN command encounters the keyword "-Mem" in COMMAND LINE$ it will automatically close all your screens, clear the internal memory buffers and close the "AMOS.library". The only thing remaining will be a small loader program which consume just 2K. So you'll have the maximum possible space for your compiled program. Note: As your program loads, the Workbench will be temporarily displayed on the screen.

If you need to transfer additional information in the command line, you can simply add it after the "-Mem". e.g

```
Command Line$="-Mem Hello World" : Run "MyProg"
```

When "MyProg" is executed the command line will only contain "Hello World"

=COMMAND LINE$= *(Read/Set command line parameters)*

C$=COMMAND LINE$

COMMAND LINE$ grabs the contents of the CLI command line and saves it for your compiled program. It can also be used with the AMOS RUN instruction to transfer information between several connected programs (as shown above).

=COMMAND LINE$ returns a string containing all the parameters which have been entered from the command line. If they've not been defined, you'll get an empty string ("") instead.

Note: You are recommended to check the command line as soon as possible after your program initialises, as it's held in a small buffer used by the AMOS graphic instructions. Examples:

```
1> Test one two three
```

Runs a program called "Test" and loads the string "one two three" into COMMAND LINE$.

```
COMMAND LINE$="string"
```

This sets the command line to "string". The string can include anything you like, and can be up to 256 characters long. It's often used just before the RUN command to preserve vital information between two chained programs. (compiled or interpreted). Example:

```
Command Line$=Str$(SCORE)
Run "Level2.AMOS"
```

Saves the score into the command line and then executes a new program called "Level2.AMOS" straight from the disk.

You could now read the score at the start of the "Level2.AMOS" program using a line like:

SCORE=Val(Command Line$)

COMPTEST *(Carry out compiler tests)*

```
COMPTEST
COMPTEST ON
COMPTEST OFF
```

AMOS Professional regularly carries out the following activities.

- Sprites and Bobs are redrawn at their new positions.
- Menus are checked for the presence of the mouse.
- The Control-C keys are tested.

If the timing in your program is really critical, these activities might cause an unwelcome interruption. So you may find it helpful to take control over the entire testing process yourself. That's the job of the COMPTEST commands.

COMPTEST ON

If COMPTEST is set to ON, checks will only be carried out before jump instructions such as GOTO or WHILE, and time consuming commands like PRINT or WAIT. This is the standard system used by the AMOS Pro interpreter.

COMPTEST OFF

The COMPTEST OFF command stops the testing completely and speeds your programs by up to ten percent. You can use it to optimise time critical sections of your routines so that they run at the maximum possible rate. This technique can be invaluable for programs such as fractal generators or 3D routines, which need to perform vast numbers of calculations every second.

Warning! COMPTEST OFF completely disables the Control-C feature! So if a problem occurs you'll be unable to get back to the AMOS Pro editor without rebooting. It's therefore vital to SAVE your program onto the disk before using this feature, as otherwise you could lose hours of valuable work!

COMPTEST

COMPTEST performs a single test in your program. It's called after a COMPTEST OFF to provide you with manual control over the entire testing process.

COMP LOAD *(Load the main compiler software)*

```
COMP LOAD ["path"]
```

COMP LOAD loads the full Compiler utility into memory. As a default, the Compiler will be taken from the APCmp file in "APSystem" folder of your current boot disk. A "file not found" message will be returned if it's not available.

The optional pathname allows you to load the Compiler from anywhere on your present system. This feature is mainly intended for owners of hard disks who wish to keep the Compiler in a separate folder such as "C:". It's also been created so that the Shell Compiler can load in the Compiler code! Examples:

AMOS> COMP LOAD

Loads the Compiler into memory from the APSystem folder.

AMOS> COMP LOAD "C:APCMP"

Reads the Compiler from the "C" directory of your startup disk.

Once it's been installed, the Compiler will remain in memory until you either leave AMOS Pro or remove it yourself with the COMP DEL instruction.

COMP DEL *(Restore the memory used by Compiler)*
COMP DEL

COMP DEL removes the Compiler from memory.

=COMP HERE *(Return Compiler status)*
f=COMP HERE

COMP HERE returns a value of TRUE(-1) if the Compiler is available from memory or FALSE (0) if it is not installed.

=COMP ERR$ *(Return last error message)*
m$=COMP ERR$

Returns the most recent error message generated by the Compiler. If there's been no error since the last compilation you'll receive an empty string "" instead.

=COMP SIZE *(Return compiled program size)*
s=COMP SIZE

The effect of COMP SIZE varies depending on when it is called, but it normally holds the size of the last successfully compiled program.
If you call it immediately after a compilation, you'll get the length of the program in bytes. Next time, you'll be presented with the number of instructions in your program. Any further attempts will return a value of zero.

=SQUASH *(Compact an area of memory)*

L=SQUASH(Address,Length,Flag,Ahead,Colour)

SQUASH compresses an area of memory into a fraction of its normal size. The squashing process can be stopped at any time by pressing Control-C.
Address holds the address in memory of your selected memory block. Usually, this will be the start of an AMOS memory bank but you can also compress other memory areas with this system as well.
Never attempt to squash a sprite or icon bank. These are not stored in one continuous memory block, so they can't be compressed using a simple SQUASH operation. You can however, freely squash all other banks including those containing samples or menus.
Length specifies the number of bytes of data to be compacted.

Warning! The squasher will automatically overwrite your existing data with the new packed version.

Flag toggles the squasher between two modes:

Flag=0: Sets SLOW MODE. This reserves the absolute minimum space for the compression process. As the name suggests, the squashing may take some time to complete.

Flag<>0: Activates FAST MODE and compacts the data in a separate memory area for the maximum possible speed. In order to use this feature, you'll need enough RAM to hold a memory block at least twice the size as the original data. If you run out of space AMOS will automatically flip back to SLOW mode.

If SLOW mode has been set, *ahead* specifies how far (in bytes) the squasher will search ahead during the compaction process. The higher the figure the greater the compaction ratio and the slower the execution speed. The minimum is 256 (fast but inefficient) and the maximum is 4096 (very slow). A reasonable value to use is about 1024.

Colour is the number of a colour (0-31) to be flashed on the screen while the data is being squashed. After your data has been compacted the squasher will return one of the following values:

L>0: This indicates the squash was successful and returns the new length of the compressed data.
L=0: The squash was halted with Control-C.
L<0: The compression has been aborted because the squashed length turns out to be longer than the original. This is very rare but it can occasionally happen. The worst offenders are sound samples or banks containing compacted IFF screens.

=UNSQUASH *(Unpack a squashed memory area)*

L=UNSQUASH (Address,Length)

UNSQUASH restores a block of data to it's original state. It's used after a SQUASH instruction to unpack the selected memory area.

Address holds the address of the first byte to be unpacked.

Length specifies the number of bytes to be de-compressed. This should be the length of the packed data returned by the SQUASH command.

Beware! UNSQUASH completely overwrites the current contents in your chosen memory block. So you'll need to reserve a bank of the original (unsquashed) length to hold the new data. You can then BLOAD the packed data at the beginning of this bank and call the UNSQUASH function. Example:

```
Rem
Rem Squasher demo
Rem This kind of picture is very difficult for squashers!
Rem
Screen Open 0,640,200,2,Hires
Colour 1,$FFF
Rem
For N=0 To 80
    Circle Rnd(639),Rnd(199),Rnd(50)+1
Next N
Rem
Screen Open 1,640,3*8,4,Hires
```

```
Curs Off
Screen Display 1,,220,, : Wait Vbl
Rem
Do
     Bell : Centre At(,1)+"Press any key to Squash"
     Wait Key
     Rem
     Screen 0 : Screen To Front 0 : Wait Vbl
     Timer=0
     S=Squash(Logbase(0),16000,-1,1024,17)
     Rem
     Screen 1 : Screen To Front 1 : Wait Vbl
     Cls
     Centre At(,0)+"Picture squashed in "+Str$(Timer/50)+" seconds"
     Centre At(,1)+"Original size: 16000 bytes, squashed size:"+Str$(S)
     Centre At(,2)+"Press any key to unsquash the picture": Wait Key
     Rem
     Screen 0 : Screen to Front 1 : Wait Vbl
     L=Unsquash(Logbase(0),S)
     Rem
     Screen 1 : Screen to Front 1 : Wait Vbl
     Cls
     Centre At(,0)+"Picture restored!"
Loop
```

Squashing using the Power Packer Library

The well known Power Packer Library can be called directly from AMOS Pro using the commands below. When you use these commands the library will be read from the Libs: directory. So you must ensure it's available.

PPLOAD *(Loads and unpacks a file crunched with the PPSAVE command)*

PPLOAD "Filename", [b]

A compacted file can be accessed from disk and unpacked into bank *b*. The file must have been previously saved using the PPSAVE command.

If the bank number is absent, the file will be unpacked into the bank from where it was originally saved.

PPSAVE *(Compacts a bank using the Power Packer Library and saves it as a file)*

PPSAVE "Filename",b [,effeciency]

This command allows you to save out any type of bank into a crunched file. The file can then be read in again using PPLOAD. Saving in this way will allow you to save valuable disk space. The optional *effeciency* parameter allows you to state how well the bank will be crunched:

0 = Fast
1 = Mediocre
2 = Good
3 = Very good
4 = Best

If efficiency is not present the command will use a value of 2 (Good) to crunch the file.

Power packer errors

Cannot open powerpacker.library (v35)

If the library cannot be accessed from the Libs: directory you'll receive this error. It's also important to be using Version 35. We found this to be the only reliable library. Some versions of power packer do not report the "Cannot pack bank" error, in which case the data is not packed as it should be. If this hapens or you just want to be sure packing will happen correctly, then ensure your Libs: directory contains the library from the "AMOSPro_Compiler" disk.

Cannot pack bank

This occurs when the size of the packed bank exceeds the size of the unpacked bank. This usually happens with samples.

Not a powerpacked bank

The data file you're attempting to load is not a powerpacked file.

Please also note

- PPLOAD will refuse to load a file that has been saved using the normal AMOS Pro SAVE command and then packed outside of AMOS by powerpacker iteself. PPSAVE actually adds information to the packed file so that PPLOAD knows which bank the data has to be unpacked into.
- PPSAVE has to reserve a buffer the size of the bank to be packed so that it can prepare the squashed data for saving. So it's important to check your memory requirements when using PPSAVE.
- PPLOAD will only need to reserve a buffer like PPSAVE does when it's been instructed to load an object or icon bank. All other banks are unsquashed directly into the actual bank.
- The new powerpacker commands work amazingly well with the AMOS Pro packed picture banks. For example, we PPSAVED a packed picture bank of 32K and created a new file of 8.5K in size. The actual IFF picture, which was 100K in size, came out at 23K when only compacted by powerpacker.

8: Compiling from the CLI

Introduction

If you're a CLI enthusiast, you'll naturally wish to access the AMOS Pro Compiler directly from the command line. This can be accomplished using the all new APCmp command! As you'd expect from the rest of the package it's a great improvement from the original Compiler and includes lots of additional features.

The new system can effortlessly compile programs from AMOS, EASY AMOS or AMOS Professional. It's also able to read Ascii files directly from a text editor such as CED. What's more, the memory management routines have been completely automated, so that the package makes the maximum use of any extra memory on your machine.

We've taken the opportunity to replace the original command line options with longer, more readable versions. These follow the standard conventions used by Amiga Dos 2.0. So all the previous ACMP commands (from the original Compiler) are now obsolete.

In all, the AMOS Pro Compiler has been given a complete face-lift and there's plenty of new stuff to catch up on. But once you've got over the culture shock, we think you'll like the changes.

Compiler overview

The syntax of the APCmp command is:

APCmp [FROM] "Source.AMOS"

Compiles an AMOS program into machine code.

After you've selected the source file, you can include a number of additional options:

Option	**Effect**	**Replaces**
TO "Destination_Name"	Chooses a destination file for your program.	(-o)
TYPE=0/1/2/3	Selects the Type of the compiled program.	(-t)
DEF/DEFAULT	Automatically creates a screen when the compiled program starts.	(-s1)
NODEF/NODEFAULT	Starts the compiled program cleanly with a blank display.	(-s0)
ERR/ERRORS	Includes the normal AMOS error messages in the compiled program.	(-e1)
NOERR/NOERRORS	Doesn't include any error messages in the compiled program.	(-e0)
WB	Starts the program in the background.	(-w1)
NOWB	Sets up a new display the moment the compiled program is run.	(-w0)
QUIET	Suppresses Compiler messages.	(-q)
LIBS="Libs_folder"	Tells the Compiler where the APSystem files are located.	(-f)
CONFIG="Config_file"	Loads the default Compiler settings from a selected file.	(-c)
LONG	Uses a 68000 JMP command for all branches in the compiled program.	(-l)
NOLONG	Uses short BRA.S instructions in the program.	
TEMP="Temp_folder"	Specifies the directory where any temporary Compiler files will be stored.	

TOKEN	Tests an Ascii file and converts it into ".AMOS" format.
INCLIB	Includes the entire AMOS Library as part of your compiled program.
NOLIB	Uses a shared AMOS.Library from the LIBS: folder.

Note that there's no equivalent to the "-d" options from the original Compiler, as they've been totally superseded by the new optimisation features. Let's have a look at some typical command lines from the CLI:

```
1> APCmp MyProg.AMOS
```

Compiles "MyProg.AMOS" and places the resulting program in a file called "MyProg".

```
1> APCmp FROM MyProg.AMOS TO Compiled_Version
```

Compiles "MyProg.AMOS" and saves the code in a new file called "Compiled_Version"

```
1> APCmp FROM MyProg.Asc TOKEN
```

Reads an Ascii version of your program from "MyProg.Asc" and converts it into a normal AMOS program called "MyProg.AMOS".

```
1> APCmp FROM MyProg.Asc TO Compiled_Version WB ERRORS NODEFAULT
```

Compiles an Ascii file with the WB,ERRORS and NODEFAULT options. The new program will start off in the background and will only be displayed after its initialisation process has been completed.

```
1> APCmp FROM MyProg.AMOS TEMP="Ram:" LIBS="Ram:APSystem"
```

Compiles "MyProg.AMOS". A Ram disk will be used to hold both the temporary files and AMOS system libraries. So compilation will be very fast.

The Compiler options in depth

FROM *(Chooses a source program to compile)*

FROM file_name.AMOS
FROM file_name.ASC

This selects a source program to be compiled from the disk. The filename can be anything you like but if it includes spaces you'll need to enclose it in quotation marks like so:

```
1> APCmp FROM "file name with spaces.AMOS"
```

"FROM" is completely optional and can be omitted completely:

```
1> APCmp File.AMOS
```

APCmp allows you to compile programs in either ".AMOS" format or in raw Ascii. Your ".AMOS" files can be created using AMOS, EASY AMOS or AMOS PRO. All source programs should be carefully tested from the editor before they're saved onto the disk. The AMOS Pro Compiler works a lot faster with tested programs.

Ascii files can be generated by any standard text editor or word processor. But you'll need to ensure that they don't contain any embedded control codes. This is especially important if you're using a word processor, as you'll probably have to save your files using a special menu option. Check out your word processor manual for more information.

Here are a few further tips to using Ascii files.

1. Don't forget to load in any memory banks explicitly as part of your program.
2. If you're running short of memory, convert your files into ".AMOS" format before compilation. Use a line like:

```
1> APCmp File.ASC TOKEN
```

If there are no mistakes, you can now compile this file with:

```
1> APCmp File.AMOS
```

3. If your word processor has a spell checker, feel free to use it! Since AMOS commands resemble standard English, most spell checkers will treat them as normal text. So you can easily check the spelling of any Rems or screen messages in your programs.

TO *(Chooses a destination file for your program. Replaces -o)*

TO "Destination_Name"

Tells the Compiler where you want to place the compiled program. If the destination is omitted the program will be stored in the current directory using an amended version of the original source name.

```
1> APCmp File.AMOS TO "Df1:File"
```

Compiles "File.AMOS" and saves the result in "File" on drive 1.

TYPE *(Sets the "type" of the compiled program. Replaces -t)*

The TYPE option allows you to choose the type of the compiled program. There are four possibilities:

TYPE=0 (Workbench)
Generates a Workbench program. This can be run from the Workbench by simply clicking on its icon. An appropriate icon will be generated automatically by the Compiler.

TYPE=1 (CLI)
Produces a program in standard CLI format. The compiled program can now be called straight from the command line. If the line includes parameters, they can be read directly from your compiled program using the COMMAND LINE$ function.

Apart from the icon, there's absolutely no difference between the CLI or the Workbench versions of your program. So you can happily execute your Workbench programs from the CLI if required.

TYPE=2 (Multitasking)
Creates a CLI program that will automatically run in the background and detaches itself from the CLI from which it was run. It's equivalent to executing your programs with the Amiga Dos RUN command.

Warning! This option automatically changes the current directory to "SYS:" at the start of your program. So if your routine accesses the disk, you'll need to include the full pathname as part of your commands. You can also reset it directly using the DIR$ function like so:

```
DIR$="Work:AMOS_Games/"
```

TYPE=3 (AMOS Professional format)
Compiles a program in ".AMOS" format. This can only be run from the AMOS Pro Editor. Any attempt to call it from the Workbench or CLI will fail. You'll also get an appropriate error message if you try to load these files into AMOS 1.3x or Easy AMOS.

NODEF/NODEFAULT *(Starts a compiled program with a blank screen. Replaces -s0)*

As a rule, your compiled programs will automatically create a default screen for your graphics. This screen is very useful during development but it produces an annoying start to your programs when you'd rather have a nice fade in to your own title screen.

If you want to generate smooth transition effects, it's better to define all the screens directly in your program. This can be achieved using the NODEFAULT option which stops the Compiler from creating screen zero. You've now got total control over the Amiga's display. But you'll have to open a new screen before using any of the text or graphic commands. Otherwise your program will immediately stop with a "Screen not opened" error.

DEF/DEFAULT *(Automatically creates a default screen. Replaces -s1)*

This forces your compiled program to create a standard screen for your text and graphics. As you'd expect from its name, it's set up as the default.

ERR/ERRORS *(Includes AMOS error messages in the compiled program. Replaces -e1)*

Normally, the standard AMOS error messages are not included with your compiled program at all. So if a problem occurs or you press Control-C, you'll be returned straight back to where the program was run from.

During development, you may find it useful to save the error messages as part of your compiled program. If you get a problem, your program will now display an appropriate error message on the screen.

Note: Since a compiled program is in machine code, there's no way of reporting the precise line number at which the problem occurred. However, you'll still have a good idea about the nature of the error. You can now track it down by loading AMOS or AMOS Pro and running the source program through the editor.

NOERR/NOERRORS *(Doesn't include error messages in the program. Replaces -e0)*

Omits the standard AMOS error messages from the compiled program. So if there's a problem, your program will abort immediately without reporting an error.

WB *(Starts the program in the background. Replaces -w1)*

WB suppresses the compiled program display and leaves the old CLI or Workbench screens intact. The effect is similar to calling the AMOS TO BACK command.

You can use it to construct your game screens invisibly in the background, without affecting the existing display. Once your screens have been safely initialised, you can then switch them neatly into place with an AMOS TO FRONT command. Here's an example:

```
Rem Compile with NODEF and WB options set
Rem e.g APCmp View.AMOS NODEF WB
Rem Turn off screen updates
Rem Open a new screen
Screen Open 0,320,200,64,Lowres
Rem Draw a simple screen
For C=0 To 100
        RAND:
        Ink Rnd(64):X1=Rnd(320) : X2=Rnd(320) : Y1=Rnd(200) : Y2=Rnd(200)
        If X1=X2 Then Goto RAND
        If Y1=Y2 Then Goto RAND
        Bar Min(X1,X2),Min(Y1,Y2) To Max(X1,X2),Max(Y1,Y2)
Next C
Rem Flick display into place
Amos To Front
Wait Key
```

This feature can be easily exploited to create standard CLI utility programs. However, if you want to output text to a CLI window, you'll have to call the Amiga Dos libraries directly.

Note: Easy AMOS users will not be able to take advantage of this feature directly. This is because the AMOS TO FRONT command is not in their instruction set. To work around this you'll have to save your program out as Ascii, edit the file in a word processor to add the AMOS TO FRONT command, and then compile the program as Ascii.

NOWB *(Sets up a new display the moment the compiled program is run. -w0)*

NOWB (the default) turns off the existing display before your program is run. When you return to the Workbench or CLI the contents of the screen windows will be completely redrawn.

QUIET *(Suppresses Compiler messages. Replaces -q)*

QUIET prevents all messages during the compilation process. The only exceptions are error reports which are displayed normally.

LIBS *(Specifies position of APSystem files. Replaces -f)*

LIBS="pathname"

LIBS instructs the Compiler to load its library files from the chosen directory. The default pathname is normally taken from the configuration settings on your start-up disk:

S:AMOSPro_Interpreter_Config

pathname is a standard Amiga Dos file spec ending with a ":" or "/" character. The Compiler will read all filenames from the configuration file and substitute their pathnames with the new directory.

You can harness this feature to access the Compiler libraries from a RAM disk. This consumes a lot of memory but results in blindingly fast compilation. Here's the procedure:

First copy the folder to the RAM disk with:

```
1> cd Ram:
1> makedir APSystem
1> cd AMOSPro_Compiler:
1> copy APSystem To Ram:APSystem
```

You can now compile your program with the LIBS option like so:

```
1> APCmp FROM Example.AMOS LIBS="Ram:APSystem"
```

CONFIG *(Changes the configuration file. Replaces -c)*

CONFIG="Config_file"

CONFIG reads the Compiler settings from the selected file. If the new configuration file is unavailable the compilation will be aborted. The default pathname is:

```
S:AMOSPro_Compiler_Config
```

LONG *(Long jumps rather than short Branches. Replaces -l)*

As a default, APCmp uses a 68000 BRA instruction for all jumps within the compiled program. This leads to the fastest possible execution, but it's incapable of handling really large program structures.

If the length of WHILE..WEND or IF..ELSE..ENDIF structures exceeds 32k, the Compiler will be unable to use the standard BRA instruction. So you'll end up with a "Program Structure too long" error.

The solution is to force APCmp to use a JMP command instead of a relative branch. This can be achieved by simply adding in a LONG directive at the end of the command line and recompiling your program as before. For example:

```
1>APCmp Large_Program.AMOS LONG
```

The program will now compile and can be run without problems.

If you're regularly compiling large programs, you can set up this system as a default using a special option from the Compiler Shell. See Chapter 6 for more details.

NOLONG *(Compiles a program using the BRA instruction for any jumps)*

Reverts to the most efficient way of handling jumps in your program. Use it if you've selected the "Long Jump" option as part of the Compiler configuration and want to squeeze the maximum space saving performance out of your compiled program.

TEMP *(Specifies where the temporary files are to be stored)*

TEMP="Temp_folder"

APCmp generates a number of internal files during compilation. These are used to hold temporary information and are deleted automatically when you return to the CLI. You can select the directory where these files are to be stored using the TEMP option.

If you've plenty of memory you can speed up the compilation dramatically by insisting that all temporary files are saved onto the Ram disk. The line you'll need is:

```
1> APCmp FROM Program.AMOS TEMP="Ram:"
```

TOKEN *(Tokenises and tests an Ascii file)*

This option converts the Compiler into a simple tokeniser. It takes an Ascii file, tokenises the commands, tests for errors, and saves the results in a normal ".AMOS" program file.

The new file can then be loaded straight into AMOS Pro or compiled using APCmp, just as if it had been created from the editor. Example:

```
1> APCmp "Test.Asc" TOKEN
```

Results in the file Test.AMOS.

NOLIB *(Use shared AMOS.Library)*

The original AMOS Compiler copied the entire AMOS library into each and every program. In practice, this was pretty wasteful, as it led to massive programs. Even a simple "Hello World" routine could take over 50K on the disk.

Fortunately, we've improved things dramatically with the AMOS Pro Compiler. The new system stores a single copy of the appropriate library routines into the LIBS: folder, just like any other Amiga application.

The AMOS.Library can now be shared between all your compiled programs. So your files will only contain the code which is actually required for your listings. As a result, the compiled programs will be around 45K smaller.

INCLIB *(Create a stand-alone program)*

If you want to create a program which is completely independent of the AMOS system, you can optionally include all the library routines into your compiled code. This will make all your programs around 45K longer than the standard versions, but you won't have to worry about the contents of the LIBS: folder on the start-up disk. You'll also be sure you're accessing the latest AMOS libraries in your program.

Amiga DOS command template

Finally, we'll present you with a complete Amiga Dos command template for the AMOS Pro Compiler.

```
APCMP FROM/A, TO/K, TYPE/K/N, ERR/K, NOERR/K, LONG/K,
      NOLONG/K,DEF=DEFAULT/K, NODEF=NODEFAULT/K, WB/K, NOWB/K,
      QUIET/K, TEMP/K/A, LIBS/K/A, CONFIG/K/A, INCLIB/K, NOLIB/K
```

9: Running a compiled program

There are several possibilities:

Stand-alone programs

These are independent of the AMOS Professional package and can be run on any Amiga, even if you don't have a copy of AMOS Pro.

Note: If your compiled program used the INCLIB option, your program will expect to find the "AMOS.library" in the LIBS: directory of the start-up disk. If this is a problem, you can easily include the necessary system routines in your compiled program. Just choose the appropriate set up option from the Compiler Shell or add in the INCLIB directive to your command line.

Workbench programs

If you've compiled your program using the "WB exec" option, you'll be able to execute it straight from the Workbench by simply clicking on its icon with the mouse.

CLI programs

These programs can be called up directly from the command line by typing their name. They are created when you select the "CLI exec" option from the compiler shell or with the TYPE=1 directive. For example:

```
1>Demo
```

This runs a program called "Demo" from the current disk.

The Amiga treats these CLI programs exactly like any other machine code command, so they can be run from within batch files or executed automatically during your start-up sequence.

You can also include parameters in the command line as well. These will be copied into the reserved variable COMMAND LINE$ when your program is initialised. For example:

```
1>Demo these are parameters
```

Executes the compiled program called "Demo" and loads the line "these are parameters" into COMMAND LINE$.

AMOS Pro compatible programs

These programs can only be run from an AMOS Pro window. They are not compatible with either the AMOS or EASY AMOS editors. So if you've yet to upgrade, you'll need to avoid this format completely and run your compiled programs from the CLI or Workbench instead.

Since these programs are executed directly from AMOS Professional they can make use of the existing AMOS system libraries in memory. As a result, they are considerably smaller than the equivalent CLI or Workbench versions.

AMOS Pro programs use their own special format, which should be selected before compilation. Set the TYPE option to "AMOS Compiled" from the Compiler Shell or use the "TYPE" directive from the CLI:

```
1>APCmp Program.AMOS TYPE=3
```

Once you've compiled a ".AMOS" program, you can load it straight into AMOS Professional and run it from its very own program window.

The compiled program is stored in its own editor window and is treated just like any normal AMOS Professional program. All memory banks are in standard format and can be loaded, saved or grabbed as required. You can even execute a compiled program from the AMOS Pro menu!

Leaving a compiled program
When your compiled programs are through, they'll automatically return you to the original calling environment. You can also abort at any time by holding down the Control and C keys, (as long as BREAK OFF hasn't been set by the compiled program).

Error messages
If you've compiled your program with the "Include error messages" or ERR option from the command line, a standard AMOS error report will be displayed when something goes wrong. Alternatively, if you've omitted these messages, your program will exit immediately in the event of an error. The error messages are displayed differently depending on the environment you are using:

Workbench: The message will be displayed in a small alert box.

CLI: The error will be reported to the current CLI window.

".AMOS" programs: You'll see the error message on the INFO line of the active editor window.

The compiler uses the same error messages as the interpreter, but there are no line numbers. So if you want to find the exact position at which the error occurred, you'll need to check out the original source program from within
your AMOS editor.

10: Technical details

To most people the technical details of the Compiler will be completely irrelevant, as they're not needed to use the package in any way.

However, if you're an expert, you may be interested in a quick glimpse into the inner workings of the Compiler.

The AMOS Pro Compiler has been given a complete overhaul since its previous incarnation. We've taken the original system and made the following improvements:

The new memory system

The Compiler now configures itself automatically to the available memory. Any buffers are allocated intelligently by the system. So there's no need to muck around with the original memory options.

Compiled programs used to be stored in a single memory hunk. Whole source codes would occupy a contiguous 200k block of memory. This was hard to load on small machines, especially when memory was running tight. It also generated lots of annoying "Out of Memory" errors when you tried to launch a new program with the RUN command.

The AMOS Pro Compiler now splits the code into three separate hunks. There's one hunk for the compiled program, another for the AMOS Libraries and a third for the support routines such as file-selectors. As a result, compiled programs are much less vulnerable to memory fragmentation and make fewer demands on your Amiga's ram.

Smaller compiled programs

The AMOS Pro Compiler now stores all Basic instructions in the "AMOS.Library" in the "LIBS:" folder. This library can be shared between any number of programs. So it doesn't need to be saved directly as part of the compiled file. This frees 45K from each program and allows you to create small CLI routines in just a few kilobytes.

If you're running a ".AMOS" type program from AMOS Pro, it will now be able to call all these functions directly, as the "AMOS.Library" has already been loaded by the interpreter.

Faster calculations

All calculations are now performed using processor registers, rather than the stack. So they're much faster than the original AMOS versions. We've also heavily optimised many operations. Where possible, ADDQ and SUBQ are substituted for ADD and SUB. And if one of the parameters is an exact power of two, LSR.L and LSL.L instructions are used to replace slow MUL or DIV operations.

The base address of the variables are now kept permanently in an address register for extra speed. Here's an example of a typical calculation:

A=B*16 will be compiled as:

```
Move.l          B(a6),d3
Moveq           #4,d0
Lsl.l   d0,d3
Move.l          d3,a(a6)
```

What's more, the Compiler now handles double precision. We've even beefed up the VAL instruction so that it works under any situation.

Faster library calls

Library calls are now performed using relative jumps to an address register, making them faster and smaller. We've also improved the error traps. The old version of the Compiler had to insert a "Lea 2(pc),a4" instruction before every function call. This is no longer need-

ed in AMOS Pro. Finally, here's an example of some compiled code for you to examine.
We'll take the line: "Screen Display 1,X+128,Y Mouse*2-50,,200" and see how it's converted into machine code:

```
Moveq #1,d3                "1"
Move.l d3,-(a3)
Move.l X(a6),d3            "X"
Add.l #128,d3              "X+128"
Move.l d3,-(a3)
Jsr Y_Mouse(a4)            "Y Mouse"
Lsl.l #1,d3                "Y Mouse*2"
Sub.l #50,d3               "Y Mouse*2-50"
Move.l d3,-(a3)
Move.l #Null,-(a3)         "handle the ,,"
Move.l #200,d3             "200"
Jsr Screen_Display(a4)     "call Screen Display"
```

As you can see, this code is very efficient. The last parameter is left in d3 to save some processor time.

11: Trouble shooting

Most problems that occur with the Compiler can be easily sorted by knowing more about the way it works. So read this chapter to familiarise yourself with the Compiler's few idiosyncrasies.

Points of confusion

- You receive an error when you try to load compiled ".AMOS" programs into AMOS Basic.

Compiled ".AMOS" programs can only be loaded into the latest AMOS Professional editor. You can't use them with either Easy AMOS or AMOS Basic.

- You receive a "Not an AMOS program error" when compiling an Ascii file.

The AMOS Pro Compiler automatically detects if your source file is in text format and adjusts itself accordingly. But it's only able to handle raw Ascii, so if you try to compile documents directly from a word processor you'll have real problems. You'll need to check out the "SAVE Ascii" or "TEXT EXPORT" options from your word processor, and resave your file in the correct format.

- You are unable to use the Help system

In order to use the help system, you need at least 150K of free RAM. Although you're unlikely to have problems if you're running the Shell from the Workbench, AMOS Pro users can occasionally encounter memory problems from the Editor.

If you run out of memory, save your current program and try loading the Compiler Shell directly into the editor window. This may make all the difference.

Memory problems

- The Compiler Shell gives an "Out of Memory" error

If you're using the Shell program from within AMOS Pro, you can occasionally run out of memory. There are several strategies you can adopt:

1. Remove the Shell from memory and compile your program from Direct mode using a line such as:

   AMOS>Compile "Source.AMOS"

This will save around 100K of space for compiling work.

2. If the Shell has copied the Compiler libraries into a ram disk, call the setup dialogue and turn this feature off. Then select the "Save Config" button from the Main setup box and reboot your Amiga with the new settings. The Compiler may run noticeably slower but you'll have masses of extra memory.

3. Run the stand alone Compiler Shell from the Workbench.

Note: The Compiler Shell will also cut down it's memory use automatically when things get a bit tight. If any animation or music is loaded they'll be the first items to be deleted. If the

Shell needs more memory it will erase the animated buttons if they're selected. The final thing it can delete are the libraries from the Ram disk.

- You get an "Out of memory error" from Direct mode or the CLI.

The AMOS Pro Compiler automatically makes the most out of any available memory. So you should be able to compile some sizable programs on even the smallest Amiga.

If the Compiler runs short of space, it will normally read your program into memory a chunk at a time, as long as it is a tested program. But this is only possible if the program has been saved from AMOS. In order to test and interpret an Ascii program, the Compiler has to load the entire file into memory, and there's no guarantee it will fit.

- You get an "Out of memory error" when you try to run a ".AMOS" compiled program from AMOS Professional.

One simple way of minimising the memory consumption is to remove all the memory banks used to hold your sprite, music or screen data from the main program. These banks can then be loaded separately from the disk at the start of your routine. Once the routine's finished with the bank it can then erase it. This will then reduce the final size of the compiled program.

If you still run out of space you may need to run your program outside of AMOS Pro. Do this by compiling your program to CLI or Workbench format, you'll then be able to use the maximum available memory on your machine.

- The AMOS RUN command fails from within a compiled program

The AMOS RUN command can be used to link together your compiled programs. However, it only works with CLI or Workbench programs. You can't run a ".AMOS" program in this way.

On small machines you may find it hard to run a second program due to memory restrictions. If so, you can optimise the memory in two ways:

1. Close all screens, erase all memory banks and clear all AMAL command strings before running the program.
2. Place the "-Mem" option in COMMAND LINE$. This forces RUN to free as much memory as possible on start-up. Example:

   Command Line$="-Mem" : Run "Myprog"

Compiler refuses to compile your AMOS program

- Program structure too long, compile with LONG option

This error occurs when an IF...ELSE...ENDIF or a WHILE...WEND needs to jump more than 32k in the compiled program. So the Compiler will need to replace the Branch instructions in the program with the equivalent Jump commands.

Don't worry if all this seems like total gobbledygook. All you have to do is set the LONG option from the setup menu and recompile. The final program may be a little longer and run marginally slower but otherwise all will be well.

Configuration Problems

The AMOS Pro Compiler relies on information held in two configuration files. If these files are accidentally corrupted the system may break down.

- If both the Compiler and AMOS Pro fail to work. You may get one of the following messages:

  "Bad Interpreter configuration file"
  "Cannot load Interpreter configuration"

The solution:

1. Copy the following files from your AMOS Pro masters onto your start-up disk:

   S:AMOSPro_Interpreter_Config
   S:AMOSPro_Compiler_Config

2. Check that you don't have a second AMOSPro_Interpreter_Config file in the same directory as "AMOSPro". If so delete the erroneous file.

3. Check that the "AMOS.Library" is inside the LIBS: directory of your current start-up disk.

- If AMOS Pro works but the Compiler fails, you may get a message such as:

  "Bad Compiler configuration file"
  "Cannot load Compiler configuration"

In this case your Compiler settings may have been corrupted. So you'll need to replace the damaged "S:AMOSPro_Compiler_Config" file with the original version on the master disks.
Also, check that you don't have a second AMOSPro_Interpreter_Config or AMOSPro_Compiler_Config file in the same directory as the file "AMOSPro". If so delete it.

- A program which normally works in AMOS Pro reports an "Extension not Loaded" error while compiling.

This error will be very rare. You've probably got a second configuration file which is being loaded by mistake. Either:

1. Delete the "AMOSPro_Interpreter_Config" file from the APSystem folder.
2. Edit the configuration so that it uses the same extension list as the Compiler version.
3. Change the pathname of the "Interpreter_Config" to point to the alternative configuration file.

- You get a "Cannot find APSystem folder" error

There's a mistake in the "S:Interpreter_Config" file. You may need to copy the default configuration from your AMOS Pro masters.

When you use APCmp or Shell, certain options seem forced

- You can receive an "Error in the Compiler configuration's command line"

Load the Compiler Shell and use the setup option to check the default command line. You may have inadvertently changed a couple of options to silly values.

Other errors

- **Cannot Open Math Libraries**

The Compiler has attempted to open the mathematics libraries in the "LIBS:" folder of your current boot disk. If you don't have a hard drive make sure that you've a copy of your AMOS Pro start-up disk in any drive.

- **Cannot locate icon information**

The Compiler can't find either the "Icon.Library" or load in the "def_compiled.info" file from the APSystem folder.

- **Division by zero**

The Compiler has attempted to optimise your program by calculating all constants in advance. If there's a division by zero on one of your lines you'll get a "Division by zero" error. e.g A=2/0

- **Cannot load Editor configuration** or **Bad editor configuration**

An error has occurred and the Compiler can't find the messages. Check your startup disk to see if they are present in the APSystem folder.

europress SOFTWARE

Europa House, Adlington Park, Macclesfield SK10 4NP. Tel: 0625 859333