WARNING!

BEFORE PROCEEDING ANY FURTHER WITH THE INSTALLATION,
READ THE CHAPTER "INSTALLING CRAFT"

When the installer has given you your personalized registration number, note it in this box:

Programmed by Hannu Rummukainen
Installer and accessories by Janne Kalliola
Example programmes by Hannu Rummukainen & Janne Kalliola
Manual by Janne Kalliola & Hannu Rummukainen
Graphics by Ari Seppii
Example module by Janne Leinonen
Additional graphics by Jussi Lkif

CONTENTS

01: CRAFT

CRAFT is an extension for AMOS The Creator and AMOS Professional. It adds almost 160 new instructions and functions to your AMOS configuration. It contains a very good colour handling system including multi-palette banks and colour spreading. You can play with two requesters, too. CRAFT has a flexible audio system to be used with Soundtracker or Protracker modules. CRAFT enables you to create pictures of Mandelbrot and Julia fractal sets, easily and fast. It also has a good selection of turtle drawing instructions.
Why did we select the name CRAFT among thousands of possibilities? There are two main reasons: Craft means skill, profession and shrewdness, which are needed when programming commercial or PD programmes on the Amiga. Another reason is that CRAFT stands for Colours, Requester, Audio, Fractals and Turtle.

02: INSTALLING CRAFT

Before doing anything else you should insert the CRAFT disc into the drive DFO and turn your Amiga on. When the programme asks you for your name, enter your first name in the upper box and your surname in the lower one. Remember that the disc in the drive should be write enabled, otherwise an error will occur. When you have done this, the installer shows your personalized registration number on the screen. Enter it into the box you will find on the first page of the handbook. Now, reset your Amiga and make a copy of your CRAFT disc. If you use Workbench's diskcopy, you have to rename the disc, because our installer doesn't understand disc named 'Copy of CRAFT". So get rid of 'Copy of'. Now reset the machine again and insert the copy in drive DFO. Keep the original in a safe place far away from magnets and direct sunlight.

You have to have one blank disc before you start the installing process. Attach the CRAFT Extras disc label to this blank disc. If you don't want to lose your second label supplied, you should attach it to another blank disc. This disc isn't needed when installing, but you need it when you want to get future updates (See chapter "Updating CRAFT). Put the disc in a safe place and start the installer again. Now there will be no boxes asking for your name, but the programme shows the main installation screen. Select the version you want to install (currently AMOS 1.3x, AMOS Pro 1.0 and AMOS Pro 1.11 (or above)). CRAFT designed for AMOS 1.34 differs from CRAFT designed for AMOS Pro, but there are no differences between versions for AMOS Pro 1.0 and AMOS Pros with higher version numbers, but the configuration files are a bit different. Note that if you have AMOS Pro Compiler and AMOS Pro version 2.0, you may use the AMOS Pro v1.11 installing process, as the extensions and configuration files don't differ from each other. When you have selected the version, the programme will take over control.

Just do what it tells you to do. Remember when the programme asks for AMOS: or AMOSPRO_System:, use the COPY of it! The original is for safe-keeping. The programme copies CRAFT to the disc and alters the configuration files, meaning you have to keep the disc unprotected. If you haven't already made backup copies of your AMOS or AMOSPro discs, do it now. When the installer has installed CRAFT on disc, it asks you to put a blank disc in the internal drive. The installer formats the disc and gives it a name of 'CRAFT_Extras:'. Note that there are no checks whether the disc is blank or not, but the installer destroys all the previous data on the disc. After formatting is finished, the installer starts to copy files to the disc. If you have only one disc drive, the programme asks you to swap discs, but if you do have another drive, you can put the CRAFT_Extras: volume into it and the CRAFT disc into the internal drive (or vice versa).

03: COPYRIGHT NOTICE

Please keep in mind that it's illegal to copy this product. Everyone who has CRAFT without the originally supplied discs and manuals has committed a crime. You should also know that it's illegal to copy this manual in any form. If you have an illegal copy of this manual, it is also an offence against the law.

If you make programmes which are passed on to other people in any way, you have to compile them. It's illegal to spread this extension in any form. The only exception to these rulings is if you put the AMOS code only into Public Domain e.g. give it to the AMOS Public Domain Library. In these cases you don't have to compile the programme as the extension is not being spread. We don't mind if you use a few of the CRAFT instructions and don't mention anything in the programme. But if you use more instructions, fractals, turtle, directory reading etc., your programme has to inform the user that there are some CRAFT instructions among the "normal" AMOS instructions. You can place this text after the AMOS credits. If you send your source code to the AMOS PD Library, you should inform the up-keeper of the library that your programme requires CRAFT to function properly.

If you make commercial products including CRAFT instructions and you don't include any credits in your programme, but send a copy of the product to Europress, you have to send us a copy, too (to our Finnish address, please see chapter "Updating CRAFT" for details). If Europress decides to publicise that the programme is coded in AMOS, we'll also announce that there were some CRAFT instructions among the AMOS instructions. But if Europress doesn't say anything, we don't say anything either. We, here at Black Legend and Solaris, and probably at Europress would be pleased, if you mention yourself that your product was made with AMOS (Pro) and CRAFT half a year after the release.

We would like to thank the following people:

Charlie Heath for the magnificent A68000 assembler
Jan Jokivuori for additional ideas
Francois Lionet for the excellent AMOS The Creator and for the superb and fantastic AMOS Professional
Risto Paasivirta for the efficient fractal coordinate system

04: BUGS

We have tried to find every bug, but we can't guarantee that this extension is totally bug free and we cannot be made responsible for any damages done using CRAFT (whether it was used improperly or not). If you find any bugs, please tell us your machine's configuration, the versions of AMOS and CRAFT you are using, the conditions where the bug was found and send them in a letter to us:

Black Legend UK Ltd.
"CRAFT - Bug report"
20 Guilford Road
St. Albans, Herts.
AL1 5JY, United Kingdom

05: UPDATING CRAFT

If you have any good ideas for new instructions, send them, too, to Black Legend. We are always interested in ideas for improving CRAFT. Every owner of CRAFT is entitled to updates. Before you can receive any updating packages, you have to fill in and send the registration card to Black Legend UK Limited.

We will inform you about updates in "Totally AMOS" magazine, too. If you want to receive our latest update, follow these simple steps: Send us the disc with the Update disc label (supplied in this box, for anti piracy reasons), a return addressed envelope, and an international reply coupon worth £1. Here's the address to send all this to:

Black Legend (Finland)
Attn: Mr. Janne Kalliola
Museotie 4
FIN-83700 POLVIJARVI, Finland

Note that any disc which has no update disc label on it can not be returned, so take good care of your disc labels. It would be best to attach it to a blank disc soon after opening the package. Another important point is that if you want to speed up the first updating process, format the Updater disc before sending it to us. But if you have already updated CRAFT before, don't do anything to the Updater disc. If you format it again we can't be sure what your current version of CRAFT is and we may send you a wrong version. So always keep the latest update of CRAFT on your update disc. If we change a lot of things, add dozens of instructions and functions, improve some systems etc., the update may probably be more expensive, but it's worth of it, especially if we include an extension to the manual. You should request a new update once or twice a year. Inform us of the version number of your CRAFT disc, and if the latest version is higher, we will send you a copy of it. But if you already have the latest version, we will keep your disc until we've made a new version of CRAFT. So don't panic, if the return of your disc is delayed. Remember that all future updates are not Public Domain like the AMOS updates. We want to cut down the illegal copying of this product, and if you're a registered user, you will surely benefit from this system. Well make updating packages, if people buy our product, and you get better and bug-free extension. So spread neither the updates nor the extensions.

06: HOW TO READ THE SYNTAX OF THE INSTRUCTIONS

The bare name of the instruction/function is written first and it's centered like this:

RESERVE AS PALETTE

If the new name is a function, it has a "=" before its name.
Next, there is the full name/names of the instruction in the left:

Reserve As Palette bank number,amount

Every first letter of the words of the instructions is in upper case and all the others are in lower case. After the definition of an instruction the parameters of the instruction are listed. They are in lower case and separated by comma (",") or "To". If there are several instructions or parameter definitions listed, you can choose any of them. The text following the instruction definition tells you what the differences between the various definitions are. If a word of an instruction is enclosed in brackets, it's optional, but it also changes the work done by the instruction. Please refer to the text following the instruction definition.

If a parameter is enclosed in square brackets ([]), it's optional. If there is another pair of square brackets inside a pair of them, the second parameter is optional, but can be used only when the parameter before it is used etc. If parameters or words in instruction definitions are separated by a slash ("I"), they're selectable. If they aren't enclosed by square brackets, you have to use one of of them. Slashes ("/") in the text following the definitions, the explanations are in the same order as the parameters in the instruction definitions. This also applies to the bare name of an instruction. If you aren't sure how to use an instruction or function, try it until you find the syntax that you were looking at. But before trying an instruction, read the text very carefully following the definition. Some instructions if used improperly can have rather drastic consequences for your system - so be careful, not sorry!

07: COLOURS

When you use packed bitmaps or you want to flash or fade the colours of the "spacked" (screen packed) screen, you have one problem: where to store the palette. You may use variables, but you have to write a lot of text to achieve the desired result. No problemo! for CRAFT; create a bank and duke the values in it. But if you don't want to play with instructions =DEEK, DOKE, =START etc., you can always use CRAFT palette banks. The palette banks can store all the 32 colours on the screen, with a single instruction. You can store many palettes in a single bank; the more memory you have, the more palettes you can store. If you don't want to get all the colours with the single instruction, you can always use masks or instructions which load one colour at the time. And the best thing is: you can save and load these banks normally.

CRAFT also includes instruction which can modify the current screen palette: you can swap colours or spread them. You can change a single component at the time and create some fabulous effects!

Palette **Banks**

RESERVE AS PALETTE

Reserve As Palette bank number,number of palettes

Reserves a palette bank. The palettes can be accessed with instructions PAL TO/FROM/SWAP BANK instructions, please see them for details.

PAL TO BANK

Pal To Bank bank number[,placekmask]]

This instruction puts the current palette in a memory bank in the position (place) given. You can do everything to this bank that you can do to other AMOS banks. You can also load it to the CRAFT Palette accessory, manipulate it, and save it again to disc. As a palette bank can hold a lot of palette definitions, the parameter place informs the programme which to alter.

The place can't be higher than the number of palettes in the bank defined with instruction RESERVE AS PALE1TE.

If you don't use the optional parameter place, the instruction will create a new bank consisting of one palette definition only. If you want to create banks with multiple palettes, please use the RESERVE AS PALETTE instruction. The optional parameter mask limits the colours transferred to the bank. Have a look in your AMOS User Manual for details.

This instruction is almost identical to the following code, but it also can reserve the bank, if needed:

```
Procedure PAL_TO_BANK[B,N,MASKI
  If Length(B)>0
    For I40 To 31
      If Btst(I,MASK)
        Doke Start(B)+N*64+I*2,Colour(I)
      End If
    Next
  End If
End Proc
```

PAL FROM BANK

Pal From Bank bank numbettplacekmask I I

... gets a palette out of a bank. Note that if there is no bank or there are not enough palettes in said bank, an error will be given. Please have a look at the instruction PAL TO BANK for more details about the parameter mask.

PAL TO BANK and PAL FROM BANK are very handy instructions, when changing colours of an IFF or "sparked" screen. You don't have to do anything, but put the current palette in a bank, save it to the ram-disc, run the CRAFT Palette accessory, modify the bank, save it again to the ram-disc, go back to the previous programme, load the bank and get the new palette out of it (... and ... and

We didn't include any fade instructions because you can, if you want to, fade colours from a bank to a screen with a simple programme. You may, if you wish, make a procedure out of the fade routine:

```
Flash Off : Curs Off
Pal To Bank 1 : Rem Save the colours
For I=0 To 15
  Colour I,0 : Rem Set all the colours to black
Next
Cls 0
Screen Open 1,20,20,16,Lowres : Rem Open another screen and hide it
Screen Hide 1
Flash Off
Pal From Bank 1 : Rem Get the bank
Screen 0
For 1=1 To 15
  Ink I
  Bar I*20,0 To I*20+20,20 : Rem Draw some boxes to see the fading
Next
Wait Key
Fade 1 To 1 : Rem Fade the palette
Wait 15
```

PAL SWAP BANK

Pal Swap Bank bank numberkplaceLmaskll

This instruction swaps the current palette and the palette in a bank.The optional mask limits the colours transferred. Please have a look in your AMOS User Manual for details on masking. If the bank doesn't exist, an error will occur. This instruction can be used to create an Undo-buffer to palette handling systems.

=PAL COUNT

=Pal Count(bank number)

Returns the number of the palettes located in the bank. If the bank is empty a value of zero is returned.

This function is very useful, if your programme handles somebody else's palette banks. If you load a bank like that, first check the number of the palettes in it. If you don't check and try to access a palette which doesn't exist, an error will occur.

<u>Colours in Banks</u>

SET BANK COLOUR

Set Bank Colour bank number,place,index,colour

Changes a colour located in a bank to the colour value given. Note that if you omit the parameter place, the instruction affects the first palette in the bank.

=BANK COLOUR

=Bank Colour(bank number,place,index)

Returns the colour located in the bank. If there are no colour available, a value of -1 is returned (please refer to the next instruction).

DEL BANK COLOUR

Del Bank Colour bank number,place,index

Deletes a colour located in the bank "bank number". If you use PAL FROM BANK or PAL SWAP BANK, the colour index won't be changed, if it's representative has been deleted.

Colour Handling

SET RED/GREEN/BLUE

Set Red colour,value
Set Green colour,value
Set Blue colour,value

... changes the red/green/blue (the infamous RGB trio) component of given colour index to the setting in "value". The instructions are very handy, if you're making an image processor with AMOS. You don't have to calculate the values of red/green/blue, but can simply use these instructions. You can alter the amounts of red/green/blue in the colour palette of the picture. You don't have to worry whether the value is too big or too small because these instructions automatically convert them (x>15 => x=15 and x<0 => x4). Try this example, which adds the value of the red component by 20 percent:

```
Flash Off
For I=1 To 15
  Ink I
  Bar 20*1,0 To 20*I+20,20
Next
Wait Key
For I=0 To 15
  Set Red I,Int(Pal Red(I)*1.2) : Rem Change the red component
Next
```

=PAL RED/GREEN/BLUE

=Pal Red(colour)
=Pal Green(colour)
=Pal Blue(colour)

These functions return the value of red, green or blue of the colour given. If the colour is positive or zero it refers to a colour register (0-63). But if it's negative, the function returns a value which is calculated by taking the current component out of the absolute value of the parameter, which is considered a colour value. If you're still unsure about the parameters, go to the Direct Moa and try the next example:

```
Print Pal Red(-$f80)
Print Pal Green(-$f80)
Print Pal Blue(-$f80)
Print Hex$(Colour(1))
Print Pal Red(1)
Print Pal Green(1)
Print Pal.Blue(1)
```

PAL COPY

Pal Copy colourl To colour2

This instruction copies the value of "colourl" to "colour2". Note that the instruction should not be used with flashing colours. When AMOS flashes a colour, it changes the value of the colour register frequently, slower or faster. All the palette instructions and functions of CRAFT read the current value of the colour register, and can't store the information given to AMOS by instruction FLASH. So when you get a palette out of a bank, the colours which were flashing when you saved them in the bank, aren't flash anymore.

PAL SWAP

Pal Swap colourl ,colour2

... swaps the values of the colour registers "colourl" and "colour2". If "colourl" was black and "colour2" was blue, after this instruction "colourl" would be blue and "colour2" black. Please remember that colours which flash cause unexpected results when used in conjunction with this instruction. Try this in the Direct Mode:

```
Pal Swap 0,1
```

PAL SPREAD

Pal Spread colourl To colour2

This instruction does the same trick as the button [Spread] in the art packages. The colour next to "colourl" is transformed so that it looks a little bit like "colour2", as in the Deluxe Paint spread function. The next colour is transformed and it looks a little bit more like "colour2" than the previous colour etc. When using this instruction, please keep in mind that flashing colours will cause some funny effects. Example:

```
Curs Off : Flash Off : Cls 0 : Rem Set up the display
Colour 15,$FOF
For I=1 To 15
  Ink I
  Bar 20*1,0 To 20*I+20,20 : Rem Draw some boxes to see the effect
Next
Wait Key
Pal Spread 0 To 15 : Rem Spread the palette
```

=GR INK/BACK/BORDER

=Gr Ink
=Gr Back
=Gr Border

These functions return the value of the current ink/back/border colour set with the INK instruction. Try the next example:

```
Print Gr Ink,Gr Back,Gr Border
Ink 1,2,3
Print Gr Ink,Gr Back,Gr Border
```

CRAFT Palette Accessory

The CRAFT Palette accessory is a very good example of using these instructions in action. Because it is an accessory, it is not commented, but the main idea of these instructions should become very clear, if you have a little bit of patience. You can also feel free to alter the programme if you wish to. Note that this programme can be used only with AMOS Pro. If you want to resize your banks or delete and insert palettes, you can use appropriate programs supplied with the disc CRAFT_Extras. Because this programme is an accessory, you can put it to the User menu and start it with a simple click of the mouse button. If you don't know how to install a new programme to the menu, you should refer to the page 04.01.25 of the AMOS Pro User Manual. There is no check of command line, so there is no need to put something to it either. Select only Fl. F2 and F3 should be left unticked.

When you start the programme, it creates a display with a lot of buttons and three sliders. There are colours which can be selected in the lower part of the screen. The three sliders are used to change the colours. The values of the sliders can be seen in the right-hand side of the slider. There are two numbers just below these; the first is the current colour register (0-31) and the second is the maximum number of colours. Note that the current colour is counted from zero to max. 31 and the maximum is counted from 2 to max. 64. The programme can handle multiple palette banks, too. In the upper right corner there are two numbers, the first one represents the current palette and the second one the maximum numbers of palettes available. You can change the current palette with the large buttons [+] and [-]. To add palettes, press button [Insert] and the programme inserts a palette AFTER the current palette. If you want to get rid of a palette, press button [Delete] and the programme destroys the current palettes if there are more palettes available. Palettes can be swapped or copied. Simply click button [P.Swap] or [P.Copy] and select the another palette number with the requester. The accessory copies the current palette to the palette selected or swaps the palettes. So if you want to insert a palette BEFORE the current palette, first insert a palette after the current palette and then swap the palettes. You can spread colours by selecting the first colour, pressing the button [Spread] and then selecting the other colour. The same technique is used when you want to swap or copy colours. You can delete colours too by doing the same as when spreading, but pressing [Delete] instead of [Spread]. All the deleted colours are displayed with a mask. If you'd make a mistake, the latest version of the palette can be returned by a simple call of [Undo]. Note that the undo buffer is updated only when you press buttons [Swap], [Copy], [Spread], [Undo] or change a colour with the sliders. Note that the button [Delete] which alters the the current palette is the lower [Delete] button. The number of colours (2-64) can be changed by pressing the little [+] or [-]. When the accessory is in the EHB (Extra Half Brite) mode, the colours from 32 to 63 are shown under the colours from 0 to 31. If you try to select one of them, the programme will automatically select the brighter version of it. If you are not familiar with the EHB mode, you should consult your original AMOS User Manual.

We have not mentioned the best thing in this programme yet. You can move the cursor up and down in the current programme window and by pressing button [Command] you can insert a PALETTE instruction, which includes all the colours, which are displayed in the lower part of the screen, to the current programme. This will help you to transfer palettes from "spacked" screens to e.g. FADE instruction. Note that this works only when the accessory is run when another programme is being edited. You can move the cursor using the keyboard, but mouse and marks are fully ignored. All the procedures can be opened by pressing the [Unfold] button.

When you are running the programme as an accessory, you can also grab a bank from the current program by pressing the button [Grab] and typing the number of the appropriate bank. Note that this bank is not merged but it replaces the previous bank. When you want to quit the programme, it asks you, wether the grabbed bank should be updated or not. If you press [Ok], the accessory transfers the current palettes to the grabbed bank, but if you press [Cancel], the grabbed bank will remain as it was before the operation. You can grab a palette of an 1FF picture, too. Simply press the button [Get IFF] and select the picture with the file selector.

You can save or load the bank by pressing the appropriate button. When the standard file selector appears, just type or select the name of the file, which you want to save or load. If you press button [Merge], the palette accessory loads the file selected and merges it to the current bank after the current place. All the palettes after the current place are relocated and can be found following the merged palettes. The button [Erase] destroys all the palettes and sets the maximum number of palettes to one. If you want to quit the programme, press the button [X] in the upper left corner or the button [Quit].

08: REQUESTERS

=SYS REQUEST

=Sys Request(t1R,t2$[43$[,t4$[45$]]]],pos$,neg$)

This function creates a requester, which requests (as a good requester should do!) something from the user.

The variables tl$ to t5$ contain the text to be displayed in the requester. The contents of pos$ is displayed in the leftmost button (the positive button) and neg$ in the rightmost button (the negative button). If you replace pos$ and neg$ with empty strings "", the requester routine uses default buttons (Retry and Cancel). If you replace only the neg$ with empty string, the requester will have only one button. Note that this function uses the standard AMOS requester, and it may have some difficulties when there is a lot of text to be displayed in the requester. If the requester gets messy, alter the strings and try again!

Another important point is that the Request.Lib for AMOS 1.3 has a bug and this function displays the requester on the WB screen when using Kickstart 2.0 (surely also with KS 3.0). To avoid some problems, use instruction AMOS TO BACK before calling this function and after the user has pressed a button, call AMOS TO FRONT. If you are using AMOS Professional, there are no problems.

The function returns -1, if the user has pressed the positive button or [RETURN], or 0, if the user pressed the negative button or [ESC]. Please don't swap the buttons so that the positive button becomes the negative button and vice versa. This would only confuse the user, and nobody wants that, eh?

```
Input "Input your name ";A$
B$="I'm your own system request"
C$="Feel free to call me when"
D$="ever you want to."
A=Sys Request("Hello "+A$," ",B$,C$,D$," BYE "," BYE BYE ")
Cls
If A
  Print "BYE"
Else
  Print "BYE BYE"
End If
```

Note that the text in a line in the requester can't be very long, and if you want to make the request look a little bit better, add spaces to both ends of strings pos$ and neg$.

=GURU ALERT

=Guru Alert(t1$[32$[43$[,t4S[45$]]1])

This function creates a red alert box. The strings tl$, t2$, t3$, t4$ and t5$ (t2$ to t5$ are optional) are displayed in the alert box. Note that the alert box is a requester, if the user presses the right mouse

button, the function returns a zero (False), but if the user presses the left button, the function returns a value of -1 (True). So you have to tell the user that he has to press the left or the right mouse button depending on his choice.

Always remember that the alert system is a very drastic action. It stops multitasking and Amiga doesn't do anything before the user presses a mouse button. The alerts are to be used only in emergency situations: a fatal error has occurred or the system is about to crash, and the user is required to do something immediately after the request.

09: AUDIO

This chapter contains the definitions of the instructions how to control the Sound-/Noise-/Protracker music - modules. You may be asking yourself why we have implemented these instructions, as AMOS and AMOS Pro have their own music systems and they can play these modules as well. Because there is no converter available in any AMOS package nowadays, you can't use AMOS music system, but you have to use the Tracker instructions instead. But you can't pause the music with them, or you can't use them on games as samples can't be played simultaneously with music. With these instructions you don't need any converters and you can switch channels off and pause the music when you want to. This may sound difficult but when you have become familiar with these instructions, you'll notice that it's quite simple after all.

Once you have used these instructions, you surely won't be using any other music systems anymore. In fact, you can't use them while the CRAFT Music System is playing a module.

Note that these instructions and functions are included in another extension, not with the other CRAFT instructions. We did it because otherwise the compiler would have added the module playing system (over four kilobytes) to the compiled program whether the Audio instructions were used or not.

Playing a Module

ST LOAD

St Load file$,bank number

Loads a Sound-/Noise-/Protracker module file$ to a bank. This module can be played with the ST PLAY instruction. Note that currently this system supports new commands presented in Protracker 2.1, and the modules which are composed with earlier versions of Sound-/Noisetracker may not work with these instructions. Another thing that you should know, is that the music played with the ST instructions aren't compatible with older AMOS music and sample instructions. If you want to use the sample instructions, you have to switch some channels off or pause the music. This may sound like a disadvantage, but it's not; you can manually select the delays when stopping music and playing samples.

Note that these instructions can't handle modules which are packed in some way. If you use Protracker and pack your modules to save disc space, unpack them before trying to play them with CRAFT music instructions. If you want to you may use the AMOS instruction TRACK LOAD instead of this instruction. We induced this instruction because TRACK LOAD is not available in the earlier AMOS 1.3x versions.

ST PLAY

St Play bank number[,position]

This instruction plays a module installed in a bank. If the optional parameter is included, the instructions starts to play the module from the position defined. This way you can have several pieces of music in the same module, and the advantage is that they can use the same samples.

ST STOP

St Stop

... stops the music started with the instruction ST PLAY. When you stop the music with this instruction, you can't restart it from the position in which you stop it, but you have to start it all over again. Please refer to the next instruction.

ST PAUSE ON/OFF

St Pause On
St Pause Off

These instructions pause/unpause the current module. If you want to play some samples in between the music, you can pause it, play the samples and unpause the music again, or use the instruction ST VOICE (see below).

Switching Channels On and Off

ST **VOICE**

St Voice bit-mask

This instruction works like the normal AMOS VOICE instruction; it switches the audio channels on and off. If a bit is set to -1 in the parameter bit-mask, the channel is active and if a bit is set to 0, the channel is not active.

If you use samples with the music system of CRAFT, you have to deactivate some channels, play the sample with them and then activate them again. Note, that if you cut an instrument of a module with this instruction, the instrument won't be restarted after you have turned the channel on. The Music System waits until it reaches a new note. This is due to the internal structure of the Amiga music hardware.

=ST CHANNEL

=St Chan nel(channel)

The function returns the status whether a channel is in use by CRAFT (-1 = True) or not (0 = False). Please refer to the instruction ST VOICE.

Creating Your Own Equalizers

ST VUMETER SPEED

St Vumeter Speed amount

Sets the decreasing speed of the vumeters of the current module. The CRAFT Music System uses the default AMOS function =VU METER, but we have added a new feature to it. If the speed is set to zero, the function =VUMETER works normally. This is the default setting. When you use a non-zero value, it'll be subtracted from the current value. This process is stopped when the value of the =VUMETER reaches zero. The instruction affects also the AMAL function =VU. This means finely dancing graphics equalizers can be achieved without complex mathematical calculations (we did them for you ...)!

System Information

=ST BASE

=St Base

Returns the address of the internal data zone of the CRAFT module playing system.

=ST VERSION

=St Version

Returns the current version number of the MusiCRAFT extension multiplied with 100. Note, that the version may be different than the version of CRAFT as music instructions are in their own extension.

10: FRACTALS

If you don't know anything or only little about fractals, read this chapter. It's by no means a thorough explanation, so you might want to acquire some further information, which can be found in a number of articles and textbooks. There are both very mathematical and very practical books to choose from.

What is a fractal? To put it simple, it's a mathematical creature which has got an infinite number of details. The classic example is a coastline; the more exactly its length is measured, the bigger the result is because the smaller and smaller details are measured, eventually even grains of sand could be rounded. Fractals can be used for example to model trees and bushes on virtual reality systems running in very, very fast machines. But because these kinds of fractals are not very beautiful and because they are hard to create, the extension can calculate two basic fractals - Julia and Mandelbrot sets. They both lie near the origin of a geometric system of coordinates. To draw the sets you have to apply some complex calculations to each pair of co-ordinates to determine whether a particular point belongs to the fractal set or not. Then the number of calculation cycles (iterations) needed to make the decision is converted to a colour number and plotted onto the screen. So an iteration is an operation that consists of several mathematical calculations. By repeating iterations some interesting results can be achieved. The difference between the Julia and Mandelbrot sets is that the shape of the Julia set depends on a constant value, which is used in the calculations. So there is an infinite number of Julia sets, although most of the Julia sets are very similar. The Mandelbrot set is always the same, but nevertheless, there's a great variety of details to be found, especially if zoomed in the edge of the fractal.

Fractal calculations are like 3D calculations: they would, as such, take several hours. CRAFT uses sonic sophisticated and very optimized machine language routines to do the necessary calculations. Unfortunately, some possible shortcuts had to be sacrificed to make the system more flexible, but nevertheless it is one of the fastest Julia & Mandelbrot programmes ever made on Amiga.

<u>Fractal Definitions</u>

FR POSITION

Fr Position x,y

This instruction specifies the geometric co-ordinates of the upper lefthand corner of the fractal window. To get the fractal system working faster and more efficiently, the co-ordinates are multiplied by 8192. This co-ordinate system has been invented by fellow Finn Risto Paasivirta.

The Mandelbrot set lies in the area $-2.0 < x < 1.1$, $-1.1 < y < 1.1$.

The Julia sets lie approximately in the same area, but the co-ordinate origin is always in the middle of the shape. So you can get an acceptable overview of a set by positioning the origin in the middle of the screen and specifying an area large enough to display the whole set. The following line sets the display area to acceptably cover any set in a 320*256 pixel sized screen. Note, that the co-ordinates are equal to -160*100, 128*100 where 100 is the step value (see FR STEP), 160 is half the width and 128 half the height of the screen. This way the origin is in the centre of the screen.

```
Fr Position -16000,12800
Fr Step 100
```

=FR X/Y POSITION

=Fr X Position
=Fr Y Position

These functions return the X/Y co-ordinates of the upper lefthand corner of the fractal window. Have a look at the previous instruction.

FR STEP

Fr Step xystep
Fr Step [xstep],[ystep]

This instruction specifies the scale of the picture, or the fractal co-ordinate step between two pixels. The unit used is 1/8192, as in the FR POSITION instruction. The bigger the step value is, the bigger the area of the fractal co-ordinate system the picture drawn covers, and the smaller the details get. A different step can be specified for both x and y co-ordinates, which allows you to change the proportions of the picture. This is very handy in Hires and Interlace modes. If you specify too big a step value, you might get lots of little fractal sets as the co-ordinate variables overflow. Legal step values are in the range 1 - 1024.

=FR X/Y STEP

=Fr X Step
=Fr Y Step

These functions return the fractal step between two pixels. See the previous instruction for details.

FR WINDOW

Fr Window screen
Fr Window lscre,en,Jx,y,width,height

With this instruction you can specify any AMOS screen area to be used for fractal pictures. The "x", "y", "width" and "height" parameters are not checked until a FR JULIA or FR MANDELBROT instruction is issued. You don't have to specify every parameter, but the commas must remain. An omitted parameter is thought to mean 'as big as possible with the current screen size'. If the last four parameters are not specified at all, the window is set to cover the whole screen. For example, FR WINDOW 1,,10,, is the same as FR WINDOW 1,0,10,320,190 if the screen number 1 happens to be sized 320x200.

The screen area used for fractal drawing can also be defined with the Clip instruction, but there is a big difference between these two methods: The fractal position defined (FR POSITION) is always the point in the upper left-hand corner of the fractal window, no matter where the "clipping" rectangle happens to be. And, of course, you can't use the CLIP instruction to make the fractal drawing area bigger than the fractal window.

You should note that the fractal instructions are not compatible with the AMOS autoback system. To draw fractals into several bitmaps, you have either to use many fractal drawing instructions, which is quite slow, or to copy the area drawn to the other bitmaps.

Advanced Instructions

FR COLOUR

Fr Colour index,colour

The colours which you see in a computer-generated fractal drawing represent the number of iterations (calculation cycles) needed to determine whether a particular point belongs to the fractal set, or not. With this instruction you can set the colours used.

The index is the number of the iteration for which you are going to set the colour. Valid index numbers are 0 - 1024. As the iteration count starts from one, the index number zero has a special meaning. It specifies the colour used when a point appears to belong to the fractal set.

The colour parameter specifies the colour number to be used. It is valid in the range 0 - 255, but AMOS can only use up to 64 of them in **EHB** mode. The colour numbers may be bigger than the screen mode would allow, because in such cases only the lower bits of the numbers are used. However, if you specify an iteration count that's lower than the number of colours in the screen, not all the screen colours can be used at a time, but you can freely define the colours used with this instruction. When you use this instruction for the first time, 1025 bytes of memory is allocated for a fractal colour table. The table is also allocated if you use the fractal drawing instructions without specifying any colours. At the time of the allocation, the fractal colour table is filled with the default settings which can be restored with the following lines of AMOS code, too:

```
For A=0 To 1024
  Fr Colour A,(A and 255)
Next A
```

=FR GET COLOUR

=Fr Get Colour(index)

Returns the colour assigned to the index given. See the previous instruction for a deeper explanation and details.

FR SCAN

Fr Scan startline[,heightl
Fr Scan All

A complete fractal picture may be quite slow to draw. If you want to perform some other operations simultaneously while drawing a fractal, you can use this instruction to draw only some horizontal lines of the fractal picture at once.

The startline parameter specifies the number of the first line to be drawn. The numbering begins from the top line of the fractal window, which is line number zero. If you want to draw more than one line, you can specify any number of lines up to 16383. Only the lines inside the fractal window will be drawn, whatever lines you had specified. It's possible that nothing is drawn to the screen. With the CLIP instruction you can restrict the area to be drawn horizontally, too.

The FR SCAN ALL instruction is used to cancel any FR SCAN settings done. It is identical to the instruction FR SCAN 0,16383. The scan area is always reset after a fractal drawing instruction (FR JULIA or FR MANDELBROT).

Drawing a Fractal

FR JULIA

Fr Julia cr,ci,iterations

This instruction draws a Julia set. The set is different for every complex number "C" (cr and ci) value. The real and imaginary parts of the complex number "C" are specified in the same way as the fractal co-ordinates, e.g. by multiplying their real values by 8192. They are valid in the range from -32768 to 32767.

You must also specify the maximum number of iterations used. The more iterations you use, the longer the calculations may take. The number of the iterations also determines the maximum number of the colours possible, which is one more than the iteration count. You can change the colour' settings with the FR COLOUR instruction.

Note that if the drawing of an image is very slow, and you don't like the image, you can always stop the drawing by pressing [CTRL]+[C]. This also applies to the FR MANDELBROT instruction.

Try the next example to produce random fractals. Random fractals don't usually look very good, but sometimes you may find a real masterpiece of the psychedelic art.

```
Randomize Timer
Screen Open 0,320,200,32,Lowres
Curs Off : Flash Off : Cls 0
Fr Position -16000,12800
Fr Step 100
Fr Window 0
Fr Julia Rnd(65535)-32768,Rnd(65535)-32768,100
```

FR MANDELBROT

Fr Mandelbrot iterations

This instruction draws the Mandelbrot set. You cannot affect the shape of the set, but otherwise the instruction is identical to the FR JULIA instruction. If you think that the Mandelbrot fractal looks very boring, you should zoom to the edge of the picture. If you try different values and you are patient, you'll see some fantastic pictures.

FR RESET

Fr Reset

... resets a fractal's settings and frees the memory occupied by the fractal colour table. This instruction is automatically executed when an AMOS programme is run or when a DEFAULT instruction is issued.

11: TURTLE

If somebody still remembers the good old programming language LOGO, he will definitely also remember the fancy turtle graphics. But if you don't know anything about LOGO, a brief explanation is needed: When you use the turtle graphics, you control an invisible turtle (LOGO showed it, but due to the structure of BASIC, it can't be displayed in AMOS). You can move it forward and backward, and turn it clockwise (right) and anti-clockwise (left). This is the main point where the turtle graphics differs from the normal graphics. When you draw with the turtle, always remember that it points somewhere, and moves either towards it or away from it.

The turtle draws a line with the current INK colour when it moves. Of course you can change this with a special instruction. When the turtle draws, its pen is down and when it doesn't, the pen is up. Simple, isn't it? If you are uncertain how to use the turtle after reading this manual, you should take a look at the example programmes supplied. When you have looked at them all, you'll find out that the turtle is a very good way to draw some geometric shapes, which are very hard to draw with normal AMOS instructions.

The CRAFT turtle is more than compatible with LOGO turtle, it's very fast and precise. This is achieved by using very sophisticated fixed point fraction arithmetics. The extension doesn't use any trigonometric functions available in Amiga math libraries, but we use the trigonometric formulas with floating point numbers scaled to integers. This system gives the extra speed and no rounding errors occur. The formulas used are:

```
sinx = x-x"3/(3*2*1)+,05/(5*4*3*2*1)-x"7/(7*6*5*4*3*2*1)
cosx = 1-x"2/(2*1)+304/(4*3*2*1)+x^6/(6*5*4*3*2*1)
asinx= x+x"3/(2*3)+ 1*3*05/(2*4*5)+1*3*5*07/(2*4*6*7)
```

If these formulas look a little bit messy, you should see the notes written by the coder of the extension, Hannu Rummukainen. We found these formulas in a book which all the students who study in Finnish college, lukio, use when they don't remember the exact form of an formula. But there was a typing mistake in the Arc sin formula, and he had to correct the formula with the help of a calculator, a pencil and a few sheets of paper. In fact, these formulas are endless, but those forms are accurate enough for the turtle. The difference between these forms and the forms with the next cycle is very small, but the next cycle takes a lot of time to calculate. The extension also uses Newton's algorithm when calculating the square root in =TR DISTANCE and TR TOWARDS routines. The algorithm is a little bit too complex to be shown here. But now it's time to introduce the turtle instructions and functions:

Moving the Turtle

TR FORWARD/FORW

Tr Forward d
Tr Forw d

... moves the turtle forward by the number of units given in "d", measured in pixels. The distance may be from -65535 to 65535 inclusive. The TR FORW is simply a shortened form of TR FORWARD. Note that if you have used the TR PEN UP instruction, the turtle won't draw while moving. This applies to all the instructions except TR MOVE and TR DRAW.

The unit which is used when moving the turtle is a pixel. You don't have to use any special functions to convert the turtle co-ordinates to graphic co-ordinates and vice versa.

TR BACK

Tr Back d

This instruction moves the turtle "d" units back. TR BACK is the opposite of the TR FORW instruction.

=TR X/Y POS

=Tr X Pos
=Tr Y Pos

... returns the co-ordinates of the turtle. These co-ordinates are not the same ones as the graphic **coordinates returned by functions =X GR and =Y GR.**

TR MOVE (REL)

Tr Move x,y
Tr Move Rel dx,dy

... moves the turtle to the point "x,y" without altering the angle. Another version of this instruction moves the turtle "dx" pixels to the right and "dy" pixels to down. Either the parameter may be omitted, just remember to write the comma.

TR DRAW (REL)

Tr Draw x,y
Tr Draw Rel x,y

These instructions are almost the same as above, but the turtle draws a line while moving. Try the next example which draws the letter 'V':

```
Tr Back 25
Tr Draw Rel 12,25
Tr Draw Re 12,-25
Tr Forw 25
```

<u>Changing the Angle of the Twtle</u>

TR ANGLE

Tr Angle angle

... sets the absolute angle of the turtle. When the turtle points up, the angle is zero and when it points down, the angle is 180. Remember that all the angles are measured in degrees. The angle can be any AMOS integer, but there is no difference between using 180 or 540 (360+180).

If you want to use radians, you have to convert them to degrees before using these instructions. Remember that one degree is PI#/180.0 radians and one radian is 180/PI# (about 57.30 degrees).

=TR GET ANGLE

=Tr Get Angle

... returns the current absolute angle of the turtle. The result is always in the range -179 to 180 inclusive. See the previous instruction.

TR RIGHT/LEFT

Tr Right angle
Tr Left angle

... turns the turtle to the right/left. Note, that these instruction don't act like TR ANGLE, but they are relative to the turtle's current angle. If the turtle's current absolute angle is 50 degrees, and you use the TR RIGHT instruction with a parameter of 45, the new absolute angle is 95 degrees.

TR TOWARDS

Tr Towards x,y

This instruction turns the turtle to face the point "x,y". If you want to know the distance between the current turtle position and the pixel "x,y", use the following function, TR DISTANCE.

Advanced Instructions

TR DISTANCE

=Tr Distance(x,y)

Returns the distance between the current position of the turtle and point "x,y" in pixels. Note, that it's easier and faster to use TR DRAW (see above) than TR FORW in conjunction with TR DISTANCE. The next example shows how much easier it is to use CRAFT instructions to calculate e.g the length of the line from (0,0) to (320,200):

```
CRAFT: Tr Move 0,0 : Print Tr Distance(320,200)
AMOS: Print Int(Sqr((320^2)+(200^2)))
```

TR PEN UP/DOWN

Tr Pen Up
Tr Pen Down

... sets the turtle draw mode on/off. These instructions are used in conjunction with the TR FORW and TR BACK instructions. The turtle uses the current drawing colour set with instruction INK.

=TR PEN STATE

=Tr Pen State

The function =TR PEN STATE returns a value of -1 (true), if the pen of the turtle is down and zero (false), if the pen is up.

Remember & Memorize

TR REMEMBER X/Y/A

Tr Remember X
Tr Remember Y
Tr Remember A

These instructions savt, the current X/Y position or the angle of the turtle.

TR MEMORIZE X/Y/A

Tr Memorize **X**
Tr Memorize Y
Tr Memorize A

These instructions set a new X/Y position or angle (A), which are saved with the TR REMEMBER X/Y/A instructions.

<u>The Home of the Turtle</u>

TR HOME

Tr Home

... returns the turtle to its home. This instruction affects the angle of the turtle. When a TR RESET instruction is issued, the home is reset (see the following instruction) and the turtle is returned there. If you don't want to alter the angle when moving turtle to its home, use this instruction conjunction with instruction TR REMEMBER A.

TR SET HOME

Tr Set Home x,y

... sets a new position of the home of the turtle. The co-ordinates of the home are in the middle of the screen as default. You can use the home of the turtle in conjunction with the instructions TR REMEMBER and TR MEMORIZE to have two points where you can always jump with TCL (see TR EXEC).

.TR X/Y HOME

.-**Tr X** Home
=Tr Y Home

... returns the co-ordinates of the home of the turtle. See the previous instruction (TR SET HOME) for details. These functions used conjunction with the next instruction gives you another way to move the turtle to its home without altering the angle:

Tr Move Tr X Home,Tr Y Home

TR EXEC

Tr Exec command$[,count]

This instruction executes a Turtle Control Language (TCL) string. If the count is specified, the string is executed several (0-2000) times. This stands for the Repeat instruction found in LOGO. Only capital letters are necessary in command names, and semicolons are used to separate the instructions. For example, "M120,100;A90;PU;F20;PD;B10" is the same as "Move 120,100;. Angle 90; PenUp; Forward 20; PenDown; Back 10". Parameters may be omitted from almost all the instructions having another parameter. This applies also to the AMOS TR instructions. The following instructions are available in TCL:

A - Angle a	Sets the new absolute angle.
B - Back d	Moves the turtle "d" steps backward.
D - Draw x,y	Draws a line from the current turtle position t o co-ordinates "x,y". The angle is left unchanged, but the new co-ordinates of the turtle are set to "x,y".
DR - DrawRel x,y	Draws a line "dx" pixels right and "dy" pixels left from the current position.
F - Forward d	Moves the turtle "d" steps forward.
H - Home	Moves the turtle to its home.
I - Ink c[,b]	This instruction is a synonym to the Pen instruction.
L - Left a	Turns the turtle left "a" degrees.
M - Move x,y	Moves the turtle to the point "x,y" without drawing a line.
MA - MemAngle	Loads the angle saved with RA.
MX - MemX	Loads the "x" coordinate saved with RX.
MY - MemY	Loads the "y" coordinate saved with RY.
MR - MoveRel x,y	Moves the turtle "dx" pixels right and "dy" pixels down.
P - Pen c[,b]	Sets the ink colour to "c" and the pattern colour to "b". You don't have to care about the parameter "b" unless you play with the SET LINE instruction.
PD - PenDown	Lower the pen of the turtle down.
PU - PenUp	Lifts the pen of the turtle up.
R - Right a	Turns the turtle right a degrees.
RA - RemA	Saves the angle of the turtle.
RX - RemX	Saves the current "x" co-ordinate of the turtle.
RY - RemY	Saves the current "y" co-ordinate of the turtle.
SH - SetHome x,y	Sets the co-ordinates of the home of the turtle.
TO - TOwards x,y	Turns the turtle to face the point "x,y".

=TR ERROR

=Tr Error

This function returns the position of the instruction that caused an error in a turtle string. If there were no errors, a value of zero is given. Note that this function is not reset when it's read, but you can read the value several times without consequent TR EXEC calls. Try the next example in the Direct Mode:

12: STRINGS

There are a lot of instruction which are designed to replace a bunch of lines of complex string code in this chapter. Some of them are used frequently, but there also are some functions which are used seldomly. But when you use them, you're grateful that there are functions like these, and you don't have to code hours and hours to create the same effect as the function.

Character Conversions

=UP/LO/FLIP CASE$

=Up Case$(string$)
=Lo Case$(string$)
=Flip Case$(string$)

The functions =UP CASES and =LO CASE$ are almost the same as standard AMOS functions t--UPPER$ and =LOWER$, but they can convert all the special characters, too. These characters include &, ii, a, o, a, c, é, n etc. Example:

```
Print Upper$("Francois Lionet") Result: FRANcOIS LIONET
Print Up Case$("Francois Lionet") Result: FRANCOIS LIONET
```

The function =FLIP CASES changes the case of the letters of the string given from upper to lower and vice versa. You can try this second example, but if you don't live in Germany, you may need to use CRAFT ASCII accessory (For more details, see the function =CHR CONV$):

```
A$="Wir miissen die GroBstadter zerstoren"
Print Upper$(A$)
Print Up Case$(A$)
Print Flip Case$(A$)
```

=CHR CONV$

=Chr Conv$(string$,chrl To chr2)

This function converts all the characters whose ASCII code is "chrl" to the character whose ASCII code is "chr2". All the other characters are left untouched. Example:

```
Print Chr Conv$("AMOS BASIC",65 To 90) Result: ZMOS BZSIC
```

If you are unsure about the ASCII code of a character, you can always use CRAFT ASCII accessory. The accessory shows all the characters and their ASCII codes (in hexadecimal). The first 32 characters are not shown because there are no visible characters specified. There are two versions of the programme for both AMOS and AMOS Professional.

The AMOS version of the programme is extremely easy to use, simply load it, look for the codes you need and press the mouse button. The AMOS Pro version is more sophisticated: You can put it in the User Menu or simply load it from disc with Open&Load. If you have started the programme as an accessory, you can add special characters to your programme. Simply select a character with the mouse and then press the button [Put char]. The highlighted character will be placed in your programme in the current place of the cursor. If you want to move the cursor, you can only use the keyboard, as marks and mouse are totally ignored. Remember that you shouldn't add characters which have ASCII codes lower than 32 to your programme. If you need them in a string, use function =CHR$ instead. We have included another useful accessory in the packet. It has nothing to do with ASCII codes, but it shows all the raw codes assigned to various keys. If you don't know what the raw codes are, please refer to AMOS function =SCANCODE. The programme is very easy to use. Simply load it from disc and run it. If you use AMOS Pro, you can put it in the User Menu. When you start the programme, it shows a table full of raw codes. Simply find out the needed ones, and then get rid of the programme by pressing the closing button in the upper left corner (in AMOS Pro version) or mouse button (in AMOS version).

=LEFT/RIGHT TRIM$

=Left Trim$(string$ktrim$D
=Right Trim$(string$[,trim$])

These functions remove leading or trailing spaces from • the string given. The functions are surprisingly similar to the functions LTRIM$ and RTRIM$ found in the notorious MicroSoft(R,TM etc.) "Quick" Basic. Examples:

```
Print Left Trim$(" CRAFT is - ");"powerful"
Print Right Trim$(" CRAFT is - ");"powerful"
```

If you want to, you can use these functions to remove any characters for spaces, simply by adding another parameter trim$, which defines which characters are to be removed. For example:

```
Print Left Trim$("AMOS","AM")
Print Left Trim$("MAGIC","AM")
```

As you see, the functions remove all the characters without paying attention to their order, until they encounter a character which is not defined in the parameter trim$.

<u>Character Counting</u>

=STR COUNT

=Str Count(search$,string$)

This function counts how many times does the search$ occur in the string$. Note, that when this function finds a search$, it jumps to the next character after the occurrence. This means that the function doesn't count any occurrences which starts inside another occurrence. For example:

```
Print Str Count("ABA","ABABABA") Result: 2
```

=BW INSTR

=Bw Instr(s$,f$[,p])

This function works like the standard function =INSTR(s$,f$[,p]), except that this function searches backwards (bw), from the end towards the beginning of the string s$. The optional parameter "p" tells the instruction where to start hunting counted on the left end of the string. Note that the function does not accept an occurrence of f$ if it would extend past the position "p".

```
Print Bw Instr("AMOS CRAFT","A")
Print Bw Instr("AMOS CRAFT","A",3)
```

Scrambling and I Jnsrrambling

=STR SCRAMBLES/UNSCR A MBLE$

=Str Scramble$(string$,password$)
=Str Unscramble$(string$,password$)

These functions scramble/unscramble string$ using the password given. The string password$ doesn't have to be a single character, but can be numbers, words, a part of a song etc. Note that scrambling is by no means a proper protection against professional information thieves, but it protects your data and text against "friends", users of your programmes etc. But anyone may be a professional information thief or a good guesser, thus we can't take any responsibility whatsoever.

If you want to maximise your security, don't use the same letters in your passwords and use long passwords. When you want to access your data, use the function =STR UNSCRAMBLE$ with the same password as with =STR SCRAMBLE$. Example:

```
Print SIT Scramble$("CRAFT is powerful","STR")
Print Str Scramble$(Str Unscramble$("CRAFT is powerful","STR"),"STR")
```

Note that these functions are case selective, so you have to use exactly the same password when scrambling and unscrambling:

```
Print Str Scramble$(Str Unscramble$("CRAFT is powerful","STR"),"str")
```

Memory Dumping

=HEX DUMP$

=Hex Dump$(address,length)
=Hex Dump$(address,length,sep)

This function dumps memory in hexadecimal from address to address+length. The parameter "sep" tells the function how many bytes to read before adding a space. The default value is four. If the "sep" is zero, the function adds no spaces at all. Examples:

```
Print Hex Dump$(Amos Base,80)
Print Hex Dump$(Amos Base,80,2)
Print Hex Dump$(Amos Base,80,8)
```

=CHR DUMP$

=Chr Dump$(address,length)

This function dumps memory from address to address+length in ASCII. All the characters that can't be printed (0-31 and 128-159) are converted to full stops (Chr$(46)). Try this:

```
Print Chr Dump$(Amos Base,80)
```

If you want to see the dump functions in action, please load the accessory CRAFT_Dumper.AMOS. When you start the programme, it shows you the standard file selector where you have to choose the file to be dumped. When the file has been loaded to memory, you can go to the next or previous page with the buttons [Next] and [Prey]. The button [Go To] displays a requester asking you to type a new page number. When you have typed it and pressed the button [OK], the programme jumps to that page. Note that the number of pages can be seen in the vertical bar on the upper part of the screen. The number before the slash is the curfent page and the number after the slash is the number of pages. When you want to quit the programme, simply press the button [X] in the upper left corner in AMOS Pro or the button [Quit] in AMOS.

<u>Strings and Memory</u>

STR POKE

Str Poke addiess,string$

This instruction puts the string given to the memory starting from "address". Never use this instruction to alter strings, where addresses can be attained with =VARPTR function. For example:

```
Reserve as work 1,100
Su Poke Start(1),"CRAFT means shrewdness"
Print Chr Dump$(Start(1),22)
```

=STR PEEK$

=Str Peek$(address,length[,stop$])

... returns a string, which is fetched from the address given. The "length" parameter defines the length of the string. The stop$ tells the function, which character is considered as the end of the string. Note that upper and lower cases aren't considered as equals. If you use the second form, the length of the string returned may be smaller than the parameter length. Example:

```
Tr Exec "F 100;R 90;F 100; RA 10"
Print Tr Error
```

The turtle errors can be found in Appendix A, in the end of this manual.

Setting Up the Turtle System

TR PROPORTIONS

Tr Proportions xy
Tr Proportions x,y

The parameters "x" and "y" specify coefficients for the turtle co-ordinates. The limits of the parameters are -16 to 16 inclusive, and zero is not allowed. Negative parameters may be used to create a mirror effect. If there are only one parameter, it'll be used for the both co-ordinates. Note, that this instruction affects the TR FORW, TR BACK, TR MOVE REL and TR DRAW REL instructions and their TCL counterparts. Note that improper use of this instruction may cause some odd effects on the screen, so you'd better know what you're doing. Try the next example which tries to draw a circle:

```
Tr Proportions 5,1
Tr Move 16,100
For       To 360 Step 10
  Tr Angle I
  Tr Forw 5
Next
```

TR RESET

Tr Reset

... resets the settings of the turtle. The angle is set to zero and the pen is set down. The turtle is returned to its reset home co-ordinates (see TR SET **HOME** instruction) at the default position (centre of the screen). The instruction resets also the positions stored with the TR REMEMBER instructions.

This instruction is also automatically executed when the DEFAULT instruction is issued or your programme is run.

=TR BASE

=Tr Base

Returns the address of the internal data area of the turtle. Please refer to Appendix B.

```
Get Sprite 1,0,0 To 16,16
Print Str Peek$(Start(1)-8,8)
Print Str Peek$(Start(1)-8,8,"e")
```

The difference between =STR PEEK$ and =CHR DUMP$ is that the =CHR DUMP$ converts all the characters that can't be printed to full stops.

13: MEMORY

In this chapter you find some very useful functions to check the state of the memory, a few instructions which have their string equivalents and a couple of improvements to AMOS instructions.

<u>State of Memory</u>

=CHIP/FAST MAX BLOCK

=Chip Max Block
=Fast Max Block

These functions return the size of the biggest continuous block of chip/fast memory. Sometimes there may be plenty of memory left, but it can't be used because there are no big blocks available as all the memory may be fragmented in little pieces. If your machine is multitasking, the values given by these functions may alter very quickly, so you have to be careful with the values given.

=MEM TYPE

=Mem Type address

... tells what kind of memory is located in the address given. The result is a bitmap:

```
bit# description
--------
 0 PUBLIC memory
 1 CHIP memory
 2 FAST memory
```

This function is mainly used in conjunction with banks. You can check the type of memory banks loaded from a disc, if you are not sure if they are CHIP or FAST banks. Note, that you can use the number of a bank instead of the address, and the function tells what kind of memory the bank is located there. Note, that due to the internal structure of the Object and Icon banks, the starting point of the bank can be in the fast memory even if the actual data is located in the chip memory.

The PUBLIC memory refers to the memory, which can't be moved to a hard disc when using virtual memory. This may sound a little bit weird, but you don't have to worry about, because there will be no need to use virtual memory with AMOS. Please note, that the function may return a zero, if you peek small addresses, because they are not on the Exec memory lists.

<u>Data in Memory</u>

MEM COPY

Mem Copy start,finish To destination

This is almost the same instruction as COPY, but it allows you to use addresses, which are not divided by four. This instruction is not as fast as COPY, but it will always beat a FOR...NEXT loop. The instruction copies the memory segment ranging from "start" to "finish" to "destination". Use the instruction with great care, because improper use may crash the computer. Consult your User Manual for more details, but before, try the next example:

```
Get Sprite 1,0,0 To 16,16
Get Icon 1,0,0 To 16,16
List Bank
Mem Copy Start(1)-8,Start(1)-1 To Start(2)-8
Wait Key
List Bank
```

=MEM STR COUNT

=Mem Str Count(start To end,search$)
=Mem Str Count(bank number,search$)

The first version of the function counts how many times does search$ occur in the memory area from "start" to "end". See the function =STR COUNT for more details and try this example:

```
Reserve As Data 1,100
Print Mem Str Count(Start(1)-8 To Start(1),"a")
```

The second version of the function counts the amount of occurrences of the search$ strings in the bank given.

MEM SCRAMBLE/UNSCRAMBLE

Mem Scramble start To finish,password$
Mem Scramble bank number,password$
Mem Unscramble start To finish,password$
Mem Unscramble bank number,password$

These instructions can be used to scramble and unscramble areas of memory or memory banks. See the functions =STR SCRAMBLE$/UNSCRAMBLE$.

14: DISC & FILES

In this chapter you'll find some very powerful instructions and functions to check the state of disc and files. You can for example read directories, and gain some extra information that can't be got when using AMOS functions. You can also check free or used disc space of a disc without changing your current directory. And there are a lot of instructions and functions still not mentioned, so start reading!

<u>Basic Functions for Reading Directories</u>

=DR NAME$

=Dr Name$(path$)

This function returns the name of the directory path$. You may wonder, why we made another directory reading function. It's true that this function does the same thing as =DIR FIRST$, but with this and some other functions you can get more information about the directory than with =DIR FIRST$ and =DIR NEXT$. Note that you always have to include the parameter path$. This function doesn't check the current directory of AMOS, but it uses the directory from which AMOS was booted default. Another important thing is that you can't use special characters like '*' or '?' with these functions. Don't worry, we have supplied an example programme which can do the trick for you. If you use special functions with =DR NAME$ and =DR NEXT$, you should read the following. You can use all the special functions while reading a directory, but it's not recommended to use them when you have read only the name of the directory. Also remember that all the special functions refer to the file of which the name was read with =DR NEXT$. If you use =DR NEXT$ again, the return values of special functions will be changed.

=DR NEXT$

=Dr Next$

... returns the next filename of the directory set by =DR NAME$. When all the files in the directory have been read, =DR NEXT$ gives an empty string "". If you continue reading the directory after getting an empty string, you will cause an error.

REMEMBER! This function changes all the information given by special functions. 'Read the instruction above for details. The next routine reads the current directory:

```
A$=Dr Name$("")
Print "Directory of ";A$
Print " - - - - - - - - - - - - - - - - - - - -
Do
   A$=Dr Next$
   Exit If A$=""
   Rem Special functions will be placed here
   Print A$
Loop
```

=DR LENGTH

=Dr Length

This function returns the length of the current file. It does the same trick as the =FILE LENGTH function described below. The length of a directory is always zero.

=DR TYPE

=Dr Type

... returns the type of the current object. If the result is bigger than zero, the last item examined is a directory, but if you get a number smaller than zero, the item is a file. Replace the REM statement in the example to =DR NEXT$ (above) with the next lines of code:

```
A$=A$+Space$(30-Len(a$))
If Dr Type>0
  A$=A$+" DIR"
Else
  A$=AS+StrS(Dr Length)
End If
```

The next two examples show the difference between AMOS and CRAFT directory reading functions. The left one is done without CRAFT and the right one is done with CRAFT:

```
A$=Dir First$("DHO:")
Repeat
  If Left$(A$,1)="*"
    Print Right$(AS,Len(AS)-1);
    Locate 31,
    Print "DIR"
  Else
    A$=RightS(AS,Len(A$)-1)
    PITUUS=Val(Mid$(A$,30,7))
    A$=Left$(AS,28)
    Print A$;
    Locate 30,
    Print PITUUS
  End If
  AS=Dir Next$
Until A$=""
```

```
AS=Dr Naine$("dh0:")
Do
        AS=Dr Next$
        Exit If A$=""
        Print A$;
        Locate 31,
        If Dr Type>0
                Print "DIR"
        Else
                Print Dr Length
        End If
Loop
```

The difference should be very clear. When using the original AIVIOS functions, the user has to use complex string mathematics, but when using CRAFT functions, the user can get all the information needed without doing any string exercises. Note that the AMOS example doesn't tell you the name of the directory and there's no way to get the protection bits or notes of the files listed (see the next two functions).

Special Directory Reading Functions

=DR COMMENT$

=Dr Comment$

This function returns the comment installed to the current file. Please see =FILE COMMENT$ and SET COMMENT for details.

=DR PROTECT

=Dr Protect

... returns the protection bits of the current file. Here is a little table which shows what each bit means. If you want accurate definitions, please see instruction SET PROTECT.

bit#	symbol	description
7	h	Hide - used by hard discs, normally 0.
6	s	Script - if 1, the file is a CLI/S hell batch file.
5	p	Pure - if 1, the file can be safely made resident.
4	a	Archive - if 1, the file is archived after last change.
3	r	Read - if 1, the file is not allowed to be read.
2	w	Write - if 1, the file is not allowed to be written to.
1	e	Execute - if 1, the file is not executable.
0	d	Delete - if 1, the file is not allowed to be deleted.

=DR FIB

=Dr Fib

This function gives you the address of the FileInfoBlock (FIB) of the current file. The FileInfoBlock contains more information about the file. This info can be accessed with the special functions. Because they are used very rarely, we haven't implemented them as functions. If you want to know more about the FIB, you'd better read Appendix C.

Quiting Directory Reading

DR FORGET

Dr Forget

This instruction erases all the data concerning the directory reading from memory. This instruction is not needed if you read the whole directory (until =DR NEXT$ returns an empty string ("")). But if you don't read the whole directory, and you want to start to read another, you have to use this instruction before proceeding. This instruction is automatically executed when an AMOS programme is run or a DEFAULT instruction is used. Note that if you are reading a directory, you cannot delete it before you use this instruction.

Disc Snare

=DB FREE/USED

=Db Free(drive$)
=Db Used(drive$)

These functions return the free/used amount of disc space in the drive given. Drive$ is a standard Amiga discname, like DFO:, AMOS:, sys: etc. You should keep in mind that these functions return the free/used space in blocks (usually 1 block equals 488 bytes). The Info command of the CLI shows the same numbers. If there is no disc in the drive, these functions return a value of -1.

If you are using standard floppy drives, DB FREE + DB USED is always 1758. On hard discs, floptical drives or Ram disc etc., the result may vary from that value. Example:

```
Print Db Free("DfO:")
Print Db Used("Df0:")
Print Db Free("DfO:")+Db Used("DfO:")
```

=DB SIZE

=Db Size(drive$)

This function returns the size of a block on a given drive. This is a very useful function, if the user has a hard disc, where the block size doesn't equal 488 bytes. See the previous functions for more details about blocks. Note that if there is no disc in the drive, the function returns -1. If you want to know the exact amount of the free/used space of a drive, use the following lines:

```
Print Db Free("DfO:")*Db Size("DfO:")
Print Db Used("Df0:")*Db Size("DfO:")
```

The AMOS functions =DFREE returns the same value as the upper line of code, but you have to change your directory if you want to use it.

=DISC STATE

=Disc State(drive$)

Returns the state of a disc drive. A value of -1 means that there is no (DOS) disc in the drive. A value of zero tells you that the disc is write protected. If the value is 1, the disc is not yet validated and you'd better wait some seconds and use this function again. A value of 2 indicates that the disc is validated and can be accessed without risk.

=DISC TYPE$

=Disc Type$(drive$)

... returns the identification code of the disc in a drive. The following codes are the most common ones found but, there may be other ones, too.

DOS	A normal AmigaDOS disc on the Old File System (OFS) format. If the disc is formatted with the Fast File System (FFS), the function returns "DOS"+Chr$(1). All you need to do is check the three leftmost letters of the string to see whether the disc is OK or not.
NDOS	Not an AmigaDOS disc.
BAD	Disc structure is corrupted or abnormal (for example a game disc)
KICK	A Kickstart disc for Amiga 1000. Note that Amiga 1000 is not the only Amiga that uses Kickstart discs. E.g. the early versions of Amiga 3000 need a Kickstart 2.0 disc.

If the disc drive is empty, the function returns an empty string "". If you want to check the disc in a drive, use function =DISC STATE first. If it returns -1, use this function to check the type of the disc and inform the user that there's something wrong on the disc, or there is no disc in the drive.

Protection Bits

SET PROTECT

Set Protect file$,bits

Changes the protection bits of the file. The orders and meanings of the bits are shown below. If you are not sure about logical operations and binary arithmetic, you can always use the instruction BSET and the function =BTST. Currently there are eight bits (if you use WB 1.2, there are only four bits (0-3) available):

bit#	Symbol	Description
7	h	Hide used by hard discs, normally 0. If the flag is set to zero, the file won't be shown in directory listings.
6	s	Script if 1, the file is a CLI/Shell batch file. AmigaDOS can execute this file without the command Execute.
5	p	Pure if 1, the file can be safely made resident.

		Study your AmigaDOS manual for more details.
4	a	Archive - if 1, the file is archived after the last change. This flag is used by backup programmes. If you are making one, you can check this flag to see if the file has been archived earlier or not.
3	r	Read - if 1, the file is not allowed to be read.
2	w	Write - if 1, it is not allowed to write to the file.
1	e	Execute - if 1, the file is not executable, but it's a data file.
0	d	Delete - if 1, the file is not allowed to be deleted. If you have very important files, you can set this flag to 1 to ensure that AmigaDOS won't delete them if you made a typing mistake etc.

Use this instruction with care. If you play with it, you may damage some extremely important protection bits of the system files. If you aren't familiar with the protection bits, you should read an AmigaDOS manual before using this instruction.

=FILE PROTECT

=File Protect(file$)

... returns the settings of the protection bits of the file given. The default value of all the bits is 0. Not all the bits are normally used, though. See the previous command to see the meanings of the bits. If you don't want to play with binary mathematics, use the next procedure, which shows you the letters of the bits of a file:

```
Procedure PROTECT_LETTERS[FILE$]
   If Exist(FILE$)
      X=File Protect(FILE$)
      For A=7 To 0 Step -1
         If Btst(A,X)
            AS=AS+Mid$("----apsh",A+1,1)
         Else
            A$=A$+Mid$("dewr----",A+1, 1)
         End If
      Next A
   End If
End Proc[A$]
```

The procedure gives the result in the =PARAM$ and it's similar to the system used in AmigaDOS command List.

File Comments

SET COMMENT

Set Comment file$,comment$

... assigns a comment to a file. This comment can be read with the function =FILE COMMENT$. You should notice that the maximum length of the comment is 79 characters and the comment will not be copied while copying file. If you want to get rid of the comment, simply use empty string ("") instead of comment.

=FILE COMMENT$

=File Comment$(file$)

This function returns the comment installed in a file. The same comment can be seen with the command List in CLI.

Additional features

=FILE LENGTH

=File Length(file$)

... returns the length of the file given in bytes. You can always open the file, use the =LOF function, but it's more complicated and somewhat slower than the =FILE LENGTH function. The length of a directory is always zero.

For example:

```
Print File Length("c:")
Print File Length("c:Run")
```

=FILE TYPE

=File Type(filespec$)

... returns the type of the filespec$ string. If it's a directory, a positive value is returned, and if it's a file, a negative value will be given.

=DISC ERROR

=Disc Error

When a disc error occurs, AMOS generates an error. Sometimes its very clear, and the programme knows immediately what's wrong. But sometimes AMOS returns an I/O error. This error message caters all the errors which usually don't pop up while programming with Basic. But when it occurs, the programme can't be sure about the real error. This situation can be got rid of by simply using the =DISC ERROR function. It returns the number of the latest AmigaDOS error. The next list defines the AmigaDOS 1.2 error numbers and messages (if the message isn't clear enough, there is another explanation in the brackets):

Nro Definition

103 Insufficient free store (Out of memory)
104 Task table full (Too many CLIs running)
120 Argument line invalid or too long
121 File is not an object module (File is not runnable)
122 Invalid resident library during load
202 Object in use
203 Object already exists
204 Directory not found
205 Object not found
206 Invalid window
209 Packet request type unknown (AmigaDOS can't carry out a task
requested by a programme, very rare error)
210 Invalid stream name (File name is too long or invalid)
211 Invalid object lock
212 Object not of required type
213 Disk not validated (See =DISC STATE)
214 Disk write protected
215 Rename across devices attempted (You can't rename from a drive
to another drive)
216 Directory not empty
218 Device not mounted
219 Seek error
220 Comment too big (See =FILE COMMENT$)
221 Disk full
222 File is protected against deletion (See =FILE PROTECT)
223 File is protected against writing (See =FILE PROTECT)
224 File is protected against reading (See =FILE PROTECT)
225 Not a DOS disk (See =DISC TYPE$)
226 No disk in drive
232 No more entries in directory (You have used the =DR NEXT$ after
it has returned an empty string "")

Note: Some error messages can differ from the error messages in AmigaDOS 2.0 or 3.0.

You can use this function to give your programme a nice professional touch. Simply copy all the error messages written in the AmigaDOS manual with their numbers to data clauses. Read these values into a dimensioned variable and when an error occurs, simply use this function to find out its number and print the number with the error message like this:

Error 204 - Directory not found
Error 222 - File is protected from deletion

Try the next example to see the function =DISC ERROR in use:

```
Request Off
On Error Goto ERR
Open In 1,"Wackowoo:Just.Testing"
End
ERR:
Print Disc Error
Wait Key
Resume Next
```

15: AMIGA SYSTEM

This chapter deals with Workbench and CLI. Without CRAFT you only could close Workbench, swap the AMOS and Workbench screens and, in AMOS Pro, execute CLI programmes. With CRAFT you can also open Workbench, swap it and other intuition screens to and fro, print text to a CLI window and read the WB preferences to for instance grab the colours of the Workbench for your programme. The best thing is still not mentioned: You can speed up your AMOS programmes without compiling them. Simply stop multitasking or increase the priority of AMOS.

Controlling Workbench

OPEN WORKBENCH

Open Workbench

This instruction opens the Workbench, if it's been closed earlier. You should notice that the Workbench screen needs quite a bit of CHIP-memory. If the Workbench screen has 4 colours, its width is 640 and its height is 256, the consumption of memory will be over 40 kbytes. So before using this instruction, use function =CHIP MAX BLOCK to check whether there is enough memory or not.

WB TO FRONT/BACK

Wb To Front
Wb To Back

These instructions have nothing to do with AMOS TO FRONT/BACK. If you have Workbench and a Workbench application, which uses its own screen, simultaneously in memory, you can flick those screens with these instructions. If you use the following instructions, you should always put WB TO FRONT instruction at the beginning of your programme, so that your WB messages will be visible.

Controlling CI d

CLI EXECUTE

Cli Execute command$

This instruction executes a DOS command installed in the string command$. The command can be a CLI command or a normal programme. If your AMOS has closed the CLI in which it was started, you should not execute programmes which tend to write information on the CLI window or try to get some information from the user through the CLI. The execution of the current AMOS programme is frozen until the programme that started with this instruction has completed its job. Remember to check that all the files needed by the programme 'hat are executed do also exist.

CLI PRINT

Cli Print text$

This instruction prints text to the CLI window AMOS was started from. If that window is closed or AMOS has been started from the Workbench, there will be no output. You should also notice, that this instruction doesn't automatically send a [RETURN] code after the string, but you have to add it, like this:

```
Cli Print "CRAFT v1.0 in action"+Chr$(10)
```

=CU HERE

=Cli Here

... returns a value of -1, if AMOS is started from a CLI and the user can write text to it. Example:

```
If Cli Here
   Print "This program was started from CLI"
Else
   Print "This program was started from WB"
End If
```

Note that if you start AMOS with the CLI command Run, the function returns a value of 0.

Preferences

WB (DEF) PREFS

Wb Def Prefs address,length
Wb Prefs address,length

These instructions copy the default or current system preferences to the memory address given. Instead of "address" you may use the number of a bank. The "length" parameter tells how many bytes to copy. These instructions are very handy if you want to change your programme to act according to the user's preferences choice. If you aren't sure, what you're doing, don't do it: These instructions are for advanced users only.

We have listed some useful system information that can be attained with these instructions. Please refer to Appendix C.

SET WB PREFS

Set Wb Prefs address,length,flag

... sets the system preferences. Address is either a real address or a bank number where the instruction gets the preferences data. The length parameter should be obvious. You are advised to read the preferences into an address first and then modify the bytes you want to change before setting the preferences with this instruction. This instruction, too, is for advanced use only and it should be used with extreme care. The flag parameter is used to decide whether to inform programmes on the new preferences. A non-zero value means that the preferences are sent to other programmes. A value of zero means the opposite. If you are setting new preferences, always use a non-zero value, because the flag parameter for the instruction is used to express whether the preferences set are intermediate (for example, the user is playing with WB colour gadgets in a preferences-type programme).

AMOS and Multitasking

MULTI ON/OFF

Multi On
Multi Off

... enables/disables multitasking. If you think that your programme needs all the processor time that your Amiga can give it, disable multitasking. But remember that you should ALWAYS restore the state of the machine at the end of your programme. Otherwise some nasty things may happen (the editor doesn't work, disc drives are jammed etc.). Always save your programmes which contain MULTI OFF instructions before trying to run them. Another important point is that you have to call MULTI ON as many times as you have called MULTI OFF. If there are several instructions, which try to disable multitasking in your programme, you must use a variable to keep a record how many times you have used the MULTI OFF instruction and create a loop which returns the state of the machine with continuous calls of instruction MULTI ON.

SET AMOS PRI

Set Amos Pri number

... sets the priority of AMOS. The priority tells the system how much processor (CPU) time to give AMOS in relation, to the other programmes running on your Amiga simultaniously. Parameter "number" may be any integer between -128 and 127. The bigger the number, the more time will be given and the faster your programme will run. This is a more sophisticated way to speed up the execution of your programme than the MULTI OFF instruction.

When using this instruction, you should not use very big or very small parameter values. If your priority of AMOS is bigger than for example the priority of the disk drive or keyboard, some nasty things may happen. And if you set the priority of AMOS too low, another programme may take over the control and AMOS is jammed.

=AMOS **PRI**

=Amos Pri

... returns the current priority setting. See the previous instruction.

Testing   the Mouse Buttons

=I-IW MOUSE KEY

=Hw Mouse Key

This function is an extended version of the =MOUSE KEY function. It works whether the AMOS screen is displayed or not. This is a very good thing for those programmers who want to get information from the user, when the programme is displaying something in the WorkBench screen. Letters HW stand for HardWare, which is used directly with this function. Try this example:

```
Amos Lock
Amos To Back
Repeat
Until Hw Mouse Key=3
Amos To Front
Amos Unlock
```

16: MISCELLANEOUS

In this chapter you can find instructions and functions which cannot be allocated to any headlines presented before. This doesn't mean that these instructions aren't handy or that its difficult to use them. When you look at the definition of the instructions, you'll probably say: "Why haven't these instructions been in AMOS in the first place?"

GR CENTRE

Gr Centre y,t$

This instruction is similar to the instruction TEXT x,y,t$ but it centres the text. If you're not familiar with graphic font systems, please refer to your AMOS manual. Try the next example:

```
Gr Centre 100,"CRAFT rules OK"
```

It's equivalent to the next code:

```
AWCRAFT rules OK"
X=Text Length(A$)
X=Screen Width/2-X/2
Text X,100,A$
```

Resetting the Machine

HARD/WARM **RESET**

Hard Reset
Warm Reset

When your programme encounters the HARD RESET instruction, it will stop running and the machine is reset completely, in other words this instruction does the same thing to Amiga as toggling the power off and then on again. This instruction can be used with virus killers. If the programme finds something abnormal in the memory, it can force Amiga to do a Hard Reset.

WARM RESET is a softer way to reset Amiga. It is similar to a [CTRL]+[AMIGAHAMIGM-reset. Also the reset-proof ram discs don't disappear whereas HARD RESET destroyer everything. The only negative thing in this instruction is that viruses won't be killed and they are already active when the first disc is put in.

GURU MEDITATION

Guru Meditation numberl,number2

This instruction causes a GURU. The numbers will be shown in the red alert box. There will be no return, and there are two alternatives you have with this instruction: you can either tell the user about an error or you can just startle him.

Remember when using this instruction that the new versions of the Workbench and AmigaDOS handle the errors in a different way than the older versions (1.2 and 1.3), Please study the AmigaDOS manual v2.0 or higher before using this instruction on new machines.

<u>Electron Beam</u>

Y BEAM

=Y Beam

... gives you the current y position of the electron beam which draws the display on your monitor or TV set. The result is given in hardware co-ordinates, and it ranges from 0 to 311 on PAL systems and from 0 to 261 on NTSC systems. This can be used for synchronizing your display to get a smoother and slicker result than the result achieved when using the WAIT VBL instruction.

BEAM WAIT

Beam Wait y

... waits until the beam reaches the position y. Use the lowest hardware co-ordinate that contains moving images with BEAM WAIT for the smoothest synchronising possible. Note that if the y is bigger than the number returned by =DISPLAY HEIGHT, an Illegal function call error is given. See also the previous function for more details about the beam.

<u>Changing Bits</u>

B/W/L.SWAP

=B.S wap (number)
=W. S wap (number)
=L. S wap(number)

These functions swap the upper and lower parts of a specified segment (Byte, Word or Long word). Only the bits which are specified with the first letter of the function are swapped, the others are reset to zero. It's better not to use these functions if you are not familiar with hexadecimals, logical operations and bit-rolling. Examples:

```
Print Hex$(B.Swap($12345678),8) Result: $00000087
Print Hex$(W.Swap($12345678),8) Result: $00007856
Print Hex$(L.Swap($12345678),8) Result: $56781234
```

System Information

=AMOS BASE

=Amos Base

... returns the address of the internal data zone of AMOS. This function is designed only for experienced programmers. If you want to learn something about the internal data zone of AMOS, please consult the machine code sources included with AMOS.

=AMOS PRO

=Amos Pro

... returns a value of -1, if the current programme is running under AMOS Professional. The same answer is returned if the programme is compiled with the AMOS Professional Compiler.

=CRAFT VERSION

=Craft Version

This function returns the current version of CRAFT. If there are bugs or we add some new instructions or functions to CRAFT, and your programme needs the newest version, you can always check the version with this function. Note that the version has to be divided by 100 before it can tell the truth, e.g. if this function returns 100, the real version is 1.00.

17: HELP

You have three alternatives how to proceed, if you don't know the syntax of an instruction, or you don't know what a function does. You can check the syntax in this manual, either from the instruction definition or from the Instruction Index (the next chapter). But if you want the definition now and you don't have time to search through your manual, you can always use the CRAFT Help System. Note that the Help System is only available for AMOS Professional.

Before you can use it, you have to put it into the User Menu. First select the Add Option from the User Menu. When the requester asks you to enter the name of the new option, write CRAFT Help. Then choose CRAFT Help from the menu, and select the CRAFT Help System programme from your disc. If you have a hard disc, you should put the programme on it. Note that this programme doesn't need any specific files, but it contains all the information needed by itself. When the "Program To Menu" window appears, select "[Fl] Load as hidden program" and if you have enough memory, you may select "[F3] Keep after run". You don't have to put anything in the command line. Finally press "Ok" and cancel the "Program To Menu" window by pressing a key. Next you have to choose "Set Key Shortcut" from the Config menu. When AMOS Pro asks you to choose an option, choose CRAFT Help from the User Menu and when AMOS Pro asks you to press a key combination, press [SHIFT]+[HELP]. Now you can always call the CRAFT Help System by pressing [SHIFT]+[HELP]. Don't forget to save your new settings.

When the Help System has been loaded, the main menu or an instruction definition will appear on screen. You can choose different topics from the main menu by pressing the left mouse key when the mouse cursor is over a button. This is the way to select other buttons, too. When you want to return to the previous page, press button [Return]. Note that this button shows the page from which you came to the current page. Try the programme and you'll learn quickly what the button [Return] does. If you press button [Prey Menu], you'll see the page that was selected from the main menu. To see the main menu press button [Main Menu]. When you want to quit, press the button in the upper left corner.

Note that the hypertext reader used in the CRAFT Help System is available on the disc and with it you can create your own hypertexts. Detailed instructions how to proceed can be found in the listing of the programme.

APPENDIX A: ERRORS

When there is an error in a CRAFT instruction, the extension returns an error to AMOS. There are two types of errors. Some of them are default AMOS errors and can be found in your AMOS User Manual. The rest of the errors are CRAFT errors. They are listed here: first, there is the name of the error and then there are some reasons why the error has occurred with some helpful instructions what should be done to get rid of the error.

<u>No fractal position defined</u>
You have forgotten to use FR POSITION instruction before drawing fractals. Add a FR POSITION instruction in your programme.

NoiractaLsten_spesifieil
Your programme doesn't set any fractal step rate, but it tries to draw a fractal. Add a FR STEP instruction to your programme.

<u>No fractal window defined</u>
You have tried to draw a fractal without defining the fractal window first. See the FR WINDOW instruction for more details.

<u>Not a palette hank</u>
You have tried to use PAL FROM BANK or PAL SWAP BANK instructions with a bank, which doesn't contain palette data. Check the bank and its number. You should also know that CRAFT uses the name of the bank to find out what kind of a bank it is. So you should never change the name of a palette bank.

Turtlenrrorladsyntax
There is a typing mistake or an unknown instruction in your TCL string. Check your string carefully and remember that the instructions should be in capital letters. Another reason for this error is that you have forgot to use separators between instructions.

<u>Turtle error: illegal function call</u>
A parameter in your TCL string is outside the allowed range. Check the string, there might be some typing errors in the numbers.

<u>Turtle error: illegal number of parameters</u>
You have forgotten some parameters or used too many parameters for a TCL instruction. If you omit a parameter, you still have to use the comma.

APPENDIX B: INTERNAL MEMORY STRUCTURES

If you are familiar with the machine code language, you'll find some interesting information in this Appendix. But beware, if you don't know what you're doing, it's better not to try it.

TheTurtleNatiable_Area

We have made the address of the internal turtle variable area available by providing the function =TR BASE. The purpose of this seemingly irrational action was not to increase the amount of instructions and functions in the extension, but to give the advanced user a possibility to get information which otherwise could not be accessed.

The AMOS/AMOS Pro extension system had some problems with floating point numbers until the release of AMOS Pro 2.00. Of course it would have been nice to be able to use floating point parameters with the fractal and turtle instructions, but you should bear in mind that integer parameters enable a drastically faster execution speed.

However, some floating point information can be acquired by converting the fixed point variables in the turtle variable area into floating point numbers. This way you can get the exact position and direction angle of the turtle. The variable area contains some less useful items of information, too. Please do not poke anything unconsidered into these addresses.

```
Offs Size Explanation
-------
 0 1 Flags: Bit 0 : 1 if the angle is OK
            Bit 1 : 1 if the angle coefficients are OK
            Bit 2 : 1 if the pen is up (no drawing)
            Bit 3 : 1 if the turtle is positioned somewhere
            Bit 4 : 1 if a turtle string is being executed
            Bit 5 : 1 if the proportions coefficients are set
            Bit 6 : 1 if the rememberers x,y are set
 2 4 The angle -180...179.9999 degrees (-2^31...2^31-1)
 6 4 The X coefficient for the angle
10 4       Y
        The low 17 bits of a coefficient are the absolute value
        0...65536 (corresponding to 0...1). Bit 31 is set if the
        coefficient is to be treated as negative.
14 4 The screen X coordinate of the turtle * 65536 (signed)
18 4
22 4 The screen X coordinate of the turtle home * 65536 + 32768
26 4
```

30 4 The X proportions coefficient * 65536 (signed)
34 4
38 1 Not used any more, maybe in a future update
39 1
40 2 The error position in the TCL string executed is:
42 2 the lower word - the upper word
44 4 The remembered X position (similar to screen X position)
48 4
52 4 The remembered angle (similar to the current angle)

A palette bank consists entirely of colour data. As you may have noticed, the length of a palette bank is exactly 64*n bytes, where n is the number of palettes in the bank. A colour is stored normally as a word in the form $ORGB. A colour that has been deleted, or masked out in the storing process, is stored as $ffff (-1). All the palettes are stored continuously without any data between them.

After the AGA version of the AMOS Pro is released, the palette bank system will have to be redone completely to allow differently sized palettes (2 to 256 colours) and 24-bit colour definitions.

APPENDIX C: USEFUL SYSTEM DATA STRUCTURES

This Appendix consists of two structure definitions. The first, the FIB, should be left untouched if you don't know what you're doing. And if you do know, you aren't allowed to poke in the FIB (it may crash the computer). The Preferences structure is a little bit different. As we have supplied you instructions to read and set the preferences, you may both peek and poke into the structure. Note, that before you can do that, you have to reserve some memory for it with instruction RESERVE AS DATA/WORK.

The File Info Block

This is the structure of the AmigaDOS File Info Block as returned by the DOS functions Examine() and ExNext(). The FIB of a file may be examined with the CRAFT functions =DR NAMES, =DR NEXT$ and =DR FIB. The DateStamp and NumBlocks fields are not yet implemented as CRAFT functions, but if you want to use them, write a line or two onto the registration card, and if we get enough requests, we'll code them into the next version of CRAFT.

Offs	Size		Explanation
0	4	DiskKey	
4	4	DirEntryType	If < 0, then a plain file. If > 0 a directory
8	108	FileName	Null terminated. Max 30 chars used for now.
116	4	Protection	The protection bits
120	4	EntryType	?
124	4	Size	Number of bytes in a file
128	4	NumB locks	Number of blocks in a file
132	12	DateStamp	The date of last change
132	4	Days	Number of days since Jan. 1, 1978
136	4	Minute	Number of minutes past midnight
140	4	Tick	Number of ticks past minute (0-49)
144	80	Comment	Null terminated.
224	36	padding	Not used.
260			

The Preferences Structure

Here is some interesting information on the AmigaDOS 1.2 preferences structure:

Offs	Size	Explanation
0	1	The height of the default system font (topaz 8 or 9)
36	64	The pointer sprite data: 2 planes * 16 words
100	1	The pointer hot spot X
101	1	
102	2	The pointer colour for the colour register 17
104	2	18
106	2	19
108	2	The pointer sensitivity (1 fast, 2 mediocre or 4 slow)
110	2	The Workbench colour 0
112	2	1
114	2	2
116	2	3
118	1	The X position of the Workbench and all the other screens
119	1	
164	2	Printer left margin in characters
166	2	Printer right margin in characters
176	2	Printer paper size: 0 US Letter, 16 US Legal, 32 Narrow tractor, 48 Wide tractor, 64 Custom
178	2	Paper length in lines
180	2	Continuous (0) or single sheet ($80)
185	1	Workbench interlace (1) or not (0)
226	2	Override default workbench width
228	2	Override default workbench height
230	1	Override default workbench depth

INSTRUCTION INDEX

Use this chapter if you want to find out the syntax of an instruction. After an instruction, there is a short definition on it.

If a parameter is enclosed by square brackets [], its optional. If there is a slash / between parameters, you have to use either parameter list at a time.

GLOSSARY

Here are some terms we have used and that may be a bit difficult to understand without a proper explanation.

Accessory is a programme that can be launched when editing another programme. When using AMOS Pro, an accessory can affect the current programme and is displayed over it.

Alert is the fancy red (yellow) box which usually appears when something has gone wrong. If you use an older Amiga (version 1.2 or 1.3) an alert is always the final thing before reset, but the newer Amigas can handle some alerts correctly.

AmigaDOS (Disk Operating System) is the low-level system which handles all the operations concerning files and discs.

Block - see Disc block

Bug is an error in a programme.

CHIP MEMORY is the part of memory which can be accessed by the Amiga's special chips; containing e.g. graphics and sound data.

CLI stands for Command Line Interface and is the programme that takes the instructions given by the user and tells AmigaDOS what to do.

Comment - see File comment

CRAFT stands for Colours, Requester, Audio, Fractals and Turtle.

Disc block is the smallest unit that can be handled when writing or reading data from disc. If you use normal AMOS instructions, you don't have to worry about blocks.

Electron beam is a beam of electrons, which draws the picture to TV and monitor.

FAST memory is the part of the memory which can be accessed only by the processor.

FIB (File Info Block) is an area of memory which contains information on a file.

File comment is a string, which is installed to a file and works like a note. The file comment can be seen with List command in CLI.

Fractal is a mathematical creature which has got an infinite number of details. Please refer to the chapter Fractals.

Iteration is an operation that consists of several mathematical calculations when drawing a fractal.

Logo is a programming language designed for young programmers. Nowadays it has disappeared almost completely and the only thing left is the indigenous turtle graphics, although it had a very flexible list handling system similar to that available in the LISP language used by AI researchers.

Mandelbrot, Benoit. The man who invented the fractal mathematics, among other things the Mandelbrot set (you surely didn't guess that one, did you?).

Module is a piece of music created with sound/noise/protracker. It contains all the information needed to play the music and also the samples.

Multitasking is a term which means that a computer (such as the Amiga) can run several programmes at the same time. This is achieved by flipping the programmes in a very high frequency.

Noisetracker is the second version of the music editor based on soundtracker. Nowadays Noisetrackers are replaced with Protracker.

Preferences are the data, which is used to set up Amiga in a proper way. They store the colours, the shape of the pointers, the printer settings and the various other things that can be changed with the Preferences programmes.

Priority is a number which specifies how much CPU time is given to various processes.

Protection bits define the legal actions which can be done to a file.

Protracker is the latest version of the music editor based on Soundtracker.

Requester is a box, which is used when Amiga needs more information from the user. Usually there are two buttons, positive and negative, but a requester can consist of sliders, multiple buttons and text fields.

Soundtracker is the original music-editor, from which e.g. Med, Protracker etc. have been derived. It has been replaced by Protracker.

TCL is the Turle Command language. Please refer to TR EXEC instruction.

Tick is a 1/50 of a second.

Turtle graphics is a way to draw simple pictures. It uses a turtle, which draws a line when it moves and the user can control the turtle and move it forward and backward or turn it left or right.