AMCAF-Manual

COLLABORATORS

	TITLE : AMCAF-Manual		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 15, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

AMCAF-Manual

1.1 AMCAF Extension Manual English

```
                AMCAF Extension V1.40 Manual by Chris Hodges.

AMCAF V1.50beta4 is the FINAL RELEASE! FREEWARE! NO DEMO VERSION ANYMORE!

Sorry, but I didn’t have time to update the manual. You’ll have to find out
the new commands since V1.40 yourself (or by looking at the history).
```

1.2 How to register AMCAF

How to register AMCAF

You don’t need to register anymore. AMCAF development has finally ceased.
I think it’s a bit overdue to abandon this project. Over 5 years of
development... phew!

However, you’re still encouraged to contact me:

Snail-Mail:

Chris Hodges
Kennedystraße 8
D-82178 Puchheim
West Germany
Tel: +49-89/8005856 (Voice/Modem)

Bank account:

Christopher Hodges
Account 359 68 63
Sparkasse Fürstenfeldbruck
BLZ 700 530 70

E-Mail:

platon@cu-muc.de

```
WWW:

http://platon.home.pages.de/

IRC:

platon42 on #amiga, #amigager (Efnet)
```

1.3 Important information about the AMCAF extension

```
                Welcome to the new world of AMCAF.

Welcome to the demo version of AMCAF. You will certainly be very satisfied
with this product. This extension contains over 200 commands and functions
for various areas. The AMCAF extension is more than 42 KB in size which
makes it the biggest extension for AMOS Professional known to me. Some
commands may work better or even require Kickstart 2.04 or higher. If you
want to take full advantage of the AMCAF extension then you should really
think about upgrading to 2.0 or 3.1, if you haven't already.

By the way, the extension must be entered at slot number 8 in the AMOS Pro
extension list, but this is done by the installation software for you!

Don't forget to read the
                 Notes
                , too!

No program is absolutely bugfree and for that reason I ask you to please
report any bugs as quickly as possible. However, it's rather useless, if
you only write: The command xxx crashes my machine. Please give me a
detailed description of your computer and enclose the part of the program
that causes the problem, but only if you like doing that kind of thing.

I welcome any suggestions that you may have, any problems that you may
encounter, bug reports for certain, or even any hints and tips that you
may have for me!


                 Contact address
                If you have got access to a ftp site, there will be free updates ↩
                   coming up
regularily on Aminet (dev/amos). Make sure you download the English
version and not the German one.

***************************************************************************
*                                                                         *
*** ATTENTION *** ATTENTION *** ATTENTION *** ATTENTION *** ATTENTION ****
*                                                                         *
* AMCAF is NOT sold by The Software Society anymore! You will only get   *
* into trouble when buying AMCAF from them, despite they only have the   *
* old german version and not the english one! Moreover, it's much more   *
* expensive than the shareware version (somebody had to pay about 120 DM *
* for it instead the 30 DM shareware fee).                               *
*                                                                         *
* So DON'T buy AMCAF from The Software Society, register it HERE.        *
*                                                                        *
**************************************************************************
```

Enjoy AMCAF and may the force be with you!

Chris Hodges

1.4 Notes about the enclosed disks

Notes about the installation and examples disks.

If you haven't already installed AMCAF, you should do so as soon as possible, because no example program runs without the AMCAF extension (which is quite obvious ;-)).

The installation process is quite easy and quicky done. Just read the instructions carefully which appear when you start the installation program.

During the process you will be asked to enter your name, if you are using a registered version. Please enter it correctly. It will be encrypted and saved into the extension file. Remember that you MUST NOT copy the registered version of AMCAF to anybody else. YOU have paid for this software, why should someone else get it for free?

AMCAF can either be written directly onto harddisk or onto a copy of your AMOSPro_System disk. When installing to disk, AMCAF allows you to either delete some files directly on the disk to get some free space for the extension or to copy the system files to a new disk (preferably FFS), which is formatted by the program directly.

Due to lack of space on the examples disk, the file "mod.no good" has been written onto the installation-disk. The only program that needs this file is the demo "NoGood.AMOS". It automatically loads the file either from the install disk or harddisk.

To unleash the power of the command "Set Rain Colour", the main Amos Pro library must be modified a bit (or better: a byte). So please first start the program "SetRainColourPatch" and then restart AMOS Professional. This patch remains permanently so you have only to start it once.

The examples haven't been written for you to start them, look at them for a moment and then load the next. Most of the programs are well documentated and allow you to learn something new.
Take your time, it's worth it.

Some commands are very sensitive agains abuse. Do save your programs regularily, especially when you've added a new routine.

1.5 Send your mail to...

If you have any questions or even found a bug, then leave me a mail or call me directly.

Don’t forget to enclosed a DETAILED DESCRIPTION OF THE ERROR with the configuration of your computer and the piece of code which doesn’t work correctly if you’ve encountered a bug.

Snail-Mail:

Chris Hodges
Kennedystraße 8
D-82178 Puchheim
West Germany
Tel: +49-89/8005856 (Voice/Modem)

Bank account:

Christopher Hodges
Account 359 68 63
Sparkasse Fürstenfeldbruck
BLZ 700 530 70

E-Mail:

platon@cu-muc.de

WWW:

http://platon.home.pages.de/

IRC:

platon42 on #amiga, #amigager (Efnet)

1.6 Some bugs you must take care of

Bug report

This is a small list of AMOS bugs you must be aware of:

1. The AMOS string management is not bugfree. Be especially careful with AMAL programs!

2. The AMOS Default routines are not called:
   If you load AMOS and then start a program, the extension default routines are not called. I.e if a music is running from the previous call, it will not be stopped.
   Conclusion: - Call any accessory (e.g the help-accessory) once.

3. Programs cannot be compiled:
   The compiler prints out the error ’Not an AMOS program’. If you use memory banks, all bank lengths must be even. If this is not the case, the programs do work from the editor but not compiled.
   Conclusion: - Extend odd banks by one byte using Bank Stretch.

4. A program that uses data lines does not work correctly when compiled:
   There must not be any comments after a data line. Data commands must be
   completely alone on a line. Otherwise the compiler will interprete
   these comments as data.
   Conclusion: - Write the comments into a seperate line.

   Even worse: a friend of mine was faced to a guru when trying to compile
   a program that uses aprox. 500 KB of data lines. As for now, I found no
   solution for this problem.

Bugs that are interesting for extension programmers:

1. Your commands or functions can only have a maximum of 9 parameters.
   Everything above this will not be put on -(a3).

2. All routines, that are jumped to directy by a token must be placed in
   the first 32 KB of code. Otherwise the command will crash. Additionally
   the distance of a RBras or RBccs jump must not exceed 32 KB.

1.7 Copyrights

Amiga is a trademark of Escom AG.
MC68xxx series are trademark of Motorola Inc.

AMOS by François Lionet.	Copyright 1990 Mandarin/Jawx.
	Copyright 1991 Europress Software Ltd.
AMOS Pro by François Lionet.	Copyright 1992 Europress Software Ltd.
AMCAF by Chris Hodges.	Copyright 1994 The Software Society.
	Copyright 1995 Chris Hodges.
AMIPS by Thomas Nölker.	Copyright 1993 The Software Society.
TOME by Aaron Fothergill.	Copyright 1991 Shadow Software.
PowerPacker by Nico François.	Copyright 1990 PowerPeak.
ProTracker 2.2 by P. Hanning.	Copyright 1992 Noxious. (PD)

Turbo Imploder by P. Struijk & A. Brouver. Freeware 1991.

1.8 You think you will be greeted? ;)

First, best regards to everybody who has registered AMCAF, thank you very
much.

Then special greetings fly to:

My parents (Hallo!)
My brother (Come on! I want to see a wonderful raytracing animation!)
My sister (Hi pumpkin!)
Hans Peter Obermeier (Anything new? ;-))
Ralf Schulz (Oh, I cannot beleve you've bought an ugly MS-Tinbox... :-( )
Markus Ungerer (Schreib mal wieder eine Kurzmail!)
Bernd Ungerer (Danke fürs Beta-Testen und für die guten (?) Vorschläge!)
Michael Ufer (You little hobby magician! Thx for the many refreshing mails)

```
Oliver Ufer (Thx for the huge amount of suggestions ;-) )
Oliver Seibert (Surprise had been the best box known to me!)
Oliver K.
Dirk Drießen (Man gönnt sich ja sonst nix ;-))
Ralph Bernecker (Hello jMS/Dr.Feelgood/Striker/fELON)
Alexander Kunz (Thx 4 da kewl tunes and support!)
Omer Sasic (+++)
Claude Müller (Greetings to Swiss! I hope your back from the army soon!)
Dirk Schulten (Hallo Maus-User! ;-) )
Andre Panser (Schreib mal wieder)
Andreas Duncker (Thx for your supporting mail)
Mathias Mischler (Thx für your support and logement!)
Andreas Zymny (Das Quotezeichen bleibt UNKONFIGURIERBAR! Basta! ;-) )
Kriegsheld (Jaja! Eigentlich ist ja heute schon morgen, gelle?)
Henning Baron
Rainer Benda
Robert Rothhardt (Thx for the best time in all my school life!)
Florian Fackler (Strato Impact rulez!)
Thomas Nölker (Good luck with your AMIPS-Extension!)
Jürgen Schäfer (Sorry wegen der Gif-Geschichte, vielleicht kommt das noch)
Greg Cox (Keep up the good work!)
Michael Cox (Thanks for adding me to the mailing list ;-) )
Marco Eberhardt (Thanks for your nice mails)
Carsten Albert
Paul Hickman (death to all tennis players! ;-))) )
Andy Church
Ben Wyatt (Greetings! ;-))) )
Petri Hakkinen (nice 3D engine!)
Roy Antonsen (stop drinking so much booze ;-) )
Martijn Wehrens (nice mails from ya! Keep that up)
Semprini (you're a REAL friend!)
Mark Wellington (nice stories from ya, keep that up!)
Daniel Rädel
Thomas Nokielski
Jari Jokivuori
and all the others, who know me and I forgot by mistake.

In addition, more greetings go to my modem friends:

Magic, Lemming, VIP, Schneemann, Killer, Marvin, Harry, Holger,
Caboose, Blue Shogun, Ralli, REYem, Nosy and Kily.

Merlin, WotaN, Vinzenz, Fritz, Braumeister, Amigaman and Kai.

Dr.Dre, Tomy, Brocker, Case, Omer and Guru.

Curses to:

Hendrik Heimer (You know why, Mr. Software Society)
François Lionet
Jester (Christoph Steinecke)
Nobody (Horst Bressem)
Intel, IBM and MicroSoft
German Telekom
Europress
Data Becker and Sybex
R2B2 ($§%$&%§$)
```

```
Phase 5

Good... I hope, I've forgotten anyone... erm yes... greetings to all
members of the RAMSES-Computerclub.

CU
    ______   __  __   ______   __   _____
   /,____/\ / /\/,/\ / __ //\ /,/\ / ___/\
  ///\___\// /_/// //,/_/'/ //// //./___\/
 /'/_/_   / __ '/ ///  __/ //'/ / \__, /\
/_____/\ /_/\/_/ //_/\_\_\//_/ / /____/ /
\_____\/ \_\/\_\/ \_\/\_\  \_\/  \____\/
```

1.9 AMCAF history

```
AMCAF extension history.

V1.50beta 11-Jan-98
-------------------
- Found some wrongly assembled lines in the protracker vibrato table.
  If someone encountered some scarce problems the modules using
  fine-vibrato, these should be gone now.
- New c2p routine by Mikael Kalms. Up to 20%-80% faster and now also
  supports plane depth from 4 to 6!
- Fixed a little bug that wouldn't play samples longer than 64KB
  correctly (just a lsr.w instead of a lsr.l).
- Probably fixed a lot of bugs and spurious crashes introduced with the
  last beta.
- Fixed a newly introduced bug in Lzstr$ and Lsstr$.
- Found something new out about AMOS and fixed some bugs with editor-
  compiled programs.
- Added new transition commands:
  - Alloc Trans Source bank
  - Set Trans Source bank/address
  - Alloc Trans Map bank,width,height
  - Set Trans Map bank/address,width,height
  - Alloc Code Bank bank,size
  - Trans Screen Runtime scr,bitplane,ox,oy
  - Trans Screen Dynamic scr,bitplane,ox,oy
  - Trans Screen Static NOT YET IMPLEMENTED
  - Trans C2p chkbuf NOT YET IMPLEMENTED
  Some last words: Allocating the Code bank to small will cause memory
  overwrites. Wrong or stupid parameter values are not checked for
  validity. The Trans commands are still a bit slow (3 VBLs for a 256x256
  pixel transition with MC68060 and all caches disabled, 0.8 VBLs for the
  same transition with all caches turned on).
  Maybe the C2P Transition will be a bit better. Still need to update the
  c2p to a state-of-the-art routine (by Kalms).
- New for ALL PT sample commands: Entering a negative value as channel
  bitmask will trigger the Pt Free Voice function which will search for
  the best channel number to use for sample replaying. Really handy
  indeed! Pt Free Voice is rather complex:
  - it first checks if the bitmask is zero and if so, it returns 0.
  - then it checks, if only one bit was set in the mask and returns
    the same.
  - after that, it finds out if all four channels (or the ones given in
    the bitmask) are currently playing a sound. If so, it returns the
    channel bit of the sound which will cut off least.
  - if there are free channels, it verifies, if music is playing at all.
    If not, it returns the first free channel.
  - so for music is playing, it looks if the user has masked out a
    channel from the music (and which is currently free).
  - then it checks for the remaining channels are playing looping sounds
    for those will be not taken if necessary.
  - the last step finds out the shortest sound currently played on the
    channels, and then returns its channel bit.
  So you'll get the most of your game sfx and music if you're using the
  free voice function!
- New functions:
  - =Pt Free Voice[(bitmask)]
- Lzstr and Lsstr now return '9999...' if number is too large to fit into
  the given amount of digits.
- Sample&music-mixer changed to be more accurate (samples could have been
  cut off some VBLs too early if the music was playing at higher cia
  speeds than 125bmp) with cia-timed mods (at least this should have been
  the case in theory).
- The sample replayer now uses a 2 byte chip mem buffer to kill the
  high pitched beep sound sometimes appeared on samples which started
  with 2 non-zero bytes. FixSamBank should now no longer be required.
- Pt Raw Play can now play looping samples by giving a negative length.
- Implemented Vu Meters to AMAL. BUT it will ONLY work, if you REMOVE
  the original AMOS Music Extension. No other way to get it work!
- Added support for negative numbers in Lzstr$ and Lsstr$.

V1.44 27-Jul-96
---------------
- Fixed a bug in Ssave.
- Fixed a minor bug in Pt Play (cia speed is now reset to 125).
- Fixed a bug in Qsqr. Now bigger numbers than 0-65535 can be used.
  (Warning: starting with numbers >65535 it gets a bit inaccurate!
  Remember that the resolution of the result remains 8 bit!)

V1.43 02-Nov-96
---------------
- Added Sload/Ssave. Just the same commands like in the music
  extension. Now you can really remove it!
- Misc bugfixes.

V1.42 18-Oct-96
---------------
- Added support for Set Tmpras for Fellipse&Fcircle commands.
- Added type check for Protracker modules.

V1.41 29-Mar-96
---------------
- Fixed bug in Xfire.
- Misc changes.

V1.40 26-Dec-95
---------------
- New commands:
  - =Pt Cpattern
  - =Pt Cpos
  - =Pt Cinstr(channel)
  - =Pt Cnote(channel)
  - Pt Sam Freq channel,freq
  - =Vclip(v,lower To upper)
  - =Vin(v,lower To upper)
  - =Vmod(v,upper)
  - =Vmod(v,lower To upper)
  - =Insstr$(a$,b$,pos)
  - =Cutstr$(a$,pos1 To pos2)
  - =Replacestr$(a$,search$ To replace$)
  - =Itemstr$(a$,item)
  - =Itemstr$(a$,item,sep$)
  - =Qarc(dx,dy)
  - =Even(val)
  - =Odd(val)
  - =Ham Point(x,y)
  - Set Object Date file$,date,time
  - =Aga Detect
  - Pal Spread c1,rgb1 To c2,rgb2
  - =Ct String(time$)
  - =Cd String(date$)
  - C2p Convert st,wx,wy To screen,ox,oy
  - C2p Shift st,wx,wy To st2,shift
  - C2p Fire st,wx,wy To st2,sub
- Pal Set Screen now does a 'View' automatically.
- Another bug in Ptile Paste has been fixed.
- Sine-Table moved and shortened, so I save about 1536 Bytes,
  added Arctan-Table.

V1.31 03-Oct-95
---------------
- Actually, the bug hasn't been in Change Bank Font but in Make Bank Font.
  Although this bug is now fixed, I suppose you have to recreate all old
  bank fonts.
- Finally removed Rnc Unpack and =Rnp.
- Fixed a bug in Pt Stop which cut off the channels, even if no music had
  been playing.
- New commands:
  - Pt Continue
- Fixed a bug in Change Bank Font.

V1.30 31-Aug-95
---------------
- NEXT OFFICAL UPDATE.
- Various changes to get the extension under the 32 KB limit.
- Best Pen now supports EHB mode.
- Found a bug in the PlaySample routine and removed it.
- Removed a little bug in Pt Raw Play.
- Obvious bug in Extpath$ fixed (Thx Ben!)
- New commands:
  - =Best Pen($RGB[,c1 To c2])
  - Bzoom s1,x1,y1,x2,y2 To s2,x3,y3,m
  - =X Smouse
  - =Y Smouse
  - Limit Smouse [x1,y1 To x2,y2]
  - Smouse X x
  - Smouse Y y
  - Smouse Speed speed
  - =Smouse Key
  - =Xfire(port,button)

V1.19 30-Jun-95
---------------
- New commands:
  - Exchange Bob i1,i2
  - Exchange Icon i1,i2

V1.18 21-Mar-95
---------------
- NEXT OFFICAL UPDATE.
- Fixed a very scarcely appearing bug in Turbo Plot.

V1.17 16-Feb-95
---------------
- Little bug in the protracker routines: mask for FineSlideUp was $D
  instead of $F, what resulted in ignoring $2 slides. (Thx Patrick)

V1.16 29-Jan-95
---------------
- Little flaw appeared with the protracker cia replay code: Pt Cia Speed
  did not work correctly in compiled programs (but I don’t know why!).
- Forgotten a Pt Sam command by mistake:
  - Pt Sam Volume [voice,] volume
- Reworked code a bit to ensure the demoversion is working (32 KB limit!).
- New palatte handling commands:
  - Pal Get Screen palnr,screen
  - Pal Set Screen palnr,screen
  - =Pal Get(palnr,colindex)
  - Pal Set palnr,colindex,colour

V1.1 28-Dec-94
--------------
- Adapted protracker replaycode to handle samples correctly.
- New commands:
  - =Pt Data Base
  - =Pt Instr Address(samnr)
  - =Pt Instr Length(samnr)
  - Pt Bank bank
  - Pt Raw Play voice,address,length,freq
  - Pt Instr Play samnr
  - Pt Instr Play voice,samnr[,freq]
  - Pt Sam Bank bank
  - Pt Sam Play samnr
  - Pt Sam Play voice,samnr[,freq]
  - Pt Sam Stop voice
- Tiny optimations on Ham Fade.
- One more bug in Ptile Paste removed.
- Bug in Speek: Speek did only allow even addresses.
- Silly bug in Lsstr$ and Lzstr$. Often they did produce trashed strings
  or didn’t return from the call.

V1.0 16-Oct-94
--------------
```

```
- VERY FIRST RELEASE VERSION!
- Removed a few little bugs in the registration code.

V1.0ß 06-Oct-94
---------------
- Imploder Load didn't free the lock on a file if the loading process was
  successful. Fixed.

V1.0ß 22-Sep-94
---------------
- Every string allocation reserves two bytes more. Now there seem to be no
  problems with the stringbuffer anymore.

V1.0ß 28-Aug-94
---------------
- Ptile Paste had a bug. Removed.
- =Extpath$ did overwrite 32 KB of variable buffer using empty strings!!!
  Argl...
- =Ham Best completely rewritten. Works even better now.
- Added one more errorcheck to Coords Read and =Count Pixels.
- Coords Bank reserved a bank which was not completely used up and could
  hold exactly one pair of coordinates less.
- Coords Bank corrupted the memory or crashed when trying to create a bank
  with zero coordinates.
  The same with Splinters Bank. Bug removed.
- =Ham Best rewritten. Now it is much quicker as before!
  Hint came from Dr. Peter Kittel, now ex-employee of Commodore, Class 95.
- Protracker: Vumeter did not support the C-Command, for that reason
  =Pt Vu returned 64 as volume most of the time. Bug removed.
- Protracker: If a channel had been turned off, neither the signal could
  be received nor have speed and other commands been taken notice of.
  Implemented subroutine to handle this case.
- Renamed Secexp to Binexp and Seclog to Binlog.
- Protracker: added more init code. Now there should not be any flaws with
  funks or patternrepeats when playing many modules after each other.
- Moved database from chipram to fastram (really should have done this
  much earlier!).
- Due to the improvements named above, the code became to long to execute
  a certain branch command. This caused AMOS to crash. Bug removed.

V0.992ß 27-Jul-94
-----------------
- Yeah, got some vacations at last and can now fully concentrate on the
  manual. Releasedate of V1.0 was set to 1.9.94.
- Extension restructured, so ensure that all functions, that do have a
  token, are placed into the upper 32 KB. Now everything should work again
  correctly.
- Now you can switch between CIA and VBL timing while playing a module.
- Encountered a small bug in Bank Copy which was killed immediately.
- Reimplemented some commands: Rnc Unpack and =Rnp.
- New commands:
  - =Qsqr(value)
  - Bcircle x,y,r,c
- Added clipping for Turbo Plot, Shade Pix and Turbo Point. Now they are
  as secure as the normal Plot and Point commands.
- Improved the speed of Ham Fade a little.
- When reaching the end of a song, Pt Signal now reports $FF.
```

```
V0.991ß 18-Jul-94
-----------------
- New commands:
  - Set Rain Colour rainnr,colour
  - Rain Fade rainnr,colour
  - Rain Fade rainnr To rainnr

V0.990ß 01-Jul-94
-----------------
- Removed some command to shrink the size of the extension:
  - Rnc Unpack
  - =Rnp
- Deleted a few error messages from Io Error$.

V0.990ß 04-Jun-94
-----------------
- 'Created' a lethal bug by resorting the command groups. Seems to be an
  AMOS interior bug, but when I write Audio Lock and Audio Free at the end
  of the extension every other command should work. However, I could not
  test if every command works, so be warned! Strangely enough, all
  commands do work when compiled, which indicates the existence of this
  AMOS bug.
  DO NOT USE AUDIO LOCK OR AUDIO UNLOCK IN THE INTERPRETER MODE!!!
- New commands:
  Blitter Copy Limit screen
  Blitter Copy Limit x1,y1 To x2,y2
  Blitter Copy sc1,pl1 [,sc2,pl2 [,sc3,pl3]] To sc4,pl4 [,minterm]

V0.989ß 03-Jun-94
-----------------
- Discovered bugs in Blitter Fill and Blitter Clear which have caused some
  strange structure faults (e.g quiting from For-Next loops)

V0.989ß 03-Jun-94
-----------------
- Made Blitter Fill more secure.
- New commands:
  - Blitter Clear screen,plane
  - Blitter Clear screen,plane,x1,y1 To x2,y2
  - Blitter Wait
  - flag=Blitter Busy
  - Shade Pix x,y
  - Shade Pix x,y,planes

V0.988ß 02-Jun-94
-----------------
- At last Turbo Draw does now support clipping. Even implemented some
  special checkroutines for the blitter mode.

V0.988ß 31-May-94
-----------------
- Hurray!!! Splinters do now work totally correctly.
- Various optimations on Splinters and Td Stars.
  - Reduced memory consumption from 32 to 22 bytes per Splinter.
  - Reduced memory consumption from 16 to 12 bytes per Td Star.
```

```
V0.987ß 29-May-94
-----------------
- New commands, to satisfy Markus:
  - Make Bank Font bank
  - Change Bank Font bank
- New function:
  - =Cop Pos
- Little flaw: Pt Play did not reset the signal to zero.
- New functions:
  - =Vec Pos Z(x,y,z)
  - =Vec Pos Z
- Removed private commands.

V0.986ß 27-May-94
-----------------
- Added three quite private temporary commands (for a maildisk)
  Will be removed immediately after completing the maildisk.
  - Private A bank1,bank2,bitplane,maxrand
  - =Private B(bank2)
  - =Private C(bank2)
  DO NOT USE!!! Wrong use will crash the computer!!!
- New function:
  - =Qrnd(value) as replacement for Rnd... does not need Randomize and is
    faster.
- Error in the tokenlist caused a wrong syntax of Blitter Fill to be
  converted into Pt Play. Funny :)
- Found and removed an error in the Blitter Fill commands. Blitter Fill
  filled the screen one line to deep -> memory got corrupted.
- There was a bug in the vector rotation calculation with negative
  positions.

V0.986ß 25-May-94
-----------------
- New command to stop Bernd from complaining:
  - Change Print Font bank

V0.986ß 24-May-94
-----------------
- Completed the Vec Rot commands. Removed =Vec Rot Adr again.
- Vec Rot commands:
  - Vec Rot Pos posx,posy,posz
  - Vec Rot Angles angx,angy,angz
  - Vec Rot Precalc
  - =Vec Rot X(x,y,z)
  - =Vec Rot X
  - =Vec Rot Y(x,y,z)
  - =Vec Rot Y

V0.985ß 24-May-94
-----------------
- Removed a bug in Cd Date$. (Thx Bernd)
- Discovered and corrected a cheap bug in Blitter Fill screen,plane.
- Removed the bug in Turbo Draw when using the blitter mode.
- New commands:
  - =Qsin(angle,factor)                 angle must be between 0 and 1023
  - =Qcos(angle,factor)                 factor is the value that is
                                        multiplied with the sinus value.
  - Vec Rot Pos midx,midy,midz          Positions for the vector rotation.
  - Vec Rot Angles angx,angy,angz       Rotationangle.
- Still buggy:
  - Vec Rot Precalc                     Creation of the matrix
  - =Vec Rot X(x,y,z)                   Calculation of the new x-value
  - =Vec Rot Y(x,y,z)                   Calculation of the new y-value
- Added the function =Vec Rot Adr for testing reasons.

V0.984ß 21-May-94
-----------------
- New syntaxes for Blitter Fill:
  - Blitter Fill screen,plane
  - Blitter Fill screen,plane,x1,y1,x2,y2
  - Blitter Fill s1,p1 To s2,p2
  - Blitter Fill s1,p1,x1,y1,x2,y2 To s2,p2
- Turbo Draw doesn't draw on non-existing bitplanes anymore. (Thx Marc)
- Change Font now adds '.font' automatically, if needed. (Thx Markus)

I'm afraid to say that there are no history entries for the other half of
the year.
```

1.10 Easy to use bank manipulation commands

```
                Bank management and modification.

AMCAF contains many commands that are dedicated to AMOS memory banks. With
all these commands it's important to use EVEN addresses, if there are some
demanded. Otherwise you will crash any computer with MC68000 processor. In
addition, the lengths of every bank must be even, or the compiler will
report a "Not an AMOS program" error, but this is a problem of AMOS and
not of AMCAF.


                 General aspects
                 - Some information about AMOS banks

Commands:


                 Bank Permanent
                 - Makes a bank
                 Permanent

                 Bank Temporary
                 - Makes a bank
                 Temporary

                 Bank To Fast
                 - Moves a bank into
                 Fast ram

                 Bank To Chip
                 - Moves a bank into
                 Chip ram
                 Bank Stretch
                 - Extends a bank after it has been reserved

                 Bank Copy
                 - Copies a bank

                 Bank Name
                 - Changes the name of a bank

                 Bank Code xxx.y
                 > Commands to encode and decode banks

                 Bank Delta Encode
                 - Prepares a bank with delta-encoding

                 Bank Delta Decode
                 - Removes the delta-encoding of a bank

Functions:


                 =Bank Checksum
                 - Calculates a checksum of a bank

                 =Bank Name$
                 - Returns the name of a bank
```

1.11 Bank Code commands

```
                Bank Code commands

Using these commands you can encode banks in many different ways to
protect them from unauthorized access and insight. Especially manual copy
protections can be made more secure by encoding the specific keyword bank.

Each command comes in two versions, one with the suffix
                 .b
                 and
one with
                 .w
                . By using the .b version the codenumber can
range from 1 to 255, the .w version allows codes from 1 to 65535. However,
the rotational commands are an exception as the codes may only reach from
1 to 7 and from 1 to 15 respectively.

Every command has the following syntax

Bank Code xxx.y code,bank
Bank Code xxx.y code,startaddress To endaddress

Command versions:


                 Bank Code Add.y
                 - Additional algorithm encoding

                 Bank Code Xor.y
                 - Xor algorithm encoding

                 Bank Code Mix.y
                 - Mix between Add und Xor

                 Bank Code Rol.y
                 - Rotation to the left

                 Bank Code Ror.y
                 - Rotation to the right
```

1.12 Disk commands

```
                Disk handling commands

AMCAF tries to cover this area as well and provides you with the commands
you have been desperately seeking for in AMOS until now.


                 Primary commands
                 > Commands for loading, saving, copying, etc.

                 File access
                 > Commands for handling files

                 Support functions
                 > Additional functions for file handling etc.

                 Packersupport
                 > Powerpacker and Imploder commands
```

1.13 Disk support functions

```
                Disk support functions

These are various functions to help you with disk access and filenames.


                 =Io Error
                 - Returns the last dos error code

                 =Io Error$
                 - Returns a dos errorstring

                 =Filename$
                 - Returns the filename of a full path
```

```
=Path$
- Returns the directory of a full path

=Extpath$
- Appends a "/" to a path if required

=Object Protection$
- Returns a
Protection flags
string

=Pattern Match
- Compares a string with a certain pattern

=Disk State
- Returns the state of a disk device

=Disk Type
- Returns the type of a volume

=Dos Hash
- Calculates the hash value of a file
```

1.14 Commands to handle DOS objects

Dos object handling commands.

The AMOS functions Dir First$ and Dir Next$ are neither very reliable nor very easy to use. Using these commands you could only get the name and the length of the object but what could you do if you wanted to know even more? Thankfully, AMCAF provides you with an easy solution for these problems.

Commands:

```
Examine Object
- Gets all information about an
Object

Examine Dir
- Inits the reading of a drawer

Examine Stop
- Stops the reading of a directory
```

Functions:

```
=Examine Next$
- Reads the next entry in a directory

=Object Name$
- Returns the name of an
```

```
                 Object

                 =Object Type
                 - Returns the type of an
                 Object

                 =Object Size
                 - Gives back the length of a file

                 =Object Blocks
                 - Returns the length of a file in blocks

                 =Object Protection
                 - Returns the
                 Protection flags
                 of an
                 object

                 =Object Time
                 - Returns the time of creation of an
                 object

                 =Object Date
                 - Returns the date of creation of an
                 object

                 =Object Comment$
                 - Gives back the filenote of an
                 Object
```

1.15 Packer support

```
                Packer support

Packer are nearly the most important tools, if you try to fit a lot of
files onto a disk. Powerpacker and File-Imploder are two of the best and
how else could it be, AMCAF does support them to some extent.


                 Ppfromdisk
                 - Loads and unpacks a powerpacked file

                 Ppunpack
                 - Unpacks a powerpacked file

                 Pptodisk
                 - Packs and saves a file as PP20


                 Imploder Load
                 - Loads and decrunches a FileImploder file

                 Imploder Unpack
                 - Decrunches a FileImploder bank
```

1.16 Primary disk commands

Primary disk commands

AMCAF gives you some very easy to use commands to load, save, and modify files.

File Copy
- Copies a file

Wload
- Loads a file Temporarily

Dload
- Loads a file Permanently

Wsave
- Saves a file to disk

Dsave
- Saves a file to disk

Protect Object
- Modifies the Protection bits of an object

Set Object Comment
- Sets the filenote of an Object

Set Object Date
- Sets the date of an Object

Launch
- Starts a new process

1.17 Interior extension commands

Interior extension commands and commands for advanced users

Again some commands and functions for advanced users. Before using these, think about what you are doing, then again, and once more, and more important, save your program before starting it.

Commands:

```
                 Extdefault
                 - Calls the default routine of an extension

                 Extremove
                 - Removes a extension from memory

                 Extreinit
                 - Tries to revoke a extension
```

Functions:

```
                 =Amos Task
                 - Returns the address of the AMOS task structure


                 =Amcaf Version$
                 - Returns an AMCAF version string

                 =Amcaf Base
                 - Gives back the address of the AMCAF data base

                 =Amcaf Length
                 - Returns the length of the AMCAF data base
```

1.18 Four player adapter support

Four player adapter

If you are writing games, it's a good thing to add a multi player option. Given the case even four or more players may compete against each others, you should think about implementing the four player adapter. Obviously, it's more comfortable to enjoy a game when using a joystick than a keyboard.

How to build a four player adapter

Functions:

```
                 =Pjoy
                 - Acquire direction of a joystick

                 =Pjup
                 - Check if joystick is pressed up

                 =Pjdown
                 - Check if joystick is pressed down

                 =Pjleft
```

```
                  - Check if joystick is pressed left

                  =Pjright
                  - Check if joystick is pressed right

                  =Pfire
                  - Check if fire button is pressed
```

1.19 Graphic and effect commands

```
                 Graphic and effect commands

AMCAF contains a lot of graphic commands, many of which you surely can use
in your games or demos. However, you should be aware that most commands do
not check the parameters you use! This improves the speed of some
commands. Remember to save your commands, due to the fact that a small
mistake can crash your computer.


                  Basic commands
                  > New basic commands

                  Graphical effect commands
                  > Effects

                  Font commands
                  > Commands for fonts loading etc.

                  Colour instructions
                  > Commands and functions for colour management

                  Blitter commands
                  > Direct control of the blitter chip

                  Other commands
                  > Misc commands and functions

                  Commands for system hacker
                  > For advanced users only.
```

1.20 Commands to directly access the amiga blitter chip

```
                 Blitter commands

AMCAF makes it possible to control the
                  Blitter
                  by hand and to
clear, fill, copy or modify
                  Bitplanes
                  directly.
```

Commands:

```
                 Blitter Copy
                 - Copying and modifying a bitplane

                 Blitter Copy Limit
                 - Setting the Blitter Copy area

                 Blitter Fill
                 - Filling polygons using the blitter

                 Blitter Clear
                 - Clearing a bitplane with the help of the blitter

                 Blitter Wait
                 - Waiting for the blitter has finished his task

                 Turbo Draw
                 - Fast replacement for Draw

                 Bcircle
                 - Drawing a circle to fill it using the blitter
```

Functions:

```
                 =Blitter Busy
                 - Returns the blitter's current state
```

1.21 Supportfunctions for colour management

Functions to manipulate colours

Playing with colour was often linked with complex formulas. But now there is a series of helping commands - only AMCAF makes it possible!

Commands:

```
                 Rain Fade
                 - Fades a rainbow out or to another one

                 Set Rain Colour
                 - Changes the affecting colour of a rainbow


                 Pal Spread
                 - Spreads from one colour entrie to another

                 Pal Get Screen
                 - Saves the palette of a screen
```

```
                  Pal Set Screen
                  - Sets the palette of a screen

                  Pal Set
                  - Changes an entry of a saved palette


                  Amcaf Aga Notation
                  - Toggle
                  AGA-Amiga
                  colour format

Functions:


                  =Pal Get
                  - Reads a saved palette entry


                  =Red Val
                  - Calculates the red value of a colour

                  =Green Val
                  - Calculates the green value of a colour

                  =Blue Val
                  - Calculates the blue value of a colour

                  =Glue Colour
                  - Generates a colour using the three colour values

                  =Mix Colour
                  - Mixes two colours


                  =Best Pen
                  - Calculates the nearest pen for $RGB

                  =Ham Point
                  - Returns the RGB value of a ham pixel

                  =Ham Colour
                  - Calculates a colour in
                  HAM mode

                  =Ham Best
                  - Calculates the best colour in
                  HAM mode

                  =Rgb To Rrggbb
                  - Converts a ECS-colour into
                  AGA colour format

                  =Rrggbb To Rgb
                  - Converts a
                  AGA colour
                  into a ECS-colour value
```

1.22 Graphical effect commands

```
                Graphical effect commands

A big part of AMCAF contains graphical effects. If you only want to have a
few stars or simply want to make a logo explode or create some neat
effects using shade bobs, AMCAF makes (nearly) everything possible!


                 Mask Copy
                 - Screen Copy with a mask

                 Pix Shift
                 > Pix Shift commands group

                 Shade Bobs
                 > Shade Bobs commands group

                 Td Stars
                 > Td Stars commands group

                 Splinters
                 > Splinters commands group
```

1.23 Font commands

```
                Font commands

AMCAF provides you with powerful instructions for font managing. For
example the Change Font command allows you to set a font without having to
get the full font list using Get Disk Fonts etc. and therefore reduces the
time to access disk considerably. Especially hard disk drive users will be
very happy about this command. Additionally, you even get the unbelevable
opportunity to store any diskfont in an AMOS memory bank and so are
totally independent of the fonts directory and the diskfont.library.

Commands:


                 Change Font
                 - Loading a font directly from disk

                 Make Bank Font
                 - Creating a font bank

                 Change Bank Font
                 - Setting the screen font using a font bank

                 Change Print Font
                 - Changing the font that is used by Print.

Functions:


                 =Font Style
                 - Getting the attributes of a font
```

1.24 Basic commands

```
                Basic graphical commands

AMCAF expands the basic command set of AMOS by some important commands.

Commands:


                 Bzoom
                 - Zooms a region

                 Ham Fade Out
                 - Fades out a ham picture

                 Convert Grey
                 - Creates a grey scale picture

                 Turbo Plot
                 - Fast replacement for Plot

                 Turbo Draw
                 - Fast replacement for Draw

                 Fcircle
                 - Draws a filled circle

                 Fellipse
                 - Draws a filled ellipse

                 Raster Wait
                 - Waits for a specific raster position

Functions:


                 =Turbo Point
                 - Fast replacement for Point

                 =X Raster
                 - Gets the X position of the raster beam

                 =Y Raster
                 - Gets the Y position of the raster beam
```

1.25 More miscellaneous graphic commands

```
                More commands


                 Set Ntsc
                 - Switches to the 60Hz NTSC mode

                 Set Pal
                 - Switches back to 50Hz PAL mode

                 Set Sprite Priority
                 - Changes the sprite priority in Dual playfield mode
```

1.26 Functions of advanced users

```
                Functions for advanced users

Here are some more commands for Assembler and C freaks.


                 =Scrn Rastport
                 - Returns the screen rastport address

                 =Scrn Bitmap
                 - Returns the screen bitmap address

                 =Scrn Layer
                 - Returns the screen layer address

                 =Scrn Layerinfo
                 - Returns the screen layer info address

                 =Scrn Region
                 - Returns the screen region address
```

1.27 Four player adapter, second mouse and pad support

```
                Gameport peripherial support

AMCAF enables you to use an optional connected four player adapter as well
as a second mouse connected to joystick port 1. Moreover, you can read out
the second fire button of a joystick or if the lowlevel.library is present
even the other 5 buttons on the pad.


                 4-Player Adapter commands
                 > Support commands for the four player adapter

                 Second mouse commands
                 > Enables second mouse usage


                 =Xfire
                 - Reads out the fire buttons on a gamepad
```

1.28 Various other commands and functions

```
                Various other commands and functions

Most of the commands have now been mentioned, leaving us with the bits and
pieces which cannot be clearly divided into a certain group.

Commands:


                 Exchange Bob
                 - Swaps the two images in the sprite bank

                 Exchange Icon
                 - Swaps the two images in the icon bank

                 Audio Lock
                 - Reserves the audio device

                 Audio Free
                 - Frees the audio device

                 Open Workbench
                 - Reopens the workbench again

                 Flush Libs
                 - Frees as much as possible memory

                 Write Cli
                 - Writes something into the cli window

                 Reset Computer
                 - Resets your computer

                 Nop
                 - No effect

Functions:


                 =Aga Detect
                 - Checks if the computer has AGA chipset

                 =Scanstr$
                 - Returns the name of a key

                 =Command Name$
                 - Acquires the name of the program
```

```
=Tool Types$
- Reads the Tool Types of an icon

=Amos Cli
- Returns the CLI number of AMOS.

=Speek
- Peeking a signed
byte

=Sdeek
- Deeking a signed
word

=Cpu
- Returns the number of the fitted CPU

=Fpu
- Acquires the id number of an coprocessor

=Nfn
- No effect

=Cop Pos
- Returns the current address of the copper list
```

1.29 Pix Shift commands group

```
                Pix Shift commands group

Using these commands you can increase or decrease the colour indexes in a
rectangular area. These instructions work pixelwise and therefore makes it
possible to limit the colours, and so the Pix Shift commands are slower
than
                 Shade Bobs
                .


                 Pix Shift Up
                 - Increase colour indexes (
                 cyclic
                )

                 Pix Shift Down
                 - Decrease colour indexes (
                 cyclic
                )

                 Pix Brighten
                 - Increase colour indexes (not
                 cyclic
                )
```

```
                 Pix Darken
                 - Decrease colour indexes (not
                 cyclic
                )

                 Make Pix Mask
                 - Picks up a mask for the shifting process


                 Shade Pix
                 - Plots a shade pixel
```

1.30 Commands to replay protracker musics

```
                Commands to replay protracker musics

As the AMOS Tracker commands are really crap and contain a lot of bugs,
of course there is a replacement for them in the AMCAF extension.

Since version 1.1 there are new commands even to replay sound effects
during the normal protracker music. The AMOS Music Extension is rather
obsolete now.

Here a brief list of all advances of the AMCAF commands:
- CIA timing or VBL timing.
- Volume control.
- Channel toggling.
- Vu meters.
- Possibility to receive signals from the module.
- Support of all Protracker effect commands.
- Playback of sound effects along with the music.

Commands:


                 Pt Play
                 - Replays a module

                 Pt Stop
                 - Stops the current music

                 Pt Continue
                 - Restarts a previously stopped music

                 Pt Volume
                 - Setting the volume of the music

                 Pt Voice
                 - Toggling the audio channels

                 Pt Cia Speed
                 - Changing the replaying speed
                 Pt Raw Play
                 - Plays a chunk of memory as sound sample

                 Pt Bank
                 - Sets the bank for the use with Pt Instr Play

                 Pt Instr Play
                 - Plays an instrument of a protracker module

                 Pt Sam Bank
                 - Sets the bank to use with AMOS samples

                 Pt Sam Play
                 - Replays a sample from an AMOS Sam Bank

                 Pt Sam Stop
                 - Stops the sfx on specific audio channels

                 Pt Sam Volume
                 - Sets the volume of a sound effect

                 Pt Sam Freq
                 - Changes the replaying speed of a sample

Functions:


                 =Pt Vu
                 - Returns the current Vumeter value

                 =Pt Signal
                 - Checking for signals from the music


                 =Pt Cpattern
                 - Gets the current song position

                 =Pt Cpos
                 - Returns the current pattern line

                 =Pt Cinstr
                 - Returns the current instrument being played

                 =Pt Cnote
                 - Gets the frequency of the current instrument


                 =Pt Instr Address
                 - Returns the address of an instrument

                 =Pt Instr Length
                 - Returns the length of an instrument

                 =Pt Data Base
                 - Gets the address of the PT-DataBase
```

1.31 Processor Tile commands group

Processor Tile (Ptile) commands group

PTiles were thought to be a replacement for
TOME
16x16 tiles. Instead of
using the
Blitter
, the processor is used which is even with a MC68000
fitted much faster at such small objects. The speed improvement is even
higher when using accelerater oards and
Fast ram
.

Until now there are only two commands, but that might be changing in
further versions of AMCAF.

Advantages:
- Very fast.
- No
Chip ram
required.

Disadvantages:
- Only tile sizes of 16x16 allowed.
- Only placeable in distances of 16x16 pixels.
- Custom bank format (no
Icon banks
)

Commands:

Ptile Bank
- Setting the Ptile bank

Paste Ptile
- Plotting a Ptile

1.32 Shade Bobs commands group

Shade Bobs commands group

Shade Bobs aren't really bobs, but "in the scene" this is the common name
for such effects. Shade Bobs increase or decrease colour values of the
pixel they are placed on. This is rather similar to colour cycling,
however using Shade Bobs the pixels on the screen are affected and not the
palette entries! When using Shade Bobs you cannot limit the colours to a
certain range but only the amount of bitplanes that will be used to be

cycled

```
                 through. Additionally, Shade Bobs may leave the screen boundaries ↩
                     .


                 Shade Bob Mask
                 - Determinate the image to use for the bobs

                 Shade Bob Planes
                 - Setting the number of bitplanes to use

                 Shade Bob Up
                 - Places a Shade Bob that increases the colours

                 Shade Bob Down
                 - Places a Shade Bob that decreases the colours
```

1.33 Commands to utilisize a second mouse

```
                Commands for a mouse in port 1

On request of some users, AMCAF now supports the reading of a second mouse
in joystick port 1. Therefore you can now realize your games which require
a second mouse in a two player mode.

Commands:


                 Limit Smouse
                 - Defines the movement region

                 Smouse Speed
                 - Sets the speed of the mouse

                 Smouse X
                 - Sets the x coordinate of the mouse

                 Smouse Y
                 - Sets the y coordinate

Functions:


                 =X Smouse
                 - Reads the x coordinate

                 =Y Smouse
                 - Reads the y coordinate of the mouse

                 =Smouse Key
                 - Checks the mouse key
```

1.34 Splinters commands group

```
                Splinters commands group

Splinters are similar to Td Stars, but they don’t destroy the background
and use the colour of the pixel they have removed and Splinters require a
list of coordinates. Each coordinate requires four bytes, i.e already a
field of 16x16 coordinates consumes 16 KB of memory.

Commands:


                 Coords Bank
                 - Reserves a bank to store the coordinates

                 Coords Read
                 - Reading the coordinates into a bank

                 Splinters Bank
                 - Reserves memory for the splinters

                 Splinters Colour
                 - Sets the to-use colours

                 Splinters Limit
                 - Changes the limits of the splinters

                 Splinters Max
                 - Sets the maximum of new appearing splinters

                 Splinters Fuel
                 - Sets the range of a splinter

                 Splinters Gravity
                 - Changes the gravity

                 Splinters Init
                 - Initialises the splinters bank

                 Splinters Do
                 - Do a complete drawing process

                 Splinters Del
                 - Clears the splinters

                 Splinters Move
                 - Moves the splinters

                 Splinters Draw
                 - Draws the splinters to the screen

                 Splinters Back
                 - Gets the background pixels

Functions:
```

```
                 =Count Pixels
                 - Counts the number of pixels in a specific area

                 =Splinters Active
                 - Returns how many splinters are still moving
```

1.35 Various string and integer supporting functions

```
                Various string and integer supporting functions

Functions:


                 =Binexp
                 - Exponential function on basis of two

                 =Binlog
                 - Logarithmic function on basis of two

                 =Lsl
                 - Quick multiplication by a power of two

                 =Lsr
                 - Quick division by a power of two

                 =Wordswap
                 - Swapping the upper and lower 16 bits

                 =Qarc
                 - Fast arc function

                 =Qsin
                 - Fast sine function

                 =Qcos
                 - Fast cosine function

                 =Qrnd
                 - Fast replacement for Rnd

                 =Qsqr
                 - Fast replacement for Sqr

                 =Asc.w
                 - Converting a
                 Word string
                 into a number

                 =Asc.l
                 - Converting a
                 Long string
                 into a number
```

```
=Vclip
- Restricts a value to a given range

=Vin
- Tests, if the value is within a range

=Vmod
- Does a modulo operation on a value

=Even
- Returns, if a number is even

=Odd
- Returns, if a number is odd


=Pattern Match
- Compares a string with a certain pattern

=Chr.w$
- Creating a
Word string

=Chr.l$
- Creating a
Long string

=Lzstr$
- Returns a right ajusted number with leading zeros

=Lsstr$
- Returns a right ajusted number

=Insstr$
- Inserts a string into a string

=Cutstr$
- Cuts out a piece of a string

=Replacestr$
- Replaces a string with another one

=Itemstr$
- Returns an 'item' contained in a string
```

1.36 Td Stars commands group

```
                Td Stars commands group

Using these commands you not only can realize 3D star effects, because the
commands are rather versatile.


                 Td Stars Bank
                 - Reserves some memory for the stars

                 Td Stars Planes
                 - Selects the planes to be used for the stars

                 Td Stars Limit
                 - Sets the limits of the stars

                 Td Stars Origin
                 - Places the origin of the stars

                 Td Stars Gravity
                 - Determins the gravity force

                 Td Stars Accelerate
                 - Toggles the acceleration

                 Td Stars Init
                 - Inits the stars

                 Td Stars Do
                 - Does a complete drawing process

                 Td Stars Del
                 - Clears the stars from the screen

                 Td Stars Move
                 - Moves the stars

                 Td Stars Draw
                 - Draws the stars
```

1.37 Vector rotation commands

```
                Commands to rotate three dimensional points.

At first: THIS SHOULD NOT BE A REPLACEMENT FOR THE 3D-EXTENSION...
At least not at this time. Nevertheless you can create vector graphics
without problems which is proved by the example programs.

Commands:


                 Vec Rot Pos
                 - Positions the camera

                 Vec Rot Angles
                 - Sets the viewing angles

                 Vec Rot Precalc
                 - Calculates the precalc matrix

Functions:
```

```
                 =Vec Rot X
                 - Calculates the 2D X value

                 =Vec Rot Y
                 - Determinates Y value

                 =Vec Rot Z
                 - Returns the Z coordinate
```

1.38 Time and date functions

```
                Time and date functions

These commands are used to convert a disk object or system time.

Functions:


                 =Current Time
                 - Acquires the current time

                 =Ct Time$
                 - Creates a complete time string

                 =Ct Hour
                 - Extracts the hour from a time

                 =Ct Minute
                 - Returns the minute of a time stamp

                 =Ct Second
                 - Calculates the second of a time

                 =Ct Tick
                 - Extracts the 1/50 from the time

                 =Ct String
                 - Evaluates a string into a time code


                 =Current Date
                 - Acquires the current date

                 =Cd Date$
                 - Creates a complete date string

                 =Cd Year
                 - Extracts the year from a date

                 =Cd Month
                 - Calculates the month

                 =Cd Day
```

```
- Returns the day of the date

=Cd Weekday
- Gets the weekday from the date stamp

=Cd String
- Converts a string into the date stamp
```

1.39 Overview over all additional information

```
General bank aspects

Icon and sprite banks

Permanent banks

Temporary banks

Chip ram

Fast ram

Ranger ram

Byte

Word

Long word

Cyclic

Aga-Amigas

HAM-Mode

Disk object

Blitter

Blitter minterms

Protection flags

TOME

4-Player adapter

AMCAF-Structure

Bitplanes
```

1.40 Information about the new A1200/A4000/CD^{32}

```
                Information about AGA-Amigas

The new Amigas A1200, A4000 and CD$^3$$^2$ have the AGA-Chipset. This new
chipset makes is possible not only to display 6 Bitplanes but even 8

                 Bitplanes
                 in nearly every resolution. These Amigas have 12 additional
colour bits to the normal 12 bits ($0RGB) and therefore can use a 24 bit
value ($00RRGGBB). Even if AMOS Pro V2.0 does not currently support AGA,
nevertheless AMCAF contains some commands for the future implementation.

The most important advantages of AGA-Amigs:
- Locale library: Many programs can use different languages
- Kickstart 3.0 with improved graphic routines
- Many, new, high resolutions
- up to 256 colours in nearly every resolution
- new
                 HAM8-Mode
                 for over 262.000 colours
- 16 colour Dual Playfields possible
- MC68020+ for more processor power
- 2 MB
                 Chip ram
                My advice: If you still have an old Amiga you should think about  ↩
                   buying a
new Amiga, it’s worth it!
```

1.41 Banks

```
                A few bits of information about banks

AMOS banks are a linear block of memory (there are two
                 Exceptions
                ). In
these chunks various kinds of data is stored. AMOS uses them mostly for
packed graphics, sounds, musics, AMAL programs, menus, resources and other
data.

Generally, there are four main types of banks:
- Banks, that are stored in
                 Chip ram
                 and remain there
                 permanently
                - Banks, that are stored in
                 Fast ram
                 and remain there
                 permanently
                - Banks, that are stored in
                 Chip ram
                 and remain there
                 temporarily
                 only
- Banks, that are stored in
                 Fast ram
                 and remain there
                 temporarily
                 only
```

1.42 What bitplanes are and what you can do with them

```
                Bitplanes

Let’s go for a small tutorial, in which bitplanes and how they work is
explained.

1. What is a bitplane?

   On Amiga the video picture is created by so called bitplanes. These are
   linear blocks in chip memory, of which every single bit represents one
   dot on the screen.

   Using only one bitplane, you can only see two different colours (either
   bit set or bit clear). If more than one bitplane is placed on top of
   others, you will get 2^n colours for n bitplanes.

   Example:

   We have an 16 colours screen. So it has got 4 bitplanes. Normally, the
   bitplanes are counted from 0 to n-1...

   Binary 0 1 0 1 = Decimal 5
          | | | |    .---v---v---v---v--------------------
          | | | `----+ 1 |   |   |   | Bitplane 0
          | | |      |--v^--v^--v^--v^--v-------------------
          | | `------+--+ 0 |   |   |   | Bitplane 1
          | |        |  |--v^--v^--v^--v^--v------------------
          | `--------+--+--+ 1 |   |   |   | Bitplane 2
          |          |  |  |--v^--v^--v^--v^--v-----------------
          `----------+--+--+--+ 0 |   |   |   | Bitplane 3
                     |  |  |  |---^---^---^---’
                     |  |  |  |
                     |  |  |
                     |  |
                     |

   You must imagine the bitplanes to be ’overlapped over each other’. The
   colour index 5 is displayed in the bitplanes like that. Which colour
   the dot on the screen finally has, is determinated by the palette
   settings.

2. Overlapping and transparency.

   Imagine, you draw a figure A in bitplane 0 and a figure B in bitplane 1
   only. This produces the following:
```

The area, on which none of the figures are displayed, has got the colour zero (both bitplanes are cleared: %00), whereas on regions, where only figure A is appearing, the colour 2^0=1 is used (bit 0 is set, bit 1 is clear: %01). Figure B alone sets the bit in bitplane 1 and therefore is displayed in colour 2^1=2 (=%10). If both figures are overlapping, the bits in both bitplanes are set and this results in colour 2^0+2^1=3 (=%11).

Now we can define the palette. For instance:

```
Palette 0,$F00,$F0,$FF0
```

The background is black ($000), figure A is red ($F00) and figure B is green ($0F0), when overlapping yellow is generated ($FF0).

By just thinking about these corelation you can achieve nice effects.

Try to find out, how the two figures will look like, when using these palettes:

1. Palette 0,$FFF,$888,$FFF
2. Palette 0,0,0,$FFF

You find the solution below.

3. Glenz and Fade

When e.g a cube appears 'transparent', out of glas or electric, the effect is called 'Glenz'. This is mostly used for vector effects. There are two major types of Glenz vectors:

a) Wire frame objects:

With Glenz wire frame objects the lines are drawn in a different bitplane per vertical plane keeping the two or three previous frames intact. These old and new frames then overlap, and by choosing the right palette (additive colour values per bitplane) the points where the lines are overlapping look lighter and somehow glitter.

Example for an additive colour palette (eight coloured screen):

```
DARK=$333 : LIGHTER=$666 : BRIGHT=$FFF


'    %000 %001 %010   %011  %100   %101    %110   %111
Palette 0,DARK,DARK,LIGHTER,DARK,LIGHTER,LIGHTER,BRIGHT
```

Just look at the bit values and count the number of setted bits to get the right colour value.

By changing the palette permanently you can also create a Motion Blur or Fade effect. To achieve this, you must set every colour with the bit for the current bitplane, in which you are drawing at the moment, to the brightest colour, all colours with the bit of the previous bitplane and not with the bit of the current one to a middle colour and so on...

b) Solid objects:

Glenz on solid object is done like this: the polygons, that are facing away from the viewer (and therefore cannot actually be seen) are drawn on an other bitplane than the polygons that face the viewer. By setting the colours according to the bitplanes, the object will seem to be transparent. The only thing to do is to mix the colours of the bitplanes, in which the different polygons are drawn.

4. Bitplane modes and their specialities.

   The old ECS-Amigas can only display up to 6 bitplanes simultaneously. So 2^6=64 colours is the maximum (excluding the HAM mode).
   Though there are some special modes:

   a) ExtraHalfBright (EHB). As the OCS and ECS chipset has only 32 colour registers, the other 32 colours are displayed at half the value. When using EHB you can produce some neat shadow effects by writing into the 6th bitplane. Note: EHB pictures cannot be faded out perfectly.

   b) Hold And Modify (HAM). Some method, to decompress six bitplanes to twelve bitplanes by hardware. Therefore the colours from 16 to 63 are used reach the wanted colour by changing the red, green or blue value of previous colour. Very good for static pictures and precalculated animation but useless for games and realtime graphics. The only sense-making oportunity to display moving objects on a HAM screen is to use sprites.
   Also see HAM mode .

   c) Dual Playfield. The Amiga chipset can display all even bitplanes (0,2,4) seperated from the odd ones (1,3,5). It's better to say, he puts the one playfield on top of the other by using colour 0 as 'window' to the other playfield. In this mode each playfield can have 2^3=8 colours.

   The chipset has got seperate control registers for even and odd bitplanes each, as each Playfield must be independant for Dual Playfield mode.

   - The BitPLane CONtrol (BPLCON0) $100.
     Here you determine the number of bitplanes and the resolution and can toggle the special modes. Bit table:

15	HIRES	Toggle hires mode
14-12	BPUx	Number of bitplanes
11	HOMOD	Toggle HoldAndModify
10	DBPLF	Toggle Dual Playfield
9	COLOR	Toggle colour burst output
8	GAUD	Use audio input from a genlock
7	8BPL	8 bitplanes (AGA)
6	SHIRES	Superhires (ECS/AGA)
3	LPEN	Activate lightpen
2	LACE	Enable interlace mode

```
        1      ERSY           Switch to external synchronization
```

- The scroll register (BPLCON1) $102.
  By using this register, the screen can be scrolled to the left by up to 15 pixels. The bits 0-3 are used for the even bitplanes, the bits 4-7 for the odd ones.

- The modulo registers (BPL1MOD/BPL2MOD) $108/$10A.
  These registers set the amount of bytes to be added to the memory of the bitplanes after each rasterline. This is utilized by playfields that are bigger than the visible area. When writing a negative value you can achieve vertical zoomers or mirror effects.

If you want to alter a register using Set Rain Colour you can calculate the new 'colour' with the following formula: (REGADR-$180)/2

Note: Enabling 5 to 6 bitplanes in low resolution or 3 to 4 bitplanes in high resolution will cost free processor time, even if you only display the screen. This is true when the running program is placed in chip ram or must access chip ram.

5. How can I access all these effects?

Simple: Define a rainbow, call
              Set Rain Colour
              and enter
the new values for the registers using Rain() instead of supporting the colours. The only limitation: Due to the AMOS rainbow restrictions you can only manipulate ONE (!) register per rasterline.

Look carefully at the example programs. These demonstrate every single effect mentioned here.

By the way: You can get the pointer to the single bitplanes using Logbase(planenr) and Phybase(planenr).

Enjoy!

Solution to the questions in 2:

1. Palette 0,$FFF,$888,$FFF
   Figure A is white and is moving 'over' figure B, which is grey, because if they overlap the colour white is created.

2. Palette 0,0,0,$FFF
   Figur A and Figur B are invisible as long as they down overlap. Only if both are placed over each other colour 3 is created which was set to $FFF (white).

1.43 The Blitter-Chip

The Blitter-Chip

The Blitter is a coprocessor inside the Amiga which is mainly used to copy and combine data (therefore BLockImageTransferER). Additionally, it can fill polygons and draw lines. The Blitter is rather fast at this and works with an unbelevable speed of up to 16 million pixels per second. All data that is accessed by the Blitter chip must be in

Chip ram

.

AMOS uses the blitter for Bobs, Icons, Screen Copy and many other commands. The MC68020 and higher is much faster if only need to copy data.

The Blitter works at word boundaries and this results in cutting down the X coordinates to the nearest multiple of 16.

To fill a polygon using the Blitter, the lines must be only one pixel thick. This is the reason why there are two different ways to draw lines.

Also see:

Turbo Draw

Bcircle

Blitter Fill

Blitter Copy

Blitter Clear

Blitter minterms

1.44 Description to Blitter minterms

Blitter minterms:

The Blitter chip knows 256 different copying and combining modes. These determinated in two steps:

1. Eight different boolean terms are used upon the three databits. Each of the terms returns true on a specific combination of A, B and C.
2. These eight results of the terms are combined together using a logical OR. This result is the target bit D.

Bit	Minterm	Input-Bit
0	$\overline{ABC}$	000
1	$\overline{AB}C$	001
2	$\overline{A}B\overline{C}$	010
3	$\overline{A}BC$	011
4	$A\overline{BC}$	100

5 $A\overline{B}C$ 101

6 $AB\overline{C}$ 110

7 ABC 111

Procedure:
1. At which of the eight combinations of ABC should D be true?
2. Now set the bits of the bitmask.
3. If not all of the three source data streams are required, every combination with the unused bits and the desired bits must be chosen.

Also see:

Blitter Copy

1.45 Byte

A byte

A bytes has got 8 bits, therefore you can display values from 0 up to 255. A byte normally has got the suffix '.b' (in assembler). Two bytes together create a
Word
, four a
Longword
.

1.46 Chip ram

Chip ram

On Amiga computers, chip ram is the area in memory, which is accessed by both custom chips and the CPU. At the moment, the size of chip ram is limited to 2 MB (even on A1200 and A4000), and old A500 do only have 512 KB chip ram by default. For that reason your program should not use more than this 512 KB of chip memory. However, this does NOT mean, that your program needs to run on 512 KB total memory!

As custom chips access the memory at the same time with the processor, the CPU is slowed down, if the program is held in chip ram. This effect is even more severe if the screen uses more than 16 colours in lowres or more than 8 colours in hires (not A1200/A4000).

Chip ram is mainly required for screens, bobs and sprites, music and sound effects, floppy disk drives and copperlists.

Also see:

Fast ram

Ranger ram

1.47 Cyclic

Cyclic

If a number reaches the upper boundary, the number is set to the lower boundary value and vice versa.

Example:

```
UB=31 (upper boundary)
LB=1  (lower boundary)
N=3   (number)
Do
  Inc N
  If N>UB Then N=LB
  If N<LB Then N=UB
  Print N
Loop
```

1.48 Fast ram

Fast ram

Fast ram is the memory area which the custom chips don’t have access to.
So the processor is not slowed down, if the program runs in fast ram.
BUT: You must not store any datas for bitplanes, bobs and sprites, music or sound effects and then try to access it by the custom chips. This has very unpleasant if not lethal results in most cases.

All Amigas excluding the A3000/A4000 do not have fast ram mounted by default.

Also see:

Chip ram

Ranger ram

1.49 How to build a four player adapter

Construction plan for a 4-player adapter

Remark: I do NOT guarantee the correctness of this plan, nor I am responsible for any damage done to your computer if you're doing something wrong. If you don't feel skilled enough to build the adapter yourself, you will find it in better electronic shops for a bargain.

```
Requirements: 1 25 pins male parallel socket.
              2 9 pins female joystick sockets.
              1 25 lines cable.
```

Blueprint of the parallel and joystick sockets.

```
                              Parallelport
  ____________________________________________________________________
 (                                                                    )
  \  1    2    3    4    5    6    7    8    9   10   11   12   13   /
   \   14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22   23 | 24   25 |  /
    \___|_|____|____|____|__|_|____|____|____|_________|_________|_/
        | |    |    |    |  | |    |    |    |         |         |
        | |    | .--|----|--^-|----|----|----|---------|-.       |
        `-|----|-|--|-v--|----|----|----|----|------.  `-|-------|-.
          |    | |  | |  |    |    |    |    |      `----|----.  | |
          |  .-|-|--|-|--' .--|----|----|----|-----------|----|--' |
          |  | | | /  |    |  `--. |    |    `--------.  |    |    |
          |  | `-|-|--|-.  |     | |    `-------------|--|-.  |    |
          |   \  | |  | |  |     | `------------------|--|-|--|-.  |
          `---|--|-|--|-|--|-.   `--------------------|--|-|--|-|--|-.
      ________|__|_|__|_|__|_|___             ________|__|_|__|_|__|_|___
     (        |  | |  | |  | |   )           (        |  | |  | |  | |   )
      \  5    4  | 3  | 2  | 1  /             \  5    4  | 3  | 2  | 1  /
       \    9    8    7    6   /               \    9    8    7    6   /
        \_____________________/                 \_____________________/

                 Joy 3                                   Joy 4
```

For the ones that think this plan is to confusing:

Parallelport:		Joy 3:		Joy 4:	
P0	2			UP	1
P1	3			DOWN	2
P2	4			LEFT	3
P3	5			RIGHT	4
P4	6	UP	1		
P5	7	DOWN	2		
P6	8	LEFT	3		
P7	9	RIGHT	4		
BUSY	11	FIRE	6		
SEL	13			FIRE	6
+5 V	14	+5 V	7	+5 V	7
GND	18	GND	8	GND	8

As already said, there's no guarantee that it will work (although mine perfectly does!).

1.50 Short description to HAM

Short explanation to the Amiga HAM-mode.

As the Amiga can only display 6 or 8 bitplanes and you need much more colours than 64 or 256 colours to achieve photo quality, the amiga developers invented a new tricky display mode: The HAM-Mode (abbreviation for "Hold And Modify").

- HAM6 (without AGA chipset):
  The first 16 colours can be set as normal, these are displayed on the screen as usual.
  The colours 16 to 31 modify the blue part of the last colour to the left. Same with the colours from 32 to 47, which alter the green value and the colours from 48-63 change the red value accordingly. So you can display all the 4096 colours using only 6 bitplanes.

- HAM8 (AGA chipset required):
  Like the HAM6 mode the first colours are displayed correctly, but in HAM8 the base palette has got 64 colours. The rest of the colours from 64 to 255 are responsible for the modification of the previous colour like shown above. Using this technique and a intelligent base palette you can display every colour in the 16777216 colours big palette.

1.51 Icons

Icons- and sprites banks

These banks do NOT consist of an linear block of memory. Therefore you neither can move, copy, encode, pack them nor make a checksum from them. In addition, icon and sprites banks must always be in Chip ram .

1.52 Longword

A longword

Four bytes result in a unit of 64 bits. These units are called longwords and are used to address up to four gigabyte of data. For that reason they are used for every absolute address in memory. As with Words , they must lie on a even address boundary.

Also see:

Byte

1.53 Disk object

Disk object

A disk object can be:
a) a file on a drive
b) a directory
c) an assign
d) a volume

1.54 Permanent

Permanent banks

This banks are called permanent, because they survive calls to Default, Erase Temp and the start of a prigram. Moreover, they are saved along with the program. Normally, permanent banks have the name ’Datas’.

Also see:

Temporary banks

1.55 The disk object protection flag

The protection flags

These flags contain information about the type of a certain
Disk object
.

The protection value consists of following bits:

```
Bit 0=0: File can be erased
Bit 1=0: File is executeable
Bit 2=0: File can be overwritten
Bit 3=0: File can be read
Bit 4=1: File has not been changed after copying
Bit 5=1: Executeable can be made resident.
Bit 6=1: File is an Amiga-DOS script
Bit 7=1: File is hidden (does not actually work)
```

As you see, this is rather chaotic. So you’re advised to use the function Object Protection$ to convert this bitmap into a String in the format "hsparwed".

Also see:

Protect Object

=Object Protection

=Object Protection$

1.56 Ranger ram

Ranger ram

Ranger ram is a special type of ram: it is neither chip nor fast ram and does only exist on A500s with trap door slot memory expansion. The custom chips cannot access this ram, although the memory is not faster et al.

There's a way to make ranger ram to chip ram. Just resolder jumper 2 on the main board and switch off the memory expansion. I am not responsible for any possible damage that may occure by doing this.
Also see:

Chip ram

Fast ram

1.57 The interior structure of the AMCAF data base

```
Version: V1.15
Length : 1964 Bytes
Note   : Contrary to most other extensions, the data base is not
         kept in the AMCAF.Lib file. Therefore AMCAF is very compact
         (*only* 40 KB!?).

                rsreset         ;Stars
St_X            rs.w    1       ;0
St_Y            rs.w    1       ;2
St_DbX          rs.w    1       ;4
St_DbY          rs.w    1       ;6
St_Sx           rs.w    1       ;8
St_Sy           rs.w    1       ;10
St_SizeOf       rs.b    0       ;12

                rsreset         ;Splinters
Sp_X            rs.w    1       ;0
Sp_Y            rs.w    1       ;2
Sp_Pos          rs.l    1       ;4
```

```
Sp_DbPos        rs.l    1       ;8
Sp_Sx           rs.w    1       ;12
Sp_Sy           rs.w    1       ;14
Sp_Col          rs.b    1       ;16
Sp_BkCol        rs.b    1       ;17
Sp_DbBkCol      rs.b    1       ;18
Sp_First        rs.b    1       ;19
Sp_Fuel         rs.w    1       ;20
Sp_SizeOf       rs.b    0       ;22

                rsreset         ;Blitterqueue
Bn_Next         rs.l    1
Bn_Function     rs.l    1
Bn_Stat         rs.w    1
Bn_Dummy        rs.w    1
Bn_BeamPos      rs.w    1
Bn_CleanUp      rs.l    1
Bn_B40l         rs.l    1       ;BLTCON0&BLTCON1
Bn_B44l         rs.l    1       ;Masks
Bn_B48l         rs.l    1       ;Source Address C
Bn_B50l         rs.w    1
Bn_B52w         rs.w    1       ;Source Address A.w
Bn_B54l         rs.l    1       ;Target Address D
Bn_B58w         rs.w    1       ;BLTSIZE
Bn_B60w         rs.w    1       ;Modulo C
Bn_B62l         rs.w    1       ;Modulo B&A
Bn_B64w         rs.w    1       ;Modulo A
Bn_B66w         rs.w    1       ;Modulo D
Bn_B72w         rs.w    1       ;BLTBDAT
Bn_B74w         rs.w    1       ;BLTADAT
Bn_XPos         rs.w    1
Bn_SizeOf       rs.b    0

                rsreset         ;AMCAF Main Datazone
O_TempBuffer    rs.b    80
O_FileInfo      rs.b    260
O_Blit          rs.b    Bn_SizeOf
O_BobAdr        rs.l    1
O_BobMask       rs.l    1
O_BobWidth      rs.w    1
O_BobHeight     rs.w    1
O_BobX          rs.w    1
O_BobY          rs.w    1
O_StarBank      rs.l    1
O_StarLimits    rs.w    4
O_StarOrigin    rs.w    2
O_StarGravity   rs.w    2
O_StarAccel     rs.w    1
O_StarPlanes    rs.w    2
O_NumStars      rs.w    1
O_CoordsBank    rs.l    1
O_SpliBank      rs.l    1
O_SpliLimits    rs.w    4
O_SpliGravity   rs.w    2
O_SpliBkCol     rs.w    1
O_SpliPlanes    rs.w    1
O_SpliFuel      rs.w    1
```

```
O_NumSpli        rs.w    1
O_MaxSpli        rs.w    1
O_SBobMask       rs.w    1
O_SBobPlanes     rs.w    1
O_SBobWidth      rs.w    1
O_SBobImageMod   rs.w    1
O_SBobLsr        rs.w    1
O_SBobLsl        rs.w    1
O_SBobFirst      rs.b    1
O_SBobLast       rs.b    1
O_QRndSeed       rs.w    1
O_QRndLast       rs.w    1
O_PTCiaVbl       rs.w    1
O_PTCiaResource  rs.l    1
O_PTCiaBase      rs.l    1
O_PTCiaTimer     rs.w    1
O_PTCiaOn        rs.w    1
O_PTInterrupt    rs.b    22
O_PTVblOn        rs.w    1
O_PTAddress      rs.l    1
O_PTBank         rs.l    1
O_PTSamBank      rs.l    1
O_PTTimerSpeed   rs.l    1
O_PTDataBase     rs.l    1
O_PTSamVolume    rs.w    1
O_AgaColor       rs.w    1
O_HamRed         rs.b    1
O_HamGreen       rs.b    1
O_HamBlue        rs.b    1
                 rs.b    1        ;Pad
O_VecRotPosX     rs.w    1
O_VecRotPosY     rs.w    1
O_VecRotPosZ     rs.w    1
O_VecRotAngX     rs.w    1
O_VecRotAngY     rs.w    1
O_VecRotAngZ     rs.w    1
O_VecRotResX     rs.w    1
O_VecRotResY     rs.w    1
O_VecRotResZ     rs.w    1
O_VecCosSines    rs.w    6
O_VecConstants   rs.w    9
O_BlitTargetPln  rs.l    1
O_BlitSourcePln  rs.l    1
O_BlitTargetMod  rs.w    1
O_BlitSourceMod  rs.w    1
O_BlitX          rs.w    1
O_BlitY          rs.w    1
O_BlitWidth      rs.w    1
O_BlitHeight     rs.w    1
O_BlitMinTerm    rs.w    1
O_BlitSourceA    rs.l    1
O_BlitSourceB    rs.l    1
O_BlitSourceC    rs.l    1
O_BlitSourceAMd  rs.w    1
O_BlitSourceBMd  rs.w    1
O_BlitSourceCMd  rs.w    1
O_BlitAX         rs.w    1
```

```
O_BlitAY         rs.w    1
O_BlitAWidth     rs.w    1
O_BlitAHeight    rs.w    1
O_PTileBank      rs.l    1
O_BufferAddress rs.l    1
O_BufferLength  rs.l    1
O_PowerPacker   rs.l    1
O_PPCrunchInfo  rs.l    1
O_DiskFontLib   rs.l    1
O_DirectoryLock rs.l    1
O_DateStamp     rs.l    3
O_OwnAreaInfo   rs.b    1
O_OwnTmpRas     rs.b    1
                rs.w    1       ;Pad
O_AreaInfo      rs.b    24
O_Coordsbuffer  rs.b    20*5
O_TmpRas        rs.b    8
O_FontTextAttr  rs.b    8
O_AudioPort     rs.b    32
O_AudioIO       rs.b    68
O_ChanMap       rs.w    1
O_AudioOpen     rs.w    1
O_AudioPortOpen rs.w    1
                rs.w    1       ;Pad
O_PaletteBufs   rs.w    32*8
O_ParseBuffer   rs.b    512
O_SizeOf        rs.l    0
```

1.58 Temporary

Temporary banks

Temporary banks only exist during the execution of the program, they are erased on every start, testing or saving process or using the commands Default or Erase Temp. Normally, temporary banks have the name 'Work'.

Also see:

Permanent banks

1.59 TOME extension

The TOME extension

TOME is a extension for AMOS Creator and recently for AMOS Professional too, which is dedicated to tile and map programming. These tiles are used in many games e.g Jump'n'Runs or strategy games. As the whole game map would be far to memory hungry when kept as standard bitmap, the main graphics are cut into small pieces, so that they can be used repeatingly. The map therefore only constists of one byte per postition which points to the corresponding tile.

The current version of TOME is TOME V4.30.

1.60 Word

A word

A word consists of two
Bytes
so that is 16 bits. With
these 16 bits you can store addresses or data up to 64KB (65536). A word
must be on a even memory address or machines with MC68000 CPU will crash
with Guru number $80000003. Since MC68020 this is not important, but you
never should assume that there is no MC68000 in the computer and use odd
addresses!

If you insist on using uneven addresses, check out the installed cpu using
the
Cpu function
, if a 68020 or higher is fitted.

Also see:

Longword

1.61 Function: =Aga Detect

flag=Aga Detect

This is a simple function that checks for the existance of the AGA
chipset, found in A1200, A4000 and CD^3^{2} machines. It returns -1 (True),
if the new chipset is mounted or 0 (False) if the ecs or ocs chipsets
are still in use.

Note that it does not check for the kickstart version only, so A2000 and
A500 machines with kick 3.0 or higher will be detected as non-aga
correctly. This function is 100% system conform and won't hack the
hardware for it's result. Therefore, if the aga chipset had been disabled
either using the bootmenu or the "Setpatch NOAGA" command, it will return
that no aga is available.

1.62 Function: =Amcaf Base

adr=Amcaf Base

Returns the base address of the
AMCAF data base

```
                .

Also see:


                 =Amcaf Length
```

1.63 Function: =Amcaf Length

```
                le=Amcaf Length

Returns the length of the
                 AMCAF data base
                .

Also see:


                 =Amcaf Base
```

1.64 Function: =Amcaf Version$

```
Print Amcaf Version$

Returns the current AMCAF version number and a greetings list.
```

1.65 Function: =Amos Cli

```
                n=Amos Cli

This function gives back the number of the cli process out of which the
program/AMOS has been started off or zero, if AMOS has been started from
Workbench. This gives you the choice to either interprete options from the
command line or from the tool types of the appropriate icon.

Also see:


                 =Command Name$

                 =Tool Types$
```

1.66 Function: =Amos Task

```
address=Amos Task
```

For advanced programmers. This function returns the address of the AMOS task structure. Using this you can for example increase the task priority:

```
Areg(1)=Amos Task
Dreg(0)=3
dummy=Execall(-$12C)
```

This program sets the task priority to +3.

1.67 Function: =Asc.l

```
long=Asc.l(long$)
```

The Asc.l-function converts a
4 bytes
long string back into a number.
This value can range between -2147483648 and +2147483647. If 'long$' contains less than four characters, you will get an error message immediately.

Also see:

=Chr.l$

=Asc.w

=Chr.w$

1.68 Function: =Asc.w

```
word=Asc.w(word$)
```

Asc.w is used to convert a
word string
to a number value. Therefore the
result will be between 0 and 65535. If the length of 'word$' is less than two, the function is aborted and an error message is returned.

Also see:

=Chr.w$

=Asc.l

=Chr.l$

1.69 Function: =Bank Checksum

```
number=Bank Checksum(bank)
number=Bank Checksum(startaddress To endaddress)

This function calculates a checksum of a bank with specific contents.
Using this checksum you can find out if the contents of a bank has been
changed. The second version of this command calculates the checksum from
the memory area from startaddress to endaddress.
```

1.70 Function: =Bank Name$

```
                name$=Bank Name(bank)

The function Bank Name$ returns the name of bank number bank.

Also see:


                 Bank Name
```

1.71 Function: =Best Pen

```
pen=Best Pen($RGB)
pen=Best Pen($RGB,c1 To c2)

This function acquires the pen which is nearest to the colour $RGB.
Optionally, you can give the range to search for the best colour.
The Best Pen function can be used to recolour pictures with limited
palette.
```

1.72 Function: =Binexp

```
                v=Binexp(a)

This function calculates the result of the exponential function 2^a, but
is much faster than the normal AMOS expression. The parameter a must lie
between 0 and 31.

Examples: Binexp(1)=2, Binexp(3)=8, Binexp(16)=65536, Binexp(24)=16777216

Also see:


                 =Binlog

                 =Lsl
```

=Lsr

1.73 Function: =Binlog

```
a=Binlog(v)
```

Binlog is the reverse function to Binexp. It returns the logarithm to the value ’v’ with basis 2. ’v’ therefore must be a power of 2, otherwise you will get an error. Binlog is handy to get the number of bitplanes out from the amount of colours of a screen (exception: HAM).

Examples: Binlog(2)=1, Binlog(8)=3, Binlog(65536)=16, Binlog(16777216)=24

Also see:

=Binexp

=Lsl

=Lsr

1.74 Function: =Blitter Busy

```
flag=Blitter Busy
```

This function returns -1 (True), if the Blitter chip is currently busy and working on a job, e.g is currently clearing a bitplane or drawing a line etc. This can be used to wait for the end of the blitter activity, although Blitter Wait is preferred in this case. Or you could Blitter Busy to decide, if you want to do some more calculations using the processor or better start the next blitter activity.

Also see:

Blitter Wait

Blitter Copy

Blitter Fill

Blitter Clear

1.75 Function: =Blue Val

```
                b=Blue Val(rgb)
```

This function acquires the blue value of a colour. Together with the other Val functions, you can separate the colour into its three contents.

Also see:

=Red Val

=Green Val

=Glue Colour

1.76 Function: =Cd Date$

```
                date$=Cd Date$(date)
```

This simple function returns the date stamp 'date' as string in the format 'WWW DD-MMM-YY'.

Also see:

=Current Date

=Cd Year

=Cd Month

=Cd Day

=Cd Weekday

=Cd String

1.77 Function: =Cd Day

```
                day=Cd Day(date)
```

This function returns the day in the month of the date stamp parameter 'date'. The result 'day' will be a value between 1 and 31.

Also see:

=Current Date

```
                 =Cd Date$

                 =Cd Year

                 =Cd Month

                 =Cd Weekday

                 =Cd String
```

1.78 Function: =Cd Month

```
                month=Cd Month(date)
```

```
Cd Month calculates the month of the argument ’date’. ’month’ therefore
lies between 1 and 12.

Also see:


                 =Current Date

                 =Cd Date$

                 =Cd Year

                 =Cd Day

                 =Cd Weekday

                 =Cd String
```

1.79 Function: =Cd String

```
                date=Cd String(date$)
```

```
This function will evaluate the valid date in the date$ string and then
return the date stamp or -1, if the string has been invalid.

The string has to be in the format "DD-MMM-YY" or "DD-month-YY".
Strings like "Today" or "Tomorrow" are also allowed, weekday strings refer
to the last occurence of the week, i.e "Monday" represents last monday and
not next monday.

This command only works on OS2.0 and higher.

Also see:


                 =Current Date
```

=Cd Date$

=Cd Year

=Cd Month

=Cd Day

=Cd Weekday

1.80 Function: =Cd Weekday

```
weekday=Cd Weekday(date)
```

Returns the weekday of the parameter 'date'. 'weekday' can range between
1 (monday) and 7 (sunday).

Also see:

=Current Date

=Cd Date$

=Cd Year

=Cd Month

=Cd Day

=Cd String

1.81 Function: =Cd Year

```
year=Cd Year(date)
```

Returns the year out from the DOS date stamp 'date'.

Also see:

=Current Date

=Cd Date$

=Cd Month

=Cd Day

```
                 =Cd Weekday

                 =Cd String
```

1.82 Function: =Chr.l$

```
                long$=Chr.l$(long)
```

The Chr.l$ function converts a number into a 4 bytes string. 'long' can be any number you like. Using this technique, you can save numbers as normal strings.

Also see:

```
                 =Asc.l

                 =Chr.w$

                 =Asc.w
```

1.83 Function: =Chr.w$

```
                word$=Chr.w$(word)
```

This function converts a number into a 2 bytes string. The upper 16 bits of the value are ignored and therefore you should only use values from 0 to 65535.

Also see:

```
                 =Asc.w

                 =Chr.l$

                 =Asc.l
```

1.84 Function: =Command Name$

```
                fname$=Command Name$
```

Returns the file name of the program under which AMOS or the compiled program has been started. This is required for example to read the own Tool Types.

Also see:

=Tool Types$

=Amos Cli

1.85 Function: =Cop Pos

```
address=Cop Pos
```

If you create your own copperlist, you can use this function to remember the position of the next copper instruction. Later you can then write to this address directly to change the values of a copper instruction.

1.86 Function: =Cound Pixels

```
amount=Count Pixels(screen,colour,x1,y1 To x2,y2)
```

Counts the pixels in the rectangular are from x1,y1 to x2,y2 in the screen with number screen, that don't have the colour index colour. This function can be used to first acquire the number of points in this area and then to reserve a coordinates bank.

Also see:

Coords Bank

Coords Read

1.87 Function: =Cpu

```
chip=Cpu
```

The cpu function returns the identification number of the central processing unit currently installed your amiga. This number can currently lie between 68000 to 68060. Kickstart 1.3 knows only CPUs to the 68020.

Also see:

=Fpu

1.88 Function: =Ct Hour

```
hour=Ct Hour(time)
```

Separates the hour from the packed time.

Also see:

=Current Time

=Ct Time$

=Ct Minute

=Ct Second

=Ct Tick

=Ct String

1.89 Function: =Ct Minute

```
minute=Ct Minute(time)
```

Returns the number of minutes passed out from the longword ’time’.

Also see:

=Current Time

=Ct Time$

=Ct Hour

=Ct Second

=Ct Tick

=Ct String

1.90 Function: =Ct Second

```
second=Ct Second(time)
```

The Ct Second function returns the number of seconds from the parameter ’time’.

Also see:

=Current Time

=Ct Time$

=Ct Hour

=Ct Minute

=Ct Tick

=Ct String

1.91 Function: =Ct String

```
time=Ct String(time$)
```

This function will evaluate the valid time in the time$ string and then return the time as number or -1, if the string has been invalid.

The time string has to be in the format "HH:MM" or "HH:MM:SS".

This command only works on OS2.0 and higher.

Also see:

=Current Time

=Ct Time$

=Ct Hour

=Ct Minute

=Ct Second

=Ct Tick

=Ct String

1.92 Function: =Ct Tick

```
tick=Ct Tick(time)
```

Calculates the number of vertical blanks (=1/50 of a second) from the parameter ’time’.

Also see:

=Current Time

=Ct Time$

=Ct Hour

=Ct Minute

=Ct Second

=Ct String

1.93 Function: =Ct Time$

```
time$=Ct Time$(time)
```

This very handy function converts the normal parameter 'time' into a string in the format 'HH:MM:SS'.

Also see:

=Current Time

=Ct Hour

=Ct Minute

=Ct Second

=Ct Tick

=Ct String

1.94 Function: =Current Date

```
date=Current Date
```

Returns the current system date. This value counts the days passed since 1st January 1978. To separate the number of years, months and day there certainly are many supporting function.

Also see:

=Current Time

=Cd Date$

```
                  =Cd Year

                  =Cd Month

                  =Cd Day

                  =Cd Weekday

                  =Cd String
```

1.95 Function: =Current Time

```
                 time=Current Time
```

```
This function contains the current time. This is NOT a value in the
standard DOS-format as this one would require two longwords.

For everyone, who’s interested: the time is created out of
Wordswap(minutes)+ticks.

Also see:
```

```
                  =Current Date

                  =Ct Time$

                  =Ct Hour

                  =Ct Minute

                  =Ct Second

                  =Ct Tick

                  =Ct String
```

1.96 Function: =Cutstr$

```
                 newstr$=Cutstr$(s$,pos1 To pos2)
```

```
Cuts out the part of the string s$ from letter position pos1 to pos2.

Example:

Print Cutstr$("Hello dear Ben!",7 To 11) would generate ’Hello Ben!’.

Also see:
```

```
                  =Insstr$

                  =Replacestr$
```

1.97 Function: =Disk State

```
flags=Disk State(directory$)

Acquires the current state of a disk drive. 'flags' is a bitmap which
holds two bits:

Bit
 0=0: The disk is not write protected, you can write onto the disk.
 0=1: The volume is write protected or is currently being validated.
 1=0: Currently the disk is not in use, nobody is reading or writing.
 1=1: The disk is currently in use.

If no disk is in the drive, it normally should return -1, but I'm afraid
in this case a requester will be opened with the title "No disk in
drive xxx:" which creates the error 'File not found'.

Please use the following workaround:

Request Off
Trap FLAGS=Disk State(DIRECTORY$)
Request On
If Errtrap Then FLAGS=-1
```

1.98 Function: =Disk Type

```
type=Disk Type(directory$)

The Disk Type function returns the type of a directory. This value can be:

0: It is a real device (e.g. 'DF0:')
1: The path is a directory (assign) (e.g. 'LIBS:')
2: It is the name of a volume (e.g. 'Workbench2.0:')

Using this function you can filter specific disk types out of a device
list.
```

1.99 Function: =Dos Hash

```
hash$=Dos Hash$(file$)

Returns the hash value of a file. Only for advanced users who want to read
directly from dos disks.
```

1.100 Function: =Even

```
flag=Even(value)
```

Returns True (-1), if a number is even or False (0), if a number is odd.

Also see:

=Odd

1.101 Function: =Examine Next$

```
file$=Examine Next$
```

Gets information about the next
Object
in the directory and returns
its name. More information can be acquired using the object functions. If the end of the directory list is reached, ’file$’ will contain an empty string and the drawer will be closed.

Also see:

Examine Dir

Examine Stop

=Object Name$

=Object Type

=Object Size

=Object Blocks

=Object Protection

=Object Time

=Object Date

=Object Comment$

1.102 Function: =Extpath$

```
newpath$=Extpath$(directory$)
```

Adds a slash to a pathname, if it is a directory. Handy if you want to

```
add a filename to a path.

Examples: Extpath$("DH2:AMOS")+"AMOSPro" returns "DH2:AMOS/AMOSPro",
          Extpath$("DH2:")+"AMOS" returns "DH2:AMOS".

Also see:


                 =Path$

                 =Filename$
```

1.103 Function: =Filename$

```
                file$=Filename$(pathfile$)

Cuts the filename out of an mixed path and file string.

Example: Filename$("DH2:AMOS/AMOSPro") results in "AMOSPro".

Also see:


                 =Path$

                 =Extpath$
```

1.104 Function: =Font Style

```
                style=Font Style

This functions replaces the AMOS function Text Styles, because this one
does not return the multicoloured font bit (Bit 6). Apart from this,
Font Style is totally identical with the AMOS function.

Also see:


                 Change Font
```

1.105 Function: =Fpu

```
                chip=Fpu

This function returns the number of an installed mathematic coprocessor
or 0, if none is fitted. As before, Kickstart 1.3 only knows the 68881.
```

On 68040/68060 machines, the cpu contains the fpu, so 68040 or 68060 will
be returned instead.

Also see:

=Cpu

1.106 Function: =Glue Colour

```
rgb=Glue Colour(r,g,b)
```

Combines red, green and blue value of a colour to a normal colour value.

Also see:

=Red Val

=Green Val

=Blue Val

1.107 Function: =Green Val

```
b=Green Val(rgb)
```

This function acquires the green value of a colour. Together with the
other Val functions, you can separate the colour into its three contents.

Also see:

=Red Val

=Blue Val

=Glue Colour

1.108 Function: =Ham Best

```
c=Ham Best(newrgb,oldrgb)
```

As you cannot achieve the desired colour by plotting only one pixel in

HAM mode
, you can use the Ham Best function to search the best colour

for the next dot. 'newrgb' holds the target colour and 'oldrgb' the colour of the pixel exactly before the current dot. Please note that the current screen must be the HAM screen!

Also see:

=Ham Colour

=Ham Point

1.109 Function: =Ham Colour

```
newrgb=Ham Colour(c,oldrgb)
```

This function calculates the new colour value, that is created, when plotting a pixel in colour 'c' directly behind the last point.

Also see:

=Ham Best

=Ham Point

1.110 Function: =Ham Point

```
rgb=Ham Point(x,y)
```

This function simplifies the process of finding the right colour of a point at the coordinates x,y on a
HAM
screen. There's no need to
proceed from the left to the right until the right pixel value is found, although it the Ham Colour method is a little bit faster when processing a picture from left to the right, whereas Ham Point can access any point on the screen indiviually without preprocessing.

If the point x,y is not on the screen, rgb will contain -1 instead of the red-green-blue value of a pixel.

Also see:

=Ham Colour

=Ham Best

1.111 Function: =Insstr$

```
newstr$=Insstr$(a$,b$,pos)
```

Inserts the string b$ at the position pos into string a$.

Example:

Print Insstr$("Hello Ben!","dear ",6) creates 'Hello dear Ben!'.

Also see:

=Cutstr$

=Replacestr$

1.112 Function: =Io Error

```
errnumber=Io Error
```

The Io Error function gives back the number of the last DOS-error.

Also see:

=Io Error$

1.113 Function: =Io Error$

```
error$=Io Error$(errnumber)
```

Returns the string of the corresponding DOS error message. If no string for the error number exists, an empty string will be returned. When using Kickstart 2.0 not the interior list is taken but the localized strings of the system.

Also see:

=Io Error

1.114 Function: =Itemstr$

```
item$=Itemstr$(s$,itemnum)
item$=Itemstr$(s$,itemnum,sep$)
```

```
This is a very handy function that helps you to avoid little string arrays
where actually no are required. s$ contains a number of so called items,
which are numbered increasingly from zero onwards. Normally, the items are
seperated with a '|' character, but you can give your own single character
for seperation in the optional string sep$.

Empty strings for s$ are not allowed and will create an error message,
however, empty items can be used without hesitation. Trying to access a
item, that does not exist, will create an error aswell.

Example:

Print Itemstr$("Ben|Semprini|Petri|Andy",1) would return 'Semprini'.

Print Itemstr$("The quick brown fox",2," ") would return 'brown'.

Print Itemstr$("zero|one|two||four|five,5) would return 'five'.

Print "The weather is "+Itemstr$("great|fine|nice|not bad|rainy|awful|apocalyptic ↩
   ",WEATHER)

would have a similar effect to:

Dim W$(6)
For A=0 To 6
  Read W$(A)
Next
Print "The weather is "+W$(WEATHER)
[...]
Data "great","fine","nice","not bad","rainy","awful","apocalyptic"
```

1.115 Function: =Lsl

```
                nv=Lsl(v,n)

Rotates the number 'v' to the left by 'n' bits. I.e that a number 'v',
which is shifted one bit to the left contains the value v*2, with two bits
v*4, with 3 bits v*8 etc. This function is very quick and should be used
instead of multiplications whereever possible.

Also see:


                 =Lsr
```

1.116 Function: =Lsr

```
                nv=Lsr(v,n)

Shifts a number 'v' to the right by 'n' bits. This function does the same
```

```
as a division by 2^n, but is much quicker.

Also see:


                 =Lsl
```

1.117 Function: =Lsstr$

```
                s$=Lsstr$(v,n)

Similar to the AMOS function Str$, Lsstr$ creates a string out of a
number. However, with Lsstr$ the number will be created right justified
with ’n’ digits, leading zeros are replaced by spaces. The sign of the
number will not be printed. ’n’ must lie within 1 to 10.

Also see:


                 =Lzstr$
```

1.118 Function: =Lzstr$

```
                s$=Lzstr$(v,n)

This function is nearly identical to Lsstr$ with the difference that
leading zeros are not replaced by space characters. The parameter ’n’ sets
the number of digits the string should contain and can range from 1 to 10.

Also see:


                 =Lsstr$
```

1.119 Function: =Mix Colour

```
newrgb=Mix Colour(rgb1,rgb2)
newrgb=Mix Colour(oldrgb,addrgb,lrgb To urgb)

The first version of the function mixes the two colours ’rgb1’ and ’rgb2’
and returns the resulting colour value.

The second version behaves a little bit differently: The colour ’addrgb’
is added to the colour value ’oldrgb’, if ’addrgb’ is a positive value or
subtracted, if the value is negative. ’lrgb’ defines the lower and ’urgb’
the upper boundary of the allowed resulting colours.
```

1.120 Function: =Nfn

```
                dummy=Nfn

This 'function' returns nothing useful. It's only used, like Nop, in speed
testing routines.

Also see:


                 Nop
```

1.121 Function: =Odd

```
                flag=Odd(value)

Returns True (-1), if a number is odd or False (0), if a number is even.

Also see:


                 =Even
```

1.122 Function: =Object Blocks

```
                numblks=Object Blocks
numblks=Object Blocks(pathfile$)

Object Blocks returns the number of blocks the current or the
                 Object
                'pathfile$' uses on a specific volume.

Also see:


                 Examine Object

                 Examine Dir

                 =Examine Next$
```

1.123 Function: =Object Comment$

```
                comment$=Object Comment$
comment$=Object Comment$(pathfile$)
```

```
This function gives back the file note of an
                 Object
                .

Also see:


                 Set Object Comment

                 Examine Object

                 Examine Dir

                 =Examine Next$
```

1.124 Function: =Object Date

```
                date=Object Date
date=Object Date(pathfile$)

Returns the date stamp of an
                 Object
                 as numeric value, which can be
separated into year, month, day and weekday using the date functions.

Also see:


                 =Cd Date$

                 =Cd Year

                 =Cd Month

                 =Cd Day

                 =Cd Weekday

                 Set Object Date

                 Examine Object

                 Examine Dir

                 =Examine Next$
```

1.125 Function: =Object Name$

```
                file$=Object Name$
file$=Object Name$(pathfile$)
```

Object Name$ returns the name of the
Object
'pathfile$'. This function
is similar to the Filename$-Function, but acts quite differently as the existance of the file is checked, more information about the object is stored in the info block and the name is read out with correct case. The first version of Object Name$ gets the name of the object from the info block which has been read out earlier using Examine Object.

Also see:

Examine Object

Examine Dir

=Examine Next$

1.126 Function: =Object Protection

```
                prot=Object Protection
prot=Object Protection(pathfile$)
```

Object Protection returns the
Protection flags
of an
Object
. The
Object Protection$ function converts this numeric value into a string in the format "hsparwed".

Also see:

Protect Object

=Object Protection$

Examine Object

Examine Dir

=Examine Next$

1.127 Function: =Object Protection$

```
prot$=Object Protection$(prot)
```

Converts the

```
                 Protection Flags
                 into a string in "hsparwed" format.

Also see:


                 Protect Object

                 =Object Protection
```

1.128 Function: =Object Size

```
                length=Object Size
length=Object Size(pathfile$)

Returns the size of a Object. The first version copies the size directly
from the current info block, the second version fills the info block with
other datas of the
                 Object
                 'pathfile$'.

Also see:


                 Examine Object

                 Examine Dir

                 =Examine Next$
```

1.129 Function: =Object Time

```
                time=Object Time
time=Object Time(pathfile$)

Returns the time of the last write access to an
                 Object
                . This value
'time' can be converted into hours, minutes, seconds and ticks using
the time functions.

Also see:


                 =Ct Time$

                 =Ct Hour

                 =Ct Minute
```

=Ct Second

=Ct Tick

Set Object Date

Examine Object

Examine Dir

=Examine Next$

1.130 Function: =Object Type

```
                type=Object Type
type=Object Type(pathfile$)
```

This functions finds out if 'pathfile$' or the object whose data is currently in the info block is a directory or a file. In case of file 'type' is greater than 0 and smaller than 0 if the Object is a drawer.

Also see:

Examine Object

Examine Dir

=Examine Next$

1.131 Function: =Pal Get

```
colour=Pal Get(palnr,index)
```

This function reads the value of the colour with the number index 'index' from the internal stored palette buffer 'palnr'.

Also see:

Pal Set Screen

Pal Get Screen

Pal Set

1.132 Function: =Path$

```
                pathway$=Path$(pathfile$)
```

Returns the pathname part of a mixed path-filename string. Could be used as a kind of Parent$-Function too.

Example: Path$("DH2:AMOS/AMOSPro") returns "DH2:AMOS".

Also see:

=Filename$

=Extpath$

1.133 Function: =Pattern Match

```
flag=Pattern Match(sourcestring$,pattern$)
```

Pattern Match checks, if the string 'sourcestring$' matches the pattern 'pattern$'. Is this the case, so True (-1) will be returned otherwise False (0). The pattern may contain any regular DOS jokers a asterik (*) will be converted into '#?' automatically.

This command only works on OS2.0 and higher.

1.134 Function: =Pfire

```
                flag=Pfire(j)
```

Corresponds to the Fire(j) function for normal joysticks. Fire(j) tests, if the firebutton for the joystick in the parallel port is currently pressed. If this is the case, True (-1) will be returned otherwise False (0).

Also see:

=Pjoy

=Pjup

=Pjleft

=Pjdown

=Pjright

1.135 Function: =Pjdown

```
                flag=Pjdown(j)
```

Corresponds to the Jdown(j) function for normal joysticks. Pjdown tests, if the joystick in the parallel port is currently pressed down. If this is the case, True (-1) will be returned otherwise False (0).

Also see:

=Pjoy

=Pjup

=Pjleft

=Pjright

=Pfire

1.136 Function: =Pjleft

```
                flag=Pjleft(j)
```

Corresponds to the Jleft(j) function for normal joysticks. Pjleft tests, if the joystick in the parallel port is currently pressed to the left. If this is the case, True (-1) will be returned otherwise False (0).

Also see:

=Pjoy

=Pjup

=Pjdown

=Pjright

=Pfire

1.137 Function: =Pjoy

```
                bitmap=Pjoy(j)
```

Corresponds to the AMOS function Joy, with the difference, that one of the parallel port joysticks is checked instead of the normal joysticks.
'j' must be either 0 or 1.

```
The value 'bitmap' contains the following bits:

Bit 0=1: the joystick is currently moved up
Bit 1=1: the joystick is currently moved down
Bit 2=1: the joystick is pressed to the left
Bit 3=1: the joystick is pressed to the right
Bit 4=1: the fire button has been pressed

Also see:
```

=Pjup

=Pjdown

=Pjleft

=Pjright

=Pfire

1.138 Function: =Pjright

```
flag=Pjright(j)
```

Corresponds to the Jright(j) function for normal joysticks. Pjright tests, if the joystick in the parallel port is currently pressed to the right. If this is the case, True (-1) will be returned otherwise False (0).

Also see:

=Pjoy

=Pjup

=Pjdown

=Pjleft

=Pfire

1.139 Function: =Pjup

```
flag=Pjup(j)
```

Corresponds to the Jup(j) function for normal joysticks. Pjup tests, if the joystick in the parallel port is currently pressed up. If this is the case, True (-1) will be returned otherwise False (0).

Also see:

```
                 =Pjoy

                 =Pjdown

                 =Pjleft

                 =Pjright

                 =Pfire
```

1.140 Function: =Pt Cinstr

```
                num=Pt Cinstr(chan)
```

Pt Cinstr returns the instrument being played on music channel chan. num is therefore a value between 0 and 31, whereas 0 tells you, that no sample has been trigged at that very moment.

Also see:

```
                 =Pt Cpattern

                 =Pt Cpos

                 =Pt Cnote

                 Pt Play

                 Pt Stop

                 Pt Continue
```

1.141 Function: =Pt Cnote

```
                num=Pt Cnote(chan)
```

Pt Cnote returns the frequency of an instrument being played on music channel chan at that very moment.

Also see:

```
                 =Pt Cpattern

                 =Pt Cpos
                 =Pt Cinstr

                 Pt Play

                 Pt Stop

                 Pt Continue
```

1.142 Function: =Pt Cpattern

```
                songpos=Pt Cpattern

This function returns the current pattern/song position. This can be used
to restart a song at a given position when you stopped the music with
Pt Stop.

Also see:


                 =Pt Cpos

                 =Pt Cinstr

                 =Pt Cnote

                 Pt Play

                 Pt Stop

                 Pt Continue
```

1.143 Function: =Pt Cpos

```
                row=Pt Cpos

This function returns the current 'row' of the songs current pattern. The
returned value is a number between 0 and 63.

Also see:


                 =Pt Cpattern

                 =Pt Cinstr

                 =Pt Cnote

                 Pt Play

                 Pt Stop
                 Pt Continue
```

1.144 Function: =Pt Data Base

```
address=Pt Data Base
```

```
Returns the starting address of the internal datatable of the protracker
replay routine. Please don’t write into this area randomly.
```

1.145 Function: =Pt Instr Address

```
                address=Pt Instr Address(instnr)
```

```
Returns the starting address of the music instrument with the
number ’instnr’ of the current Protracker module.
```

```
Also see:
```

```
                 =Pt Instr Length

                 Pt Bank

                 Pt Instr Play
```

1.146 Function: =Pt Instr Length

```
                length=Pt Instr Length(instnr)
```

```
Returns the length of the instrument ’instnr’.
```

```
Also see:
```

```
                 =Pt Instr Address

                 Pt Bank

                 Pt Instr Play
```

1.147 Function: =Pt Signal

```
                signal=Pt Signal

The various music trackers Soundtracker, Noisetracker, Startrekker or
Protracker feature effect commands e.g to change the volume of a note,
or for portamento. Example for a Protracker noteline:
setzen oder für Portamento. Beispiel an einer Protracker Notenzeile:

00 C-1 1000 --- 0000 G-2 2000 --- 0000
   ^ ^ ^
   | | `-----------------------------------------------.
   | `----------------------.                          |
   `---------------.        |                          |
   Plays the note C in the 1st octave using instrument 1.

Additionally, you can use effects, e.g like in this line here:

00 C-1 1C20 --- 0F04 G-2 2E00 --- 0801
        ^^^      ^^^      ^^^      ^^^
Set vol.to $20  Speed  Filter Off  \|/
         (=32)  4 VBLs              Y
 .----------------------------------'
 |
Now this is a effect, which is unused in all trackers. It has been left
free to synchronize various events to the music. The value behind the '8'
is directly transferred to Pt Signal, when the music reaches the specific
point. (Warning! The value used in Protracker is entered as hex number!)

With this technique you can wait for a special accustic signal, or sync
some kind of musician bobs to the music.

If there is no signal waiting, zero is returned in the varible 'signal'.
After using Pt Signal, the old signal will be cleared. Pt Signal can only
keep one value at once, i.e if the last value hasn't been read out, the
next signal will overwrite the last one and is therefore lost.

If the end of the music is reached an $FF signal is sent automatically.

Also see:
```

Pt Cia Speed

Pt Play

Pt Stop

Pt Volume

Pt Voice

=Pt Vu

=Pt Cpattern

=Pt Cinstr

```
                 =Pt Cnote
```

1.148 Function: =Pt Vu

```
                vol=Pt Vu(channel)
```

Allows you to read the current volume of channel number 'channel'. If a new note is played, 'vol' contains the volume level else 0.

Also see:

```
                 Pt Cia Speed

                 Pt Play

                 Pt Stop

                 Pt Volume

                 Pt Voice

                 =Pt Signal

                 =Pt Cpattern

                 =Pt Cinstr

                 =Pt Cnote
```

1.149 Function: =Qarc

```
                angle=Qarc(deltax,deltay)
```

This function will return the angle to a point at the relative coordinates deltax,deltay. angle is in the same format used with Qsin and Qcos, so an angle of 360 degree is equalent to 1024. Moreover this function is *very* fast and rather accurate.

It's normally used for all kinds of 'aiming-at' routines, e.g to get the angle from one player to another to shoot a missle at ;-)

Also see:

```
                 =Qsin

                 =Qcos
```

1.150 Function: =Qcos

```
                value=Qcos(angle,radius)

This is a function to replace the original cosine function of AMOS. It is
faster and even allows you to multiple the value with the a parameter.
Moreover, no maths libraries are required. However, the angle of the sine
'angle' is not value in radians or degrees, but with Qcos, an angle of
360 degrees equals a value of 1024.

Also see:
```

=Qsin

=Qarc

1.151 Function: =Qrnd

```
value=Qrnd(maxrnd)

Qrnd is totally identical to the Rnd function, with the only difference,
that this one is much faster.
```

1.152 Function: =Qsin

```
                value=Qsin(angle,radius)

This is a function to replace the original sine function of AMOS. It is
faster and even allows you to multiple the value with the a parameter.
Moreover, no maths libraries are required. However, the angle of the sine
'angle' is not value in radians or degrees, but with Qsin, an angle of
360 degrees equals a value of 1024.

Also see:
```

=Qcos

=Qarc

1.153 Function: =Qsqr

```
root=Qsqr(value)

Calculates the square root from the value 'value'. However, it only works
with integer and is faster than the AMOS square root function =Sqr.
```

1.154 Function: =Red Val

```
                r=Red Val(rgb)

This function acquires the red value of a colour. Together with the other
Val functions, you can separate the colour into its three contents.

Also see:


                 =Green Val

                 =Blue Val

                 =Glue Colour
```

1.155 Function: =Replacestr$

```
                newstr$=Replacestr$(s$,search$ To replace$)

This function searches in the string s$ for occurences of search$ and
replaces them with the string replace$.

Also see:


                 =Insstr$

                 =Cutstr$
```

1.156 Function: =Rgb To Rrggbb

```
                rrggbb=Rgb To Rrggbb(rgb)

This function calculates a 12 bit colour value into a 24 bit AGA value.
The missing bits are set to zeros. Using this colour format you can
assign 256 different colour values each to the red, green or blue part,
that is 16777216 different values in total.

Also see:


                 Rrggbb To Rgb
```

1.157 Function: =Rrggbb To Rgb

```
                rgb=Rrggbb To Rgb(rrggbb)
```

The function Rrggbb To Rgb converts a 24 bit colour value into a 12 bits value, the other 12 bits will be discarded.

Also see:

```
                 Rgb To Rrggbb
```

1.158 Function: =Scanstr$

```
key$=Scanstr$(scancode)
```

This very handy function returns the name of a key according to the parameter 'scancode', which can be fetched using the AMOS Scancode function. If there is no key for the scancode, an empty string will be returned.

1.159 Function: =Scrn Bitmap

```
                bitmap=Scrn Bitmap
```

Fetches the address of the graphics/bitmap structure of the current AMOS screen.

Also see:

```
                 =Scrn Rastport

                 =Scrn Layer

                 =Scrn Layerinfo

                 =Scrn Region
```

1.160 Function: =Scrn Layer

```
                layer=Scrn Layer
```

Fetches the address of the graphics/layer structure of the current AMOS screen.

Also see:

=Scrn Rastport

=Scrn Bitmap

=Scrn Layerinfo

=Scrn Region

1.161 Function: =Scrn Layerinfo

```
layerinfo=Scrn Layerinfo
```

Fetches the address of the graphics/LayerInfo structure of the current AMOS screen.

Also see:

=Scrn Rastport

=Scrn Bitmap

=Scrn Layer

=Scrn Region

1.162 Function: =Scrn Rastport

```
rastport=Scrn Rastport
```

Fetches the address of the graphics/rastport structure of the current AMOS screen.

Also see:

=Scrn Bitmap

=Scrn Layer

=Scrn Layerinfo

=Scrn Region

1.163 Function: =Scrn Region

```
                 region=Scrn Region
```

Fetches the address of the graphics/region structure of the current AMOS screen.

Also see:

=Scrn Rastport

=Scrn Bitmap

=Scrn Layer

=Scrn Layerinfo

1.164 Function: =Sdeek

```
                 value=Sdeek(address)
```

The Sdeek function reads a
Word
from the EVEN memory address
'address', exactly the AMOS function Deek. However, Bit 15 is used as sign bit so the result will be a value between -32678 and 32677. You are adviced to use this function instead of Deek if you have doked a negative value into memory and then want to read it again.

Also see:

=Speek

1.165 Function: =Smouse Key

```
                 mk=Smouse Key
```

Returns the state of the mouse buttons in port 1.

```
Bit
 0 left mouse button
 1 right mouse button
 2 middle mouse button
```

Also see:

=X Smouse

=Y Smouse

Limit Smouse

Smouse Speed

Smouse X

Smouse Y

1.166 Function: =Speek

```
value=Speek(address)
```

The Speek function reads a
Byte
from the memory address ’address’,
exactly the AMOS function Peek. However, Bit 7 is used as sign
bit so the result will be a value between -128 and 127. You are
adviced to use this function instead of Peek if you have poked a negative
value into memory and then want to read it again.

Also see:

=Sdeek

1.167 Function: =Splinters Active

```
amount=Splinters Active
```

This function returns how many Splinters are active at the moment. This
can be used to know when the animation has finished.

Also see:

Splinters Back

Splinters Bank

Splinters Colour

Splinters Del

Splinters Do

Splinters Draw

Splinters Fuel

```
                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 Splinters Move
```

1.168 Function: =Tool Types$

```
                tools$=Tool Types$(filename$)
```

With this function you can acquire the Tool Type of an icon. The supplied file ’filename$’ must not have a~’.info’ appended! The various Tool Types are seperated by a line feed character (Chr$(10)). So they can be printed out easily using Print.

The name of your own program can be fetched using Command Name$.

Also see:

```
                 =Command Name$
```

1.169 Function: =Turbo Point

```
                c=Turbo Point(x,y)
```

This functions requests the colour of the pixel at the coordinates x,y. It is much faster as the normal Point instruction.

Also see:

```
                 Turbo Plot
```

1.170 Function: =Vclip

```
                newval=Vclip(val,lower To upper)
```

The Vclip function works like a Min(Max(val,lower),upper) expression, it restricts a value val into its lower and upper boundaries.

Also see:

=Vin

=Vmod

1.171 Function: =Vec Rot X

```
                newx=Vec Rot X(x,y,z)
newx=Vec Rot X
```

This function calculates the X coordinate, which results in the vector rotation of a three dimensional point x,y,z around all three axis. This coordinate is automatically projected from 3D to 2D by dividing it through the distance.

If you call the function with the parameters x,y,z all three new coordinates are calculated, i.e the y,z position too. If you require the Y coordinate too, when simply write the Vec Rot Y command WITHOUT the parameters. Same with Vec Rot Z.

This parameter system can be also used on Vec Rot Y in the same way.

Also see:

=Vec Rot Y

=Vec Rot Z

Vec Rot Pos

Vec Rot Angles

Vec Rot Precalc

1.172 Function: =Vec Rot Y

```
                newy=Vec Rot Y(x,y,z)
newy=Vec Rot Y
```

Calculates the new Y coordinate after the rotation. For more detailed information see Vec Rot X.

Also see:

=Vec Rot X

=Vec Rot Z

```
                 Vec Rot Pos

                 Vec Rot Angles

                 Vec Rot Precalc
```

1.173 Function: =Vec Rot Z

```
                newz=Vec Rot Z(x,y,z)
newz=Vec Rot Z

Calculates the new Z coordinate after the rotation. Please note that the
current camera position has already been added to this value. The Z
coordinate can be used to sort objects by distance or to evaluate a
brightness level.

More information on the syntaxes or the parameters see Vec Rot X.

Also see:


                 =Vec Rot X

                 =Vec Rot Y

                 Vec Rot Pos

                 Vec Rot Angles

                 Vec Rot Precalc
```

1.174 Function: =Vin

```
                flag=Vin(val,lower To upper)

The Vin function will create an exquisite bottle of wine... Hehe ;-)

No, actually, this function will check, if a value lies between lower
and upper. Is this the case, flag will return True (-1), otherwise
False (0).

Also see:


                 =Vclip

                 =Vmod
```

1.175 Function: =Vmod

```
                newval=Vmod(val,upper)
newval=Vmod(val,lower To upper)
```

Vmod restricts a value val into the lower and upper boundaries. However, it does it in another way compared to Vclip. If val exceeds upper by 1, it will be set to lower, if it exceeds upper by 2, it will be set to lower+1. If it goes deeper than lower by 1, it will be set to upper and so on.

So this function is not the same as

Add val,delta,lower To upper

If lower is omitted, zero is taken as lower boundary.

Example:

Print Vmod(100,50 To 150) would return 100 (should be clear).

Print Vmod(151,50 To 150) would return 50.

Print Vmod(152,50 To 150) would return 51.

Print Vmod(49,50 To 150) would return 150.

Print Vmod(0,50 To 150) would return 101.

Also see:

=Vclip

=Vin

1.176 Function: =Wordswap

```
                newval=Wordswap(value)
```

Swaps the upper and the lower Word of a value.

1.177 Function: =X Raster

```
                x=X Raster
```

This function returns the current X position of the raster beam in hardware coordinates. This value is not very accurate because the raster beam is very fast, sigh.

Also see:

```
                 Raster Wait

                 =Y Raster
```

1.178 Function: =X Smouse

```
                xpos=X Smouse
```

Returns the x coordinate of the second mouse.

Also see:

```
                 =Y Smouse

                 =Smouse Key

                 Limit Smouse

                 Smouse Speed

                 Smouse X

                 Smouse Y
```

1.179 Function: =Xfire

```
flag=Xfire(port,button)
```

This function returns the state of a possibly existant second fire button
on a joystick or joypad in the gameport numbered port. If the lowlevel-
library is available, all the other buttons can be checked aswell. This
library is distributed along with Kickstart 3.1 but even works with
2.0.

Possible values for the button parameter:

```
0: This is the normal fire button which exists on every joystick.
1: Second fire button on a few joysticks or the blue button on a gamepad.

2: Yellow fire button on a CD$^3$$^2$ controller.
3: Green fire button
4: Reverse button on the gamepad
5: Forward button
6: Play/Pause button
```

1.180 Function: =Y Raster

```
                y=Y Raster
```

Y Raster returns the current Y position of the raster beam. This function can be used to see how many rasterlines a specific part of a program takes.

Also see:

```
                 Raster Wait

                 =X Raster
```

1.181 Function: =Y Smouse

```
                ypos=Y Smouse
```

Returns the y coordinate of the second mouse.

Also see:

```
                 =X Smouse

                 =Smouse Key

                 Limit Smouse

                 Smouse Speed

                 Smouse X

                 Smouse Y
```

1.182 Command: Amcaf Aga Notation

```
                Amcaf Aga Notation On
Amcaf Aga Notation Off
```

Normally, AMCAF commands, which take colour values as parameter, use a normal 12 bit colour value in the format $RGB. Since machines like the

A4000/A1200
there are 24 bit colour values in the Format $RRGGBB.

After calling Amcaf Aga Notation On, all AMCAF commands and functions take 24 bit values, except
Rrrggbb To Rgb

and
Rgb To Rrggbb
.

The default setting is 12 bit.

1.183 Command: Audio Free

```
Audio Free
```

Using Audio Free you can free previously allocated sound channels again.

Also see:

Audio Lock

1.184 Command: Audio Lock

```
Audio Lock
```

When you start AMOS, the audio.device will be not informed, that AMOS wants to have the audio channels. Due to this flaw, other programs that are running in the background can replay a sound at any time, which will irritate the AMOS sound system. To avoid this and to by more system friendly, you can use the command Audio Lock to reserve the sound channels. As AMOS has a bug, you'll have to do it like that:

```
Extremove 1 : Rem removes music extension
Trap Audio Lock : Rem reserves the audio channels
ERR=Errtrap : Rem tests, if audio lock was successful
Extreinit 1 : Rem initialises the music extension again
Extdefault 1 : Rem calls the default routine of the extension
```

This Routine should be called as soon as possible and BEFORE you start to play music or sounds.

If the audiochannels are already in use by any other program, ERR holds an error code, otherwise zero.

To free the channels again, you have to call Audio Free.

Also see:

Audio Free

Extremove

Extreinit

Extdefault

1.185 Command: Bank Code Add

```
                Bank Code Add.b code,bank
Bank Code Add.b code,startaddress To endaddress
Bank Code Add.w code,bank
Bank Code Add.w code,startaddress To endaddress
```

Encodes the bank numbered 'bank' using the key code 'code'. To decode the bank, the same instruction has to be used with the negative key code.

Note: This encoding routine works by adding the value and is therefore very easy to decode.

Also see:

Bank Code Xor.y

Bank Code Mix.y

1.186 Command: Bank Code Mix

```
                Bank Code Mix.b code,bank
Bank Code Mix.b code,startaddress To endaddress
Bank Code Mix.w code,bank
Bank Code Mix.w code,startaddress To endaddress
```

The third possibility to encode a bank, is with Bank Code Mix. So coded banks should be hard to decode. To decode a bank you should use the same key code as seen with Bank Code Xor before.

Also see:

Bank Code Add.y

Bank Code Xor.y

1.187 Command: Bank Code Rol

```
                Bank Code Rol.b code,bank
Bank Code Rol.b code,startaddress To endaddress
Bank Code Rol.w code,bank
Bank Code Rol.w code,startaddress To endaddress
```

```
Using these commands every bit in each byte or word is rotated by ’code’
bits to the left. (Rol=RotateLeft). This results in a restriction of the
’code’ parameter from 1 to 7 on ’.b’ and 1 to 15 on ’.w’ command version.
To decode a bank either use the negative code with the same instruction or
the same key code along with the Bank Code Ror command.

Also see:


                 Bank Code Ror.y
```

1.188 Command: Bank Code Ror

```
                Bank Code Ror.b code,bank
Bank Code Ror.b code,startaddress To endaddress
Bank Code Ror.w code,bank
Bank Code Ror.w code,startaddress To endaddress

Similar to Bank Code Rol, but this time the bits are shifted to the right
instead of left.

Also see:


                 Bank Code Rol.y
```

1.189 Command: Bank Code Xor

```
                Bank Code Xor.b code,bank
Bank Code Xor.b code,startaddress To endaddress
Bank Code Xor.w code,bank
Bank Code Xor.w code,startaddress To endaddress

Bank Code Xor encrypts a bank in a similar way to Bank Code Add using an
other algorithm. Each byte or word of the bank is combined by an
’logical exclusive or’. To decode the bank simply use the same command
along with the same key code.
Note: A Xor encrytion is not so easy to crack without the right code.
Good codes are $AA and $55.

Also see:


                 Bank Code Add.y

                 Bank Code Mix.y
```

1.190 Command: Bank Copy

```
                Bank Copy sourcebank To targetbank
Bank Copy startaddress,endaddress To targetbank

Creates a identical copy of the bank with the number 'sourcebank' in the
bank numbered 'targetbank'. The second version does only copy a specific
part of the source bank into the target bank, creating a
                 temporary
                bank of same content.
```

1.191 Command: Bank Delta Decode

```
                Bank Delta Decode bank
Bank Delta Decode startaddress To endaddress

These commands decode a previously delta-encoded memory region. See
Bank Delta Encode for more details on delta encoding.

Also see:


                 Bank Delta Encode
```

1.192 Command: Bank Delta Encode

```
                Bank Delta Encode bank
Bank Delta Encode startaddress To endaddress

Bank Delta Encode encodes a memory bank with the so-called delta algo.
This is not packing, however, but yields better pack ratios on 8-bit sound
samples. Delta encoding just stores the difference from one byte to the
next, so it certain hull curve patterns in samples can be seen more
'clearly' for packing algorithms.

On Protracker Modules, you could use:

Pt Bank 3
Bank Delta Encode Pt Instr Address(1) To Start(3)+Length(3)

to encode the samples only, and

Pt Bank 3
Bank Delta Decode Pt Instr Address(1) To Start(3)+Length(3)

to reverse this effect again.

Also see:
```

```
                 Bank Delta Decode
```

1.193 Command: Bank Name

```
                Bank Name bank,name$
```

```
Renames the bank with the number 'bank' to the 8 characters long name
'name$'. Most AMOS commands ignore this ID, but e.g the AMOS Tracker
commands require a bank named 'Tracker '.

Also see:


                 =Bank Name$
```

1.194 Command: Bank Permanent

```
                Bank Permanent bank
```

```
If you defined a bank as 'Work', but afterwards want this bank to be

                 Permanently
                 in memory and in the program, you could use Bank Permanent.
This will make the bank stay resident in your program until it is erased.
This command also has use on MED-Modules, which were loaded with the
Med Load command and on Power and Imploder unpacked banks.

Also see:


                 Bank Temporary

                 Dload

                 Ppfromdisk

                 Imploder Load

                 Imploder Unpack
```

1.195 Command: Bank Stretch

```
                Bank Stretch bank To length
```

```
Extents the bank numbered 'bank' to the given new length 'length'. During
this process, the starting address of the bank is changed. This command
does not work on
```

Icon and sprite banks
.

1.196 Command: Bank Temporary

```
Bank Temporary bank
```

Changes a bank to
Temporary-type
, i.e it will be erased on the start of
your program or when calling the Default-Command.

Also see:

Bank Permanent

Wload

1.197 Command: Bank To Chip

```
Bank To Chip bank
```

Bank To Chip is the reverse command to Bank To Fast. It moves a bank into

Chip ram
. Obviously, the starting address will therefore change.

Also see:

Bank To Fast

1.198 Command: Bank To Fast

```
Bank To Fast bank
```

Moves a bank into
Fast ram
, if any is available. Naturally, the
bank will get a new starting address.
Bank To Fast won't work with
Icon and sprite bank
.
Warning: Do not try to replay musicis or sounds that resist in fast ram.

Also see:

Bank To Chip

1.199 Command: Bcircle

```
Bcircle x,y,r,plane
```

Draws an empty circle around x,y with radius 'r' into the bitplane 'plane'. This line is really only one pixel thick to ensure the circle can be filled by the
Blitter chip
.

Also see:

Blitter Fill

1.200 Command: Blitter Clear

```
Blitter Clear screen,bitplane
Blitter Clear screen,bitplane,x1,y1 To x2,y2
```

The
Blitter chip
can be used to clear
Bitplanes
as well.
In comparison to the AMOS command Cls, Blitter Clear allows you to wipe single bitplane instead of all.

Optionally you may include the coordinates of a rectangular region where the command should have effect on.

Also see:

Blitter Copy

Blitter Fill

Blitter Wait

=Blitter Busy

1.201 Command: Blitter Copy

```
                Blitter Copy sourcescreen,sourceplane To targetscreen,targetplane
Blitter Copy sourcescreen,sourceplane To targetscreen,targetplane,miniterm
Blitter Copy s1,p1,s2,p2 To targetscreen,targetplane
Blitter Copy s1,p1,s2,p2 To targetscreen,targetplane,miniterm
Blitter Copy s1,p1,s2,p2,s3,p3 To targetscreen,targetplane
Blitter Copy s1,p1,s2,p2,s3,p3 To targetscreen,targetplane,miniterm

With the help of Blitter Copy you can copy a
                 Bitplane
                 of a screen to
another or the same bitplane on another or the same screen. The Amiga

                 Blitter chip
                 is used to fulfill this action. If two or three
source screens and bitplanes parameters are given, the graphics can be
combined by so called miniterms and the result is then written into
the target screen.

*** IMPORTANT ***

1. Before you can call Blitter Copy, you MUST set the limits of the
   operation using Blitter Copy Limit.
2. Take care that all specified screens have these dimensions and all
   screens are big enough.

The optional parameter miniterm contains a bitmask which defines the way
the source planes should be combined.

If these values are omitted, these default values are used:

One source   : the bitplane is copies normally (=%11110000).
Two sources  : the bitplanes are combined using logical OR (=%11111100).
Three sources: the bitplanes are combined using logical OR (=%11111110).

Frequently used miniterm values:

Sourceplanes one, two and three represent A, B and C.

1: %11110000: copy bitplane                : A
1: %00001111: invert bitplane              : NOT A
2: %11111100: use logical OR on the planes : A OR B
2: %00000011: OR planes and invert them    : NOT (A OR B)
2: %11000000: the two planes are ANDed     : A AND B
2: %00111111: planes are ANDed and inverted : NOT (A AND B)
2: %00111100: the planes will be XORed     : A XOR B
2: %11000011: planes are XORed and inverted : NOT (A XOR B)
3: %11111110: OR all planes (create mask)  : A OR B OR C
3: %00000001: OR all planes and invert them : NOT (A OR B OR C)
3: %10000000: Do a logical AND on all planes: A AND B AND C
3: %01111111: AND all planes and invert them: NOT (A AND B AND C)

If you want to create other combinations, simply try out the values you
like (nothing can happen except of bad looking screens). If you have
some knowledge on boolean algebra, read
                 Blitter miniterms
                .
```

Also see:

Blitter Copy Limit

Blitter Fill

Blitter Clear

Blitter Wait

=Blitter Busy

1.202 Command: Blitter Copy Limit

```
                Blitter Copy Limit screen
Blitter Copy Limit x1,y1 To x2,y2
```

Using this command you define the rectangluar area which will be used for the Blitter Copy command. If you only specify the ’screen’ parameter, the full screen dimensions of the screen numbered ’screen’ will be taken. Otherwise x1,y1 represents the upper left corner and x2,y2 the lower right corner of the region which will be affected by the copying and combining process. This area will be used for ALL screnns, therefore you must ensure that every screen is at least as big as the lower right corner of the specified limits.

Also see:

Blitter Copy

1.203 Command: Blitter Fill

```
                Blitter Fill screen,bitplane
Blitter Fill screen,bitplane,x1,y1,x2,y2
Blitter Fill sourcescreen,sourceplane To targetscreen,targetplane
Blitter Fill ss,sp,x1,y1,x2,y2 To ts,tp
```

With Blitter Fill you can fill polygons. However, there are some limitations: the filling algorithm of the
Blitter chip
is very simple.
It does only fill the gap between two dots of a horizontal line. Therefore the limiting lines may only be one pixel think. These lines can be either created using Turbo Draw or Bcircle.

The ’screen’ and ’bitplane’ parameter specify the screen which contains the area to be filled. Additionally you can give a rectangular region which is defined by the coordinate pairs x1,y1 and x2,y2.

Moverover you can give a second screen 'targetscreen' and a second bitplane number 'targetplane' into which the filled figures are written. In this mode, the source screen ist not altered in any way.

If more than one filled figure is to be drawn and these don't overlap, you can draw all limiting lines first and then fill all figures in one cycle.

The Blitter processes the screen from the lower right to the upper left. Due to this feature you should draw on Double Buffered or hidden screens to avoid flickering.

Also see:

Turbo Draw

Bcircle

Blitter Copy

Blitter Clear

Blitter Wait

=Blitter Busy

1.204 Command: Blitter Wait

```
Blitter Wait
```

This command waits until the
Blitter
has finished his work. You are
adviced to use this command before calling e.g Print or any other command that does not require the blitter, that will draw onto a screen, which is currently altered by the blitter. You should use Blitter Wait in front of Wait Vbl, too.

Also see:

=Blitter Busy

Blitter Copy

Blitter Fill

Blitter Clear

1.205 Command: Bzoom

```
Bzoom s1,x1,y1,x2,y2 To s2,x3,y3,factor

Using this command you can zoom a region on the screen to a integer
multiple very fast. As result the graphics are double, four times or eight
times as wide and from 1 to 15 times as high as before. The command zooms
the rectangular area from x1,y1 to x2,y2 on screen s1 onto the screen s2
at the coordinates x3,y3. The ending coordinates result from the zooming
factor.

The coordinates x1 and x2 are rounded down to the next multiple of eight,
x3 is even rounded to the nearest multiple of 16. Moreover, you must
ensure that the area to be zoomed fits on the target screen, because no
clipping is done at all.

The factor parameter defines how much the original image should be zoomed.
It is divided into two nibbles: $yx. y may be a value from 1 to F, x must
be either 1,2,4 or 8.

Example factors:
$11 copies the image without zooming.
$22 zooms the picture to double width and height.
$34 zooms the graphics to 4 times width and 3 times height.
```

1.206 Command: Change Bank Font

```
                Change Bank Font bank

This is a really sensible command which sets the text font on the current
screen to the one saved in the memory bank numbered 'bank'.

If you change the font using this command, no diskfont.library is required
any more. This command is also quite important if you write your programs
to be installed on harddisk that use your own fonts.

Bank Fonts can be created easily with the Make Bank Font command.

Also see:


                 Make Bank Font

                 Change Font
```

1.207 Command: Change Font

```
                Change Font "fontname.font"
Change Font "fontname.font",height
Change Font "fontname.font",height,style
```

Opens a font directly from disk. This font will be taken for future text print outs. 'height' represents the vertical size of the font, a default height of 8 will be filled in if this parameter is omitted. The 'style' parameter is only required in special cases and sets the type of the requested charset (see Set Text).

Since OS 2.0 it's even possible to load multicoloured fonts. The poor guys on Kickstart 1.3 can use the program ColorText (which e.g is included on the DPaint disk) to patch the graphics.library to handle these fonts as well. The diskfont.library (length=51200 bytes) found on the Fonts disk can scale your font to your desired size.

If you open many fonts, you should call the command
Flush Libs
from
time to time to remove unused fonts from memory.

With the Make Bank Font command you can convert this font into a memory bank, so you don't require the diskfont.library and the font file anymore.

Additionally, you can change the font which is used for the Print instruction using Change Print Font.

Also see:

=Font Style

Make Bank Font

Change Bank Font

Change Print Font

1.208 Command: Change Print Font

```
Change Print Font bank
```

Change Print Font changes the charset which is used for the Print command. This font always 8x8 pixels big and contains 256 characters, which is stored in a memory bank of exactly 2 KB (=2048 bytes). Therefore the 'bank' parameter has to point exactly to such a bank. These Print-Font banks can be created by hand (reserve the bank, write 8 bytes per chars) or with the help of the Font-Editor supplied on the Accessories disk (AMOSPro_Accessories:Font8x8_Editor.AMOS) and then can be loaded using Dload.

1.209 Command: Convert Grey

```
Convert Grey sourcescreen To targetscreen
```

```
Using this command you can convert any screen into a grey scale image,
even
                 HAM
                 and ExtraHalfBright screens are supported.
The only thing you have to do, is to open the screen and set the correct
palette. The parameter 'sourcescreen' defines the screen to be grey
scaled, 'targetscreen' is the number of the screen where the grey image
is to be created on. The number of colours in the target screen will be
taken in account, but it makes no sense to open a HAM screen for that
purpose.
```

1.210 Command: Coords Bank

```
                Coords Bank bank,coords
Coords Bank bank

Reserves the bank numbered 'bank' to store a coordinates array. These
banks are utilitised e.g for Splinters. 'coords' holds the amount of
coordinates the bank should be able to hold. Each coordinate requires 4
bytes. If this parameter is omitted the existing bank will only be
switched to without erasing it. So you can jump between predefined banks.

Also see:


                 Coords Read

                 =Count Pixel
```

1.211 Command: Coords Read

```
                Coords Read screen,colour,x1,y1 To x2,y2,bank,mode

This command is required to read the coordinates for Splinters into a
bank. This memory bank with the number 'bank' must have been defined
previously with a call to the Coords Bank command. The 'screen' parameter
holds the number of the screen on which the pixels are. 'colour'
represents the background colour, that will be left out when reading in
the dots. The rectangle which is described by the coordinates x1,y1 and
x2,y2 will be then read in and all dots, which don't have the colour
'colour', will be stored in the bank. 'mode' can be either 0, if you want
the coords to be read in strict order, or 1, if you want the coords
to be shuffled.

Also see:


                 Coords Bank

                 =Count Pixel
```

1.212 Command: Dload

```
                Dload file$,bank
```

Just like Wload, this command loads a file into memory, but this time a
Permanent
'Datas'-Bank will be created.

Also see:

Wload

Bank Permanent

1.213 Command: Dsave

```
                Dsave file$,bank
```

Dsave is exactly the same as Wsave in every aspect.

Also see:

Wsave

1.214 Command: Examine Dir

```
                Examine Dir directory$
```

This command loads all information about the drawer 'directory$' into the FileInfoBlock. Additionally, the contents of the directory can be read out by Examine Next$.

Also see:

=Examine Next$

Examine Stop

=Object Name$

=Object Type

```
=Object Size

=Object Blocks

=Object Protection

=Object Time

=Object Date

=Object Comment$
```

1.215 Command: Examine Object

```
Examine Object file$
```

Examine Object supplies you with all available information about the

Object
named 'file$'. These datas can then be requested using the Object functions without any parameters.

Also see:

```
=Object Name$

=Object Type

=Object Size

=Object Blocks

=Object Protection

=Object Time

=Object Date

=Object Comment$
```

1.216 Command: Examine Stop

```
Examine Stop
```

Aborts the reading process of a directory. After this command, you may not make any further calls to Examine Next$.

Also see:

```
                 Examine Dir

                 =Examine Next$
```

1.217 Command: Exchange Bob

```
                Exchange Bob i1,i2
```

This command simply swaps the two images i1 and i2 in the current sprite bank. i1 and i2 must exist as a valid image, otherwise an error will be reported.

Also see:

```
                 Exchange Icon
```

1.218 Command: Exchange Icon

```
                Exchange Icon i1,i2
```

This command simply swaps the two images i1 and i2 in the current icon bank. i1 and i2 must exist as a valid image, otherwise an error will be reported.

Also see:

```
                 Exchange Bob
```

1.219 Command: Extdefault

```
                Extdefault extnb
```

This command has been written especially for advanced users. With this one you can call the routine in the AMOS extension in slot ’extnb’ which is normally executed after the start of a program or by the Default command. Often this routine resets the interior database to the default values.

Extdefault should be called after relinking a extension with Extreinit.

If no extension is loaded in slot number ’extnb’, this instruction has no effect.

Also see:

Extreinit

Extremove

Audio Lock

1.220 Command: Extreinit

```
Extreinit extnb
```

This command causes a restart of the extension numbered 'extnb'. Extreinit may only be called on a extension that has been removed using Extremove. Otherwise, you can lose memory or even crash your computer.

Also see:

Extremove

Extdefault

Audio Lock

1.221 Command: Extremove

```
Extremove extnb
```

The Extremove command removes the extension in the slot 'extnb' from memory like when exiting AMOS. After this command no further instructions requiring this extension may be called. As AMOS tries to unlink the extension itself on quitting, this could cause some trouble. So don't remove an extension too long, revoke it with Extreinit before exiting.

Also see:

Extreinit

Extdefault

Audio Lock

1.222 Command: Fcircle

```
                Fcircle x,y,r
```

Fcircle is a command which has been missed in AMOS for a long time. It draws a filled circle with the radius 'r' and center~x,y.

Also see:

```
                 Fellipse
```

1.223 Command: Fellipse

```
                Fellipse x,y,rx,ry
```

This command draws an ellipse, similar to the AMOS Ellipse command. However, Fellipse draws a filled ellipse.

Also see:

```
                 Fcircle
```

1.224 Command: File Copy

```
File Copy sourcefile$ To targetfile$
```

Copies the file with the name 'sourcefile$' to the file 'targetfile$'. This command allows you to even copy a file of 3 MB in size, even if you only got 100 KB of free memory.

1.225 Command: Flush Libs

```
Flush Libs
```

This command closes all libraries, fonts and devices, which are currently not in use and tries to get as much memory as possible. During this process the PowerPacker and the diskfont.library will be 'flushed', too.

1.226 Command: Ham Fade Out

```
                Ham Fade Out screen
```

As
                 HAM screens
                 cannot be faded out with the normal Fade commands, AMCAF

provides you with a world sensation: The HAM fader! Although you can only fade out pictures, it’s more than nothing, isn’t it?

Ham Fade Out darkens the screen ’screen’ by one single step. After calling it 16 times, the Ham screen is completely black.

Technically, it’s not possible to fade in a ham screen without enormous processor power, but for fading out, a modified Shade Bobs routine is used.

1.227 Command: Imploader Load

```
                Imploder Load file$,bank
```

This instruction tries to load and unpack a file-imploded file. The result will be stored in the bank numbered~’bank’. If the specified file is not packed using file imploder, it will be loaded into memory normally. If the value of ’bank’ is a negative number, the file will be unpacked into

Chip ram
.

Imploder Load does not need more memory to unpack the packed bank. Additionally, Imploder Load is extremly fast.

The File Imploder is part of the Turbo Imploder 4.0 package.
Turbo Imploder 4.0 is a very fast packer for executables, written by Albert-Jan Brouwer and Peter Struijk.

The File Imploder can be found on the installation disk in the C: drawer (AMCAF_Install:C/Fimp) along with the manual AMCAF_Install:C/Fimp.man).

Also see:

Imploder Unpack

1.228 Command: Imploder Unpack

```
                Imploder Unpack sbank to tbank
```

Unpacks the File Imploder packed file, which is stored in the memory bank ’sbank’, into the target bank ’tbank’. If the source bank does not contain a File Imploded file (which can be determinated be the first longword ’IMP!’) a error message will be returned. A negative ’tbank’ value will force the bank to be unpacked into
Chip ram
.

Also see:

Imploder Load

1.229 Command: Launch

```
Launch file$
Launch file$,stack
```

The Launch instruction starts a new task which is saved on disk with the name 'file$'. The optional parameter 'stack' holds the size of the stack memory to be allocated (default value is 4096).
Note: Many programs don't work when started from AMOS, as they are not part of a workbench or CLI environment.

1.230 Command: Limit Smouse

```
                Limit Smouse
Limit Smouse x1,y1 To x2,y2
```

Defines the region in which the mouse can be moved on the screen. If the parameters are omitted, the full size of the current screen will be used as default.

Also see:

=X Smouse

=Y Smouse

=Smouse Key

Smouse Speed

Smouse X

Smouse Y

1.231 Command: Make Bank Font

```
Make Bank Font bank
```

Using this powerful command you can store any amiga font in a memory bank. The current font on the current screen will be taken to create the font bank numbered 'bank'. To change the current text font simply use the Change Font instruction.

These banks don’t require the ’diskfont.library’ or other disk access any more, once they have been created. To load a font from a bank use the Change Bank Font command.

Also see:

Change Bank Font

Change Font

1.232 Command: Make Pix Mask

```
Make Pix Mask screen,x1,y1 To x2,y2,bank
```

Grabs a specific part of the screen and saves it into the bank with the number bank. This command can be used to create a mask for the Pix Shift instruction. If you use such a mask, the size must be exactly the same like the limits you specify with the Pix Shift commands.

Also see:

Pix Shift Up

Pix Shift Down

Pix Brighten

Pix Darken

1.233 Command: Mask Copy

```
Mask Copy screen1,x1,y1,x2,y2 To screen2,x3,y3,maskaddress
```

Copies a part of a screen to an other, just like Screen Copy. However, a mask bitplane can be given. ’maskaddress’ represents the startaddress of the
Bitplane
.

1.234 Command: Nop

```
Nop
```

This ’command’ has no effect et al. It’s only use is for speed testing routines.

Also see:

    =Nfn

1.235 Command: Open Workbench

Open Workbench

Tries to open the workbench again, if it has been closed previously. The Workbench screen can be closed using the AMOS command Close Workbench to get more memory.

1.236 Command: Pal Get Screen

```
Pal Get Screen palnr,screen
```

This functions stores the complete palette of screen number ’screen’ in the interior palette memory number ’palnr’. ’palnr’ must be a value between 0 and 7.

This command is used to quickly store a specific palette of a screen in a buffer. To restore the old palette, you can call the Pal Set Screen command.

Also see:

    Pal Set Screen

    Pal Set

    =Pal Get

1.237 Command: Pal Set

```
Pal Set palnr,index,colour
```

Changes the colour entry numbered ’index’ in the palette with the number ’palnr’ to the colour value ’colour’. palnr must be range from 0 to 7.

Also see:

    Pal Set Screen

    Pal Get Screen

```
                 =Pal Get
```

1.238 Command: Pal Set Screen

```
                Pal Set Screen palnr,screen
```

```
Writes back the previously stored colour palette 'palnr' to the screen
number 'screen'.

Also see:
```

```
                 Pal Get Screen

                 Pal Set

                 =Pal Get
```

1.239 Command: Pal Spread

```
Pal Spread c1,rgb1 To c2,rgb2

Creates a smooth blend between the two colours rgb1 and rgb2. The
resulting colour set will be stored between c1 and c2.
```

1.240 Command: Paste Ptile

```
                Paste Ptile x,y,t
```

```
Places a Ptile block at the postion x,y. These coordinates must be given
as block positions, that means that position 1,4 corresponds to the
screen coordinates 16,64.

Also see:
```

```
                 Ptile Bank
```

1.241 Command: Pix Brighten

```
                Pix Brighten screen,c1,c2,x1,y1 To x2,y2
Pix Brighten screen,c1,c2,x1,y1 To x2,y2,bank
```

```
Using this command, you can increase the colour indexes in the rectangular
area from x1,y1 to x2y2. The 'screen' parameter therefore contains the
number of the screen on which this regions is situated. 'c1' and 'c2' hold
the border colours, which should be taken into account for the colour
cycling, other colours are not affected.

If you supply the optional parameter 'bank', you can use a previously
calculated mask which can be created with the Make Pix Mask command.

In difference to Pix Shift Up, the colour indexes don't override 'c2'.

Also see:


                 Pix Darken

                 Pix Shift Up

                 Pix Shift Down

                 Make Pix Mask
```

1.242 Command: Pix Darken

```
                Pix Darken screen,c1,c2,x1,y1 To x2,y2
Pix Darken screen,c1,c2,x1,y1 To x2,y2,bank

Using this command, you can decrease the colour indexes in the rectangular
area from x1,y1 to x2y2. The 'screen' parameter therefore contains the
number of the screen on which this regions is situated. 'c1' and 'c2' hold
the border colours, which should be taken into account for the colour
cycling, other colours are not affected.

If you supply the optional parameter 'bank', you can use a previously
calculated mask which can be created with the Make Pix Mask command.

In difference to Pix Shift Down, the colour indexes don't go below 'c1'.

Also see:


                 Pix Brighten

                 Pix Shift Up

                 Pix Shift Down

                 Make Pix Mask
```

1.243 Command: Pix Shift Down

```
                Pix Shift Down screen,c1,c2,x1,y1 To x2,y2
Pix Shift Down screen,c1,c2,x1,y1 To x2,y2,bank
```

Using this command, you can decrease the colour indexes in the rectangular area from x1,y1 to x2y2. The 'screen' parameter therefore contains the number of the screen on which this regions is situated. 'c1' and 'c2' hold the border colours, which should be taken into account for the colour cycling, other colours are not affected.

If you supply the optional parameter 'bank', you can use a previously calculated mask which can be created with the Make Pix Mask command.

However, if the colour indexes go below 'c1', they are set back to 'c2' again.

Also see:

Pix Shift Up

Pix Brighten

Pix Darken

Make Pix Mask

1.244 Command: Pix Shift Up

```
                Pix Shift Up screen,c1,c2,x1,y1 To x2,y2
Pix Shift Up screen,c1,c2,x1,y1 To x2,y2,bank
```

Using this command, you can increase the colour indexes in the rectangular area from x1,y1 to x2y2. The 'screen' parameter therefore contains the number of the screen on which this regions is situated. 'c1' and 'c2' hold the border colours, which should be taken into account for the colour cycling, other colours are not affected.

If you supply the optional parameter 'bank', you can use a previously calculated mask which can be created with the Make Pix Mask command.

However, if the colour indexes override 'c2', they are set back to 'c1' again.

Also see:

Pix Shift Down

Pix Brighten

Pix Darken

Make Pix Mask

1.245 Command: Ppfromdisk

```
                Ppfromdisk file$,bank
```

Loads the file named 'file$' into memory bank number 'bank' and decruches it, if it has been packed using PowerPacker. Is 'bank' a negative value, the file is unpacked into

Chip ram

instead. If the file is Imploder packed, the Imploder Load command will be called, and if it is not packed, it will be loaded normally using Wload.

Also see:

Pptodisk

Ppunpack

1.246 Command: Pptodisk

```
                Pptodisk file$,bank
Pptodisk file$,bank,efficiency
```

The Pptodisk command crunches and saves the bank numbered 'bank' into the file 'file$' using the PowerPacker algorithm. The optional 'efficiency' parameter determinates how good the bank is to be packed and must range between 0 (very fast, but less efficient) and 4 (best, but slow).
The powerpacker.library is requires for this and the two more commands.

Sorry for the name 'Pptodisk' but 'Ppsave' has already been used by AMOS.

Also see:

Ppfromdisk

Ppunpack

1.247 Command: Ppunpack

```
                Ppunpack sourcebank To targetbank
```

Decrunches a powerpacked file in 'sourcebank' into the bank number 'targetbank'. Is 'targetbank' a negative value, so it will be decrunched into

Chip ram
.
BTW: Powerpacked files can be identified by the first
Longword
containing ’PP20’.

Also see:

Pptodisk

Ppfromdisk

1.248 Command: Protect Object

```
Protect Object pathfile$,prot
```

Changes the
Protection Flags
of the
Object
’pathfile$’ to the
bitmapped value ’prot’.

Also see:

=Object Protection

1.249 Command: Pt Bank

```
Pt Bank bank
```

This command is used if you want to play back instruments from a music
module but the music bank has not yet been specified with Pt Play.

Also see:

Pt Play

Pt Continue

Pt Instr Play

=Pt Instr Address

=Pt Instr Length

1.250 Command: Pt Cia Speed

```
                Pt Cia Speed bpm
```

With the help of this command, you can set the speed for replaying the Protracker modules. However, the music may change the speed on it's own if it has been composed for CIA-Timing. The parameter 'bpm' set the number of beats per minute or if you specifiy a value of zero, the timing will be switched from CIA-Timing to Vertical Blank Timing. Then the bpm rate is automatically set to exacty 125 and all CIA-Timing from the module are ignored if such appear in the music.

CIA-Timing causes some problems in conjunction with AMOS sprites, which show up as a flickering at regular intervals. If you work with sprites you should switch back to VBL-Timing where possible using Pt Cia Speed 0.

If a music module is not replayed correctly using VBL-Timing, i.e is running to fast or slow, you must switch to Cia-Timing. This can achieved with a Pt Cia Speed 125 command directly in front of a Pt Play instruction.

The default is CIA-Timing with 125 bpm.

Also see:

```
                 Pt Play

                 Pt Stop

                 Pt Continue

                 Pt Voice

                 =Pt Cpattern

                 =Pt Cpos
```

1.251 Command: Pt Continue

```
                Pt Continue
```

Restarts the protracker music at that point it has previously been stopped using Pt Stop. Do not attempt use Pt Continue without having started the music once with Pt Play.

Also see:

```
                 Pt Play

                 Pt Stop
```

Pt Cia Speed

Pt Bank

=Pt Cpattern

=Pt Cpos

1.252 Command: Pt Instr Play

```
                Pt Instr Play instnr
Pt Instr Play voice,instnr
Pt Instr Play voice,instnr,freq
```

The Pt Instr Play instruction replays an instrument of the current Protracker music module.

The 'voice' parameter contains a bitmask that says on which channels the instrument number 'instnr' should be played. If it is ommitted, the sound effect is output on all four channels. 'instnr' must range between 1 and 31. The optional argument 'freq' holds the frequency which should be used for the sample.

If you want to replay a sample repeatingly, just specify a negative value for 'instnr'. The volume of the instrument can be set with the Pt Sam Volume command.

Before you can replay an instrument of a module, you first have to specify the bank that holds the music. This can be achieved either by a call to Pt Bank or Pt Play.

Also see:

Pt Bank

Pt Play

Pt Sam Stop

Pt Sam Volume

Pt Sam Play

Pt Raw Play

=Pt Instr Address

=Pt Instr Length

1.253 Command: Pt Play

```
                Pt Play bank
Pt Play bank,songpos
```

Pt Play stars a Protracker music module which has be situated in memory bank number 'bank'. Optionally, you can specify a song position, which the music should be played from.

Before you start the music with Pt Play you should choose either Vertical Blank-Timing or CIA-Timing for the module using Pt Cia Speed.

With help of the Pt Instr Play instruction you can replay any instrument of the music track. This method has the advantage that you can reuse a sample that occures in the music for your sound effects.

If you want to use the instruments only, without replaying the music, you can call the Pt Bank command instead.

The music will be turned off again with a call to the Pt Stop instruction.

Also see:

Pt Cia Speed

Pt Stop

Pt Continue

Pt Volume

Pt Voice

Pt Sam Play

Pt Instr Play

Pt Bank

=Pt Instr Address

=Pt Instr Length

=Pt Vu

=Pt Signal

=Pt Cpattern

=Pt Cpos

=Pt Cinstr

=Pt Cnote

1.254 Command: Pt Raw Play

```
                Pt Raw Play voice,address,length,freq
```

This command corresponds to the normal AMOS command Sam Raw. Pt Raw Play replays a piece of memory as audio sample. The 'voice' parameter sets the channels to use, 'address' contains the starting address of the memory block, 'length' the length of the sample in bytes and 'freq' holds the replaying speed in Hertz.

Also see:

Pt Sam Play

Pt Instr Play

Pt Sam Stop

Pt Sam Volume

Pt Sam Freq

1.255 Command: Pt Sam Bank

```
                Pt Sam Bank bank
```

If you want to replay standard AMOS samples with AMCAF, you have to inform AMCAF about the Sample-Bank, which contains the sound effects. And that's exactly what Pt Sam Bank does. 'bank' holds the bank number of the AMOS sample bank.

Please remember, that you can replay the instruments of music modules directly.

Also see:

Pt Sam Play

Pt Sam Stop

Pt Sam Volume

Pt Sam Freq

Pt Instr Play

Pt Raw Play

1.256 Command: Pt Sam Freq

```
                Pt Sam Freq chan,freq
```

This command changes the frequency of the sample currently being played on channel chan to the new frequency in freq. If no sample is currently played on this channel, it will have no effect nor will it have effect on the protracker music.

Also see:

Pt Sam Play

Pt Sam Stop

Pt Sam Volume

Pt Instr Play

Pt Raw Play

1.257 Command: Pt Sam Play

```
                Pt Sam Play samnr
Pt Sam Play voice,samnr
Pt Sam Play voice,samnr,freq
```

The Pt Sam Play command plays a sample out of an AMOS sample bank. The advantage to the normal Sam Play instruction is that the sounds ’interact’ with the Protracker music. Moreover, no notes are missed out, like AMOS does this on accelerated Amigas.

The ’voice’ parameter contains a bitmask, that describes, on which channels the sample number ’samnr’ should be replayed. If it is ommitted, the sound effect will be played on all four sound channels. The optional argument ’freq’ holds the replaying speed, that shall be used.

If you want the sample to be looping, just give a negative value for ’samnr’. The volume of the sample can be changed using Pt Sam Volume.

Also see:

Pt Sam Bank

Pt Sam Stop

Pt Sam Volume

Pt Sam Freq

Pt Instr Play

```
                 Pt Raw Play
```

1.258 Command: Pt Sam Stop

```
                Pt Sam Stop voice
```

```
Stops a sound effect on the channels, which are defined in the bitmap
'voice'. This is valid for samples started by Pt Sam Play, Pt Instr Play
and Pt Raw Play.

Also see:
```

```
                 Pt Sam Play

                 Pt Instr Play

                 Pt Raw Play

                 Pt Sam Bank

                 Pt Sam Stop

                 Pt Sam Volume

                 Pt Sam Freq
```

1.259 Command: Pt Sam Volume

```
                Pt Sam Volume volume
Pt Sam Volume voice,volume
```

```
This instruction sets the volume for the following sample replays. The
'volume' paramter must range from 0 to 64. If you specify the optional
parameter 'voice', the command only has effect on the currently played
sample, but not on the following samples.

Also see:
```

```
                 Pt Sam Play

                 Pt Sam Bank

                 Pt Sam Stop

                 Pt Sam Freq

                 Pt Instr Play
                 Pt Raw Play
```

1.260 Command: Pt Stop

```
                Pt Stop
```

```
Stops the current music module. You can restart the music with the
Pt Continue command from the very position you stopped it.

Also see:


                 Pt Cia Speed

                 Pt Play

                 Pt Continue

                 Pt Volume

                 Pt Voice

                 =Pt Vu

                 =Pt Signal

                 =Pt Cpattern

                 =Pt Cpos
```

1.261 Command: Pt Voice

```
                Pt Voice bitmask
```

```
This command defines the channels which should be used for the music and
which should be free for e.g sound effects.

The 'bitmask' parameter contains -similar to the AMOS Voice command- four
bits, that are assigned to one channel each. If a bit is set, this channel
can be heard. Pt Voice %1111 turns on all sound channels.

Also see:


                 Pt Cia Speed

                 Pt Play

                 Pt Stop
```

Pt Continue

Pt Volume

=Pt Vu

=Pt Signal

1.262 Command: Pt Volume

```
Pt Volume value
```

The Pt Volume instruction sets the volume of the music. The parameter 'value' may range from 0 (silent) to 64 (full volume).

Also see:

Pt Cia Speed

Pt Play

Pt Stop

Pt Continue

Pt Voice

=Pt Vu

=Pt Signal

1.263 Command: Ptile Bank

```
Ptile Bank bank
```

Sets the bank to use for Ptiles. Actually, you should not read this command description. The Ptile commands seem to be only of very low use and are rather uninteresting for you.

Also see:

Paste Ptile

1.264 Command: Rain Fade

```
Rain Fade rainbownr,$RGB
Rain Fade rainbownr To targetrainbow

Using this powerful instruction you can fade the rainbow numbered
'rainbownr' out or to other colours. With the first version of Rain Fade
you may specify the target colour '$RGB' to which all colours of the
rainbow will be faded. The other versions fades exactly to the colours
of an other rainbow with the number 'targetrainbow'.

Rain Fade works step by step only. Therefore you need a maximum of 16
calls to reach the new colour values.
```

1.265 Command: Raster Wait

```
                Raster Wait y
Raster Wait x,y

This function can be used to wait for a specific position of the raster
beam. The x and y parameters contain the hardware coordinates from which
the program should continue.

Also see:


                 =X Raster

                 =Y Raster
```

1.266 Command: Reset Computer

```
Reset Computer

This very dangerous instruction reboots your Amiga. Obviously, this
command never returns to the program.
```

1.267 Command: Set Ntsc

```
                Set Ntsc

This command switches to 60Hz NTSC screen mode.

Also see:


                 Set Pal
```

1.268 Command: Set Object Comment

```
                Set Object Comment pathfile$,comment$
```

This command sets the comment of the Object 'pathfile$' to the string 'comment$'.

Also see:

```
                 =Object Comment$
```

1.269 Command: Set Object Date

```
                Set Object Date pathfile$,date,time
```

This command changes the date of the Object 'pathfile$' to the given date and time stamp.

This command only works on OS2.0 and higher.

Also see:

```
                 =Object Date

                 =Object Time
```

1.270 Command: Set Pal

```
                Set Pal
```

Set Pal returns to the normal 50Hz PAL mode.

Also see:

```
                 Set Ntsc
```

1.271 Command: Set Rain Colour

```
                Set Rain Colour rainbownr,newcolour
```

With the help of this command you can change the colour index of a rainbow which has been set previously by Set Rainbow to a new value. This means that you can remove the irretating limit to the first 16 colours and are now able to access all 32 colours. But there are even more possibilities! A colour index of -63 enables you to alter the hardware scrolling register, so you can create fancy water and wobbel effects.

For more effects you need to install the SetRainPatch. Also see
Notes
.

Just look at the example programs for more explainations.

1.272 Command: Set Sprite Priority

```
Set Sprite Priority bitmap
```

Sets the prioritity of the sprites in conjunctions with the playfields to the value 'bitmap'. 'bitmap' is a bit mask in the following format:

```
Bits 0-2: 0=all sprites will be displayed behind playfield 1
          1=the sprites 0-1 appear in front of playfield 1
          2=the sprites 0-3 are drawn in front of the first playfield
          3=the sprites 0-5 appear in front of playfield 1
          4=all sprites will be displayed in front of playfield 1
Bits 3-5: 0=all sprites will be displayed behind playfield 2
          1=the sprites 0-1 appear in front of playfield 2
          2=the sprites 0-3 are drawn in front of the second playfield
          3=the sprites 0-5 appear in front of playfield 2
          4=all sprites will be displayed in front of playfield 2
```

This is a little bit confusing, I admit, but just try it out.

1.273 Command: Shade Bob Down

```
                Shade Bob Down screen,x,y,image
```

This instruction draws a Shade Bob on the screen numbered 'screen' at the coordinates x,y. This bob only decreases the colour indexes only. 'image' holds the image number of the sprite bank which should be used to draw the bob. Either the mask or the first bitplane of the object is used for this process, according to the setting of Shade Bob Mask.

Remember, that this command supports the hot spot of the bob image.

Also see:

Shade Bob Up

Shade Bob Mask

Shade Bob Planes

1.274 Command: Shade Bob Planes

```
Shade Bob Planes amount
```

Set the amount of
Bitplanes
to be cycled through. This can be used to
protect the graphics in higher bitplanes from the influences of Shade
Bobs. ’amount’ sets the number of bitplanes, that should be drawn in and
must be a value between 1 and 6.

Shade Bob Up

Shade Bob Down

Shade Bob Mask

1.275 Command: Shade Bob Up

```
Shade Bob Up screen,x,y,image
```

This instruction draws a Shade Bob on the screen numbered ’screen’ at the
coordinates x,y. This bob only increases the colour indexes only. ’image’
holds the image number of the sprite bank which should be used to draw the
bob. Either the mask or the first bitplane of the object is used for this
process, according to the setting of Shade Bob Mask.

Remember, that this command supports the hot spot of the bob image.

Also see:

Shade Bob Down

Shade Bob Mask

Shade Bob Planes

1.276 Command: Shade Pix

```
                Shade Pix x,y
Shade Pix x,y,planes
```

```
This instructions increases the colour value at the given point x,y on the
current screen. If the highest colour is reached, the colour is resetted
to zero. The optional parameter 'planes' holds the number of
                 Bitplanes
                to be cycled and may range from 1 to 6.
```

1.277 Command: Shade Bob Mask

```
                Shade Bob Mask flag
```

```
Using Shade Bob Mask you may specify, if the mask of a object or the first
bitplane is to be taken for drawing. If flag is set to zero, bitplane 0
is used for the Shade Bob Up and Shade Bob Down instructions, other values
for flag instruct the commands to use the mask of the object.

Also see:


                 Shade Bob Up

                 Shade Bob Down

                 Shade Bob Planes
```

1.278 Command: Smouse Speed

```
                Smouse Speed value
```

```
Sets the speed of the mouse. value therefore represents the factor by
which power of 2 the mouse should be slowed down. 0 is the maximum speed
whereas 1 is about the speed of the normal AMOS mouse. Higher values than
4 are not sensible.

Also see:


                 =X Smouse

                 =Y Smouse

                 =Smouse Key

                 Limit Smouse

                 Smouse X
Smouse Y
```

1.279 Command: Smouse X

```
Smouse X xpos
```

Sets the x coordinate of the mouse to xpos.

Also see:

```
=X Smouse
=Y Smouse
=Smouse Key
Limit Smouse
Smouse Speed
Smouse Y
```

1.280 Command: Smouse Y

```
Smouse Y ypos
```

Sets the y coordinate of the mouse to ypos.

Also see:

```
=X Smouse
=Y Smouse
=Smouse Key
Limit Smouse
Smouse Speed
Smouse X
```

1.281 Command: Splinters Back

```
Splinters Back
```

Saves the background, on which the Splinters are to be drawn in the next step.

Also see:

Splinters Bank

Splinters Colour

Splinters Del

Splinters Do

Splinters Draw

Splinters Fuel

Splinters Gravity

Splinters Init

Splinters Limit

Splinters Max

Splinters Move

=Splinters Active

1.282 Command: Splinters Bank

```
Splinters Bank bank,splinum
```

Reserves a memory bank for a maximum of ’splinum’ Splinters. Each Splinter requires 22 bytes of memory.

Also see:

Splinters Back

Splinters Colour

Splinters Del

Splinters Do

Splinters Draw

Splinters Fuel

Splinters Gravity

Splinters Init

Splinters Limit

Splinters Max

Splinters Move

=Splinters Active

1.283 Command: Splinters Colour

```
Splinters Colour bkcolour,planes
```

Determinates, which colour is to be left, after a Splinter has released a dot from that point. The 'planes' parameter describes the number of bitplanes to be taken for the Splinters operation. Normally, this value should be equal to the number of available bitplanes. However, it can be useful for some effects to reduce this value.

Also see:

Splinters Back

Splinters Bank

Splinters Del

Splinters Do

Splinters Draw

Splinters Fuel

Splinters Gravity

Splinters Init

Splinters Limit

Splinters Max

Splinters Move

=Splinters Active

1.284 Command: Splinters Single Del/Splinters Double Del

```
                Splinters Single Del
Splinters Double Del

Clears the splinters from the screen again. As the clearing process
must either wipe the pre-last pixels from the screen (when using Double
Buffering), or the last pixels (with Single Buffered screens), you have
to take the appropriate command for the right screen type.

The background is automatically restored.

Also see:


                 Splinters Back

                 Splinters Bank

                 Splinters Colour

                 Splinters Do

                 Splinters Draw

                 Splinters Fuel

                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 Splinters Move

                 =Splinters Active
```

1.285 Command: Splinters Single Do/Splinters Double Do

```
                Splinters Single Do
Splinters Double Do

Clears, draws and moves all Splinters in one single step. For Double
Buffered screens you have to call the command Splinters Double Do, for
normal ones Splinters Single Do. If, however, you want to control the
process by hand, so you have to call the commands Splinters Single Del
or Splinters Double Del, then Splinters Move, Splinters Back and
Splinters Draw in this order.

Also see:
```

```
                 Splinters Back

                 Splinters Bank

                 Splinters Colour

                 Splinters Del

                 Splinters Draw

                 Splinters Fuel

                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 Splinters Move

                 =Splinters Active
```

1.286 Command: Splinters Draw

```
                Splinters Draw
```

Draws the Splinters onto the screen.

Also see:

```
                 Splinters Back

                 Splinters Bank

                 Splinters Colour

                 Splinters Del

                 Splinters Do

                 Splinters Fuel

                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max
```

Splinters Move

=Splinters Active

1.287 Command: Splinters Fuel

```
Splinters Fuel time
```

This command is used to set the amount of time the Splinters move over the screen until they disappear automatically. The 'time' parameter holds the number of steps the splinters are moved before they vanish. If you set 'time' to 0, the Splinters only disappear at the edges of the screen.

Also see:

Splinters Back

Splinters Bank

Splinters Colour

Splinters Del

Splinters Do

Splinters Draw

Splinters Gravity

Splinters Init

Splinters Limit

Splinters Max

Splinters Move

=Splinters Active

1.288 Command: Splinters Gravity

```
Splinters Gravity sx,sy
```

Set the direction which the Splinters shall drift into. 'sx' contains a value which is added each step to the horizontal speed and may be a positive number for right drift or a negative one for left movement. Same with 'sy', but this one has effect on the vertical speed.

Also see:

```
                  Splinters Back

                  Splinters Bank

                  Splinters Colour

                  Splinters Del

                  Splinters Do

                  Splinters Draw

                  Splinters Fuel

                  Splinters Gravity

                  Splinters Init

                  Splinters Limit

                  Splinters Max

                  Splinters Move

                  =Splinters Active
```

1.289 Command: Splinters Init

```
                 Splinters Init
```

```
Initialises the Splinters. They are fed with the coordinates and speeds
you have specified earlier.

Also see:
```

```
                  Splinters Back

                  Splinters Bank

                  Splinters Colour

                  Splinters Del

                  Splinters Do

                  Splinters Draw

                  Splinters Fuel

                  Splinters Gravity
```

```
                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 Splinters Move

                 =Splinters Active
```

1.290 Command: Splinters Limit

```
                Splinters Limit
Splinters Limit x1,y1 To x2,y2

Sets the limits for the Splinters to a rectangluar area on the screen. If
you don’t give any parameters, AMCAF uses the limits of the current
screen.

Also see:


                 Splinters Back

                 Splinters Bank

                 Splinters Colour

                 Splinters Del

                 Splinters Do

                 Splinters Draw

                 Splinters Fuel

                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 Splinters Move

                 =Splinters Active
```

1.291 Command: Splinters Max

```
                Splinters Max amount
```

Changes the max. amount of new Splinters to appear on each step. Therefore a pulsing effect can be avoided. If the value ’amount’ is set to 0, no more Splinters are created. When set to −1, there won’t be a limit.

Also see:

Splinters Back

Splinters Bank

Splinters Colour

Splinters Del

Splinters Do

Splinters Draw

Splinters Fuel

Splinters Gravity

Splinters Init

Splinters Limit

Splinters Max

Splinters Move

=Splinters Active

1.292 Command: Splinters Move

```
                Splinters Move
```

This command moves the Splinters one step.

Also see:

Splinters Back

Splinters Bank

Splinters Colour

Splinters Del

Splinters Do

```
                 Splinters Draw

                 Splinters Fuel

                 Splinters Gravity

                 Splinters Init

                 Splinters Limit

                 Splinters Max

                 =Splinters Active
```

1.293 Command: Td Stars Accelerate

```
                Td Stars Accelerate On
Td Stars Accelerate Off

Determinates, if the stars are to be accelerated during the flight.

Also see:


                 Td Stars Bank

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Limit

                 Td Stars Move

                 Td Stars Origin

                 Td Stars Planes
```

1.294 Command: Td Stars Bank

```
                Td Stars Bank bank,stars

This instruction reserves a bank for 3D stars. The 'bank' parameter holds
```

```
the number of the memory bank which should be used for the stars. 'stars'
contains the number of stars to be saved in this bank.

Each star consumes 12 bytes of memory.

Also see:


                 Td Stars Accelerate

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Limit

                 Td Stars Move

                 Td Stars Origin

                 Td Stars Planes
```

1.295 Command: Td Stars Single Del/Td Stars Double Del

```
                Td Stars Single Del
Td Stars Double Del

Wipes the stars from the screen. You have to distinguish between Single
and Double Buffered screens. Use Td Stars Double Del on Double Buffered

Note: The background is not saved during the drawing process.

Also see:


                 Td Stars Accelerate

                 Td Stars Bank

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Limit
```

Td Stars Move

Td Stars Origin

Td Stars Planes

1.296 Command: Td Stars Single Do/Td Stars Double Do

```
                Td Stars Single Do
Td Stars Double Do
```

Clears, draws and moves all the stars in one single step. For Double Buffered screens you have to call the command Td Stars Double Do, for normal ones Td Stars Single Do. If, however, you want to control the process by hand, so you have to call the commands Td Stars Single Del or Td Stars Double Del, then Td Stars Move and Td Stars Draw in this order.

Also see:

Td Stars Accelerate

Td Stars Bank

Td Stars Del

Td Stars Draw

Td Stars Gravity

Td Stars Init

Td Stars Limit

Td Stars Move

Td Stars Origin

Td Stars Planes

1.297 Command: Td Stars Draw

```
                Td Stars Draw
```

This command draws all the stars onto screen. To clear the stars again you have to use either Td Stars Single Del or Td Stars Double Del according to the screen type.

Also see:

Td Stars Accelerate

Td Stars Bank

Td Stars Del

Td Stars Do

Td Stars Gravity

Td Stars Init

Td Stars Limit

Td Stars Move

Td Stars Origin

Td Stars Planes

1.298 Command: Td Stars Gravity

```
Td Stars Gravity sx,sy
```

Set the direction which the stars shall drift into. 'sx' contains a value which is added each step to the horizontal speed and may be a positive number for right drift or a negative one for left movement. Same with 'sy', but this one has effect on the vertical speed.

Normally, these values are set to 0, but you could use other values in conjunction with Td Stars Accelerate to create an effect of spreading sparks.

Also see:

Td Stars Accelerate

Td Stars Bank

Td Stars Del

Td Stars Do

Td Stars Draw

Td Stars Init

Td Stars Limit

Td Stars Move

```
                 Td Stars Origin

                 Td Stars Planes
```

1.299 Command: Td Stars Init

```
                Td Stars Init
```

Initialises the stars. That means, that the stars are moved by random values to avoid that they all start in the origin. This command should therefore be called once after all parameters have been set.

Also see:

```
                 Td Stars Accelerate

                 Td Stars Bank

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Limit

                 Td Stars Move

                 Td Stars Origin

                 Td Stars Planes
```

1.300 Command: Td Stars Limit

```
                Td Stars Limit
Td Stars Limit x1,y1 To x2,y2
```

Limits the stars into a specific area on the screen. If no parameters are given, the full screen sizes are used.

x1,y1 and x2,y2 create a rectangular region, in which the stars will be drawin in. These coordinates must lie WITHIN the screen dimensions, otherwise the stars could corrupt your memory.

Also see:

```
                 Td Stars Accelerate

                 Td Stars Bank

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Move

                 Td Stars Origin

                 Td Stars Planes
```

1.301 Command: Td Stars Move

```
                Td Stars Move
Td Stars Move [star]

The Td Stars Move instruction moves all the stars by on step. To see the
result, you have to draw the stars with a call to Td Stars Draw.

The second version of this command is not so much used. You can call it to
move a single star, the star with the number 'star'.

Also see:


                 Td Stars Accelerate

                 Td Stars Bank

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Limit

                 Td Stars Origin

                 Td Stars Planes
```

1.302 Command: Td Stars Origin

```
                Td Stars Origin x,y
```

Sets the origin, where stars start from, as soon as they have left the screen. The coordinates x,y must lie on the screen and within the drawing area, you have defined using Td Stars Limit earlier.

The Td Stars Limit instruction automatically places the origin in the middle of the specified area. Therefore, this command has to be placed after Td Stars Limit.

Also see:

Td Stars Accelerate

Td Stars Bank

Td Stars Del

Td Stars Do

Td Stars Draw

Td Stars Gravity

Td Stars Init

Td Stars Limit

Td Stars Move

Td Stars Planes

1.303 Command: Td Stars Planes

```
                Td Stars Planes pl1,pl2
```

Td Stars Planes is used to specify the
Bitplanes
which the stars should
be drawn on. Normally, these are the bitplanes 0 and 1, but these defaults can be modified with this instruction.

Also see:

Td Stars Accelerate

```
                 Td Stars Bank

                 Td Stars Del

                 Td Stars Do

                 Td Stars Draw

                 Td Stars Gravity

                 Td Stars Init

                 Td Stars Limit

                 Td Stars Move

                 Td Stars Origin
```

1.304 Command: Turbo Draw

```
                Turbo Draw x1,y1 To x2,y2,c
Turbo Draw x1,y1 To x2,y2,c,bitplanes

This instruction replaces the AMOS Draw command. The x1,y1 coordinates
represent the starting point of the line and x2,y2 the ending point. The
coordinates needn't to be on the screen, the line is clipped
automatically. The colour of the line 'c' must be given, whereas the
'bitplanes' parameter is optional. It determinates, into which
                 Bitplanes
                the line should be drawn. So Bit 0 represents bitplane 0, bit 1  ↩
                    represents
bitplane 1 etc. If the corresponding bit is set, a line will be drawn in
the bitplane, otherwise not. If 'bitplanes' is ommitted, every bitplane is
drawn into. Turbo Draw supports a line pattern, which can be changed using
the AMOS Set Line command.

With Turbo Draw, you can draw even more and special lines, so-called
Blitter Lines. These lines are only drawn with one dot in each horizontal
line and are used to create polygons which can then be filled with the
blitter chip. To switch to this mode, the 'bitplane' parameter must be
given as negative number, everything else can remain as it is. Please note
that an extra line is drawn automatically if the blitter line leaves the
right boundary of the screen.

Also see:


                 Blitter Fill
```

1.305 Command: Turbo Plot

```
                Turbo Plot x,y,c
```

Draws a dot at the coordinates x,y using the colour 'c'. This command is about 3-14 factors faster than the AMOS plot command, but does not care about logical combinations which could be defined with Gr Writing.

Also see:

```
                 =Turbo Point
```

1.306 Command: Vec Rot Angles

```
                Vec Rot Angles angx,angy,angz
```

Sets the three viewing angles, which are used for the vector rotation. 'angx' represents the angle, the coordinate is to be rotated along the X-axis. Same with 'angy' and 'angz', which are related to the Y- and the Z-axis. The angles are -like with Qsin and Qcos- neither values in degree nor radians format, but a revolution (360 degrees) equals the value 1024.

After you have given the angles and the new position, you can calculate the new vector matrix using the Vec Rot Precalc command.

Also see:

```
                 Vec Rot Pos

                 Vec Rot Precalc

                 =Vec Rot X

                 =Vec Rot Y

                 =Vec Rot Z

                 =Qsin

                 =Qcos
```

1.307 Command: Vec Rot Pos

```
                Vec Rot Pos posx,posy,posz
```

Using Vec Rot Pos you can change the position of the camera. 'posx' and 'posy' hold the movement position of the camera in X and Y direction. The value in 'posz' contains the distance to the camera.

There is no need to recalculate the matrix if you only change the position of the camera.

Also see:

Vec Rot Angles

Vec Rot Precalc

=Vec Rot X

=Vec Rot Y

=Vec Rot Z

1.308 Command: Vec Rot Precalc

```
Vec Rot Precalc
```

This command must be called before every vector rotations, always after you have changed the viewing angle. It calculates a internal matrix that holds the results of often needed calculations which are required for the rotation in three dimensions.

After you have called Vec Rot Precalc, you can use the Vec Rot X,Y and Z functions.

Also see:

Vec Rot Pos

Vec Rot Angles

=Vec Rot X

=Vec Rot Y

=Vec Rot Z

1.309 Command: Wload

```
Wload file$,bank
```

This command loads the file named ’file$’ completely into memory, storing it in bank number ’bank’. The bank is defined as
’Work’
.

```
If 'bank' is a negative number, the file is loaded into
                 Chip ram
                 instead.

Wload could be replaced by the following commands:

Open In 1,FILE$ : LE=Lof(1) : Close 1
Reserve As Work BANK,LE
Bload FILE$,BANK

Also see:


                 Dload

                 Bank Temporary
```

1.310 Command: Write Cli

```
Write Cli text$

The Write Cli instruction writes the string 'text$' into the CLI window,
AMOS or your program has been start from. If this window does not exist,
no text will be output.
```

1.311 Command: Wsave

```
                Wsave file$,bank

Saves the bank with the number 'bank' as file named 'file$' onto the
current disk drive. This file contains no AMOS overhead, that means that
only the pure binary data is saved.

Also see:


                 Dsave
```

1.312 Overview to all available commands and functions

```
                Functions:


                 =Aga Detect
                 - Checks if the computer has AGA chipset

                 =Amcaf Base
                 - Gives back the address of the AMCAF data base
```

```
=Amcaf Length
- Returns the length of the AMCAF data base

=Amcaf Version$
- Returns an AMCAF version string

=Amos Cli
- Returns the CLI number of AMOS.

=Amos Task
- Returns the address of the AMOS task structure

=Asc.l
- Converting a
Long string
into a number

=Asc.w
- Converting a
Word string
into a number

=Bank Checksum
- Calculates a checksum of a bank

=Bank Name$
- Returns the name of a bank

=Best Pen
- Calculates the nearest pen for $RGB

=Binexp
- Exponential function on basis of two

=Binlog
- Logarithmic function on basis of two

=Blitter Busy
- Returns the blitter's current state

=Blue Val
- Calculates the blue value of a colour

=Cd Date$
- Creates a complete date string

=Cd Day
- Returns the day of the date

=Cd Month
- Calculates the month

=Cd String
- Converts a string into the date stamp

=Cd Weekday
```

```
- Gets the weekday from the date stamp

=Cd Year
- Extracts the year from a date

=Chr.l$
- Creating a
Long string

=Chr.w$
- Creating a
Word string

=Command Name$
- Acquires the name of the program

=Cop Pos
- Returns the current address of the copper list

=Count Pixels
- Counts the number of pixels in a specific area

=Cpu
- Returns the number of the fitted CPU

=Ct Hour
- Extracts the hour from a time

=Ct Minute
- Returns the minute of a time stamp

=Ct Second
- Calculates the second of a time

=Ct String
- Evaluates a string into a time code

=Ct Tick
- Extracts the 1/50 from the time.

=Ct Time$
- Creates a complete time string

=Current Date
- Acquires the current date

=Current Time
- Acquires the current time

=Cutstr$
- Cuts out a piece of a string

=Disk State
- Returns the state of a disk device

=Disk Type
- Returns the type of a volume
```

```
=Dos Hash
- Calculates the hash value of a file

=Even
- Returns, if a number is even

=Examine Next$
- Reads the next entry in a directory

=Extpath$
- Appends a "/" to a path if required

=Filename$
- Returns the filename of a full path

=Font Style
- Getting the attributes of a font

=Fpu
- Acquires the id number of an coprocessor

=Glue Colour
- Generates a colour using the three colour values

=Green Val
- Calculates the green value of a colour

=Ham Best
- Calculates the best colour in
HAM mode

=Ham Colour
- Calculates a colour in
HAM mode

=Ham Point
- Returns the RGB value of a ham pixel

=Insstr$
- Inserts a string into a string

=Io Error
- Returns the last dos error code

=Io Error$
- Returns a dos errorstring

=Itemstr$
- Returns an 'item' contained in a string

=Lsl
- Quick multiplication by a power of two

=Lsr
- Quick division by a power of two
```

```
=Lsstr$
- Returns a right ajusted number

=Lzstr$
- Returns a right ajusted number with leading zeros

=Mix Colour
- Mixes two colours

=Nfn
- No effect

=Odd
- Returns, if a number is odd

=Object Blocks
- Returns the length of a file in blocks

=Object Comment$
- Gives back the filenote of an
Object

=Object Date
- Returns the date of creation of an
Object

=Object Name$
- Returns the name of an
Object

=Object Protection
- Returns the
Protection flags
of an
Object

=Object Protection$
- Returns a
Protection flags
string

=Object Size
- Gives back the length of a file

=Object Time
- Returns the time of creation of an
Object

=Object Type
- Returns the type of an
Object

=Pal Get
- Reads a saved palette entry

=Path$
- Returns the directory of a full path
```

```
=Pattern Match
- Compares a string with a certain pattern

=Pfire
- Check if fire button is pressed

=Pjdown
- Check if joystick is pressed down

=Pjleft
- Check if joystick is pressed left

=Pjoy
- Acquire direction of a joystick

=Pjright
- Check if joystick is pressed right

=Pjup
- Check if joystick is pressed up

=Pt Cinstr
- Returns the current instrument being played

=Pt Cnote
- Gets the frequency of the current instrument

=Pt Cpattern
- Gets the current song position

=Pt Cpos
- Returns the current pattern line

=Pt Data Base
- Gets the address of the PT-DataBase

=Pt Instr Address
- Returns the address of an instrument

=Pt Instr Length
- Returns the length of an instrument

=Pt Signal
- Checking for signals from the music

=Pt Vu
- Returns the current Vumeter value

=Qarc
- Fast arc function

=Qcos
- Fast cosine function

=Qrnd
- Fast replacement for Rnd
```

```
=Qsin
- Fast sine function

=Qsqr
- Fast replacement for Sqr

=Red Val
- Calculates the red value of a colour

=Replacestr$
- Replaces a string with another one

=Rgb To Rrggbb
- Converts a ECS-colour into
AGA colour format

=Rrggbb To Rgb
- Converts a
AGA colour
into a ECS-colour value

=Scanstr$
- Returns the name of a key

=Scrn Bitmap
- Returns the screen bitmap address

=Scrn Layer
- Returns the screen layer address

=Scrn Layerinfo
- Returns the screen layerinfo address

=Scrn Rastport
- Returns the screen rastport address

=Scrn Region
- Returns the screen region address

=Sdeek
- Deeking a signed
word

=Smouse Key
- Checks the mouse key

=Speek
- Peeking a signed
byte

=Splinters Active
- Returns how many splinters are still moving

=Tool Types$
- Reads the Tool Types of an icon
```

```
                 =Turbo Point
                 - Fast replacement for Point

                 =Vclip
                 - Restricts a value to a given range

                 =Vec Rot X
                 - Calculates the 2D X value

                 =Vec Rot Y
                 - Determinates Y value

                 =Vec Rot Z
                 - Returns the Z coordinate

                 =Vin
                 - Tests, if the value is within a range

                 =Vmod
                 - Does a modulo operation on a value

                 =Wordswap
                 - Swapping the upper and lower 16 bits

                 =X Raster
                 - Gets the X position of the raster beam

                 =X Smouse
                 - Reads the x coordinate

                 =Xfire
                 - Reads out the fire buttons on a gamepad

                 =Y Raster
                 - Gets the Y position of the raster beam

                 =Y Smouse
                 - Reads the y coordinate of the mouse

Commands:


                 Amcaf Aga Notation
                 - Toggle
                 AGA-Amiga
                 colour format

                 Audio Free
                 - Frees the audio device

                 Audio Lock
                 - Reserves the audio device

                 Bank Code Add.y
                 - Additional algorithm encoding

                 Bank Code Mix.y
```

```
- Mix between Add und Xor

Bank Code Rol.y
- Rotation to the left

Bank Code Ror.y
- Rotation to the right

Bank Code Xor.y
- Xor algorithm encoding

Bank Copy
- Copies a bank

Bank Delta Decode
- Removes the delta-encoding of a bank

Bank Delta Encode
- Prepares a bank with delta-encoding

Bank Name
- Changes the name of a bank

Bank Permanent
- Makes a bank
Permanent

Bank Stretch
- Extends a bank after it has been reserved

Bank Temporary
- Makes a bank
Temporary

Bank To Chip
- Moves a bank into
Chip ram

Bank To Fast
- Moves a bank into
Fast ram

Bcircle
- Drawing a circle to fill it using the blitter

Blitter Clear
- Clearing a bitplane with the help of the blitter

Blitter Copy
- Copying and modifying a bitplane

Blitter Copy Limit
- Setting the Blitter Copy area

Blitter Fill
- Filling polygons using the blitter
```

```
Blitter Wait
- Waiting for the blitter has finished his task

Bzoom
- Zooms a region

Change Bank Font
- Setting the screen font using a font bank

Change Font
- Loading a font directly from disk

Change Print Font
- Changing the font that is used by Print.

Convert Grey
- Creates a grey scale picture

Coords Bank
- Reserves a bank to store the coordinates

Coords Read
- Reading the coordinates into a bank

Dload
- Loads a file
Permanently

Dsave
- Saves a file to disk

Examine Dir
- Inits the reading of a drawer

Examine Object
- Gets all information about an
Object

Examine Stop
- Stops the reading of a directory

Exchange Bob
- Swaps the two images in the sprite bank

Exchange Icon
- Swaps the two images in the icon bank

Extdefault
- Calls the default routine of an extension

Extreinit
- Tries to revoke a extension

Extremove
- Removes a extension from memory

Fcircle
```

```
- Draws a filled circle

Fellipse
- Draws a filled ellipse

File Copy
- Copies a file

Flush Libs
- Frees as much as possible memory

Ham Fade Out
- Fades out a ham picture

Imploder Load
- Loads and decrunches a FileImploder file

Imploder Unpack
- Decrunches a FileImploder bank

Launch
- Starts a new process

Limit Smouse
- Defines the movement region

Make Bank Font
- Creating a font bank

Make Pix Mask
- Picks up a mask for the shifting process

Mask Copy
- Screen Copy with a mask

Nop
- No effect

Open Workbench
- Reopens the workbench again

Pal Get Screen
- Saves the palette of a screen

Pal Set
- Changes an entry of a saved palette

Pal Set Screen
- Sets the palette of a screen

Pal Spread
- Spreads from one colour entrie to another

Pix Brighten
- Increase colour indexes (not
cyclic
)
```

```
 Pix Darken
 - Decrease colour indexes (not
 cyclic
)

 Pix Shift Down
 - Decrease colour indexes (
 cyclic
)

 Pix Shift Up
 - Increase colour indexes (
 cyclic
)

 Ppfromdisk
 - Loads and unpacks a powerpacked file

 Pptodisk
 - Packs and saves a file as PP20

 Ppunpack
 - Unpacks a powerpacked file

 Protect Object
 - Modifies the
 Protection bits
 of an
 Object

 Pt Bank
 - Sets the bank for the use with Pt Instr Play

 Pt Cia Speed
 - Changing the replaying speed

 Pt Continue
 - Restarts a previously stopped music

 Pt Instr Play
 - Plays an instrument of a protracker module

 Pt Play
 - Replays a module

 Pt Raw Play
 - Plays a chunk of memory as sound sample

 Pt Sam Bank
 - Sets the bank to use with AMOS samples

 Pt Sam Freq
 - Changes the replaying speed of a sample

 Pt Sam Play
 - Replays a sample from an AMOS Sam Bank
```

```
Pt Sam Stop
- Stops the sfx on specific audio channels

Pt Sam Volume
- Sets the volume of a sound effect

Pt Stop
- Stops the current music

Pt Voice
- Toggling the audio channels

Pt Volume
- Setting the volume of the music

Rain Fade
- Fades a rainbow out or to another one

Raster Wait
- Waits for a specific raster position

Reset Computer
- Resets your computer

Set Ntsc
- Switches to the 60Hz NTSC mode

Set Object Comment
- Sets the filenote of an
Object

Set Object Date
- Sets the date of an
Object

Set Pal
- Switches back to 50Hz PAL mode

Set Rain Colour
- Changes the affecting colour of a rainbow

Set Sprite Priority
- Changes the sprite priority in Dual playfield mode

Shade Bob Down
- Places a Shade Bob that decreases the colours

Shade Bob Mask
- Determinate the image to use for the bobs

Shade Bob Planes
- Setting the number of bitplanes to use

Shade Bob Up
- Places a Shade Bob that increases the colours
```

```
Shade Pix
- Plots a shade pixel

Smouse Speed
- Sets the speed of the mouse

Smouse X
- Sets the x coordinate of the mouse

Smouse Y
- Sets the y coordinate

Splinters Back
- Gets the background pixels

Splinters Bank
- Reserves memory for the splinters

Splinters Colour
- Sets the to-use colours

Splinters Del
- Clears the splinters

Splinters Do
- Do a complete drawing process

Splinters Draw
- Draws the splinters to the screen

Splinters Fuel
- Sets the range of a splinter

Splinters Gravity
- Changes the gravity

Splinters Init
- Initialises the splinters bank

Splinters Limit
- Changes the limits of the splinters

Splinters Max
- Sets the maximum of new appearing splinters

Splinters Move
- Moves the splinters

Td Stars Accelerate
- Toggles the acceleration

Td Stars Bank
- Reserves some memory for the stars

Td Stars Del
- Clears the stars from the screen
```

```
Td Stars Do
- Does a complete drawing process

Td Stars Draw
- Draws the stars

Td Stars Gravity
- Determins the gravity force

Td Stars Init
- Inits the stars

Td Stars Limit
- Sets the limits of the stars

Td Stars Move
- Moves the stars

Td Stars Origin
- Places the origin of the stars

Td Stars Planes
- Selects the planes to be used for the stars

Turbo Draw
- Fast replacement for Draw

Turbo Plot
- Fast replacement for Plot

Vec Rot Angles
- Sets the viewing angles

Vec Rot Pos
- Positions the camera

Vec Rot Precalc
- Calculates the precalc matrix

Wload
- Loads a file
Temporarily

Write Cli
- Writes something into the cli window

Wsave
- Saves a file to disk
```