# Tools

Tor Erik Ottinsen

| COLLABORATORS | | | |
|---|---|---|---|
| | *TITLE* :<br><br>Tools | | |
| *ACTION* | *NAME* | *DATE* | *SIGNATURE* |
| WRITTEN BY | Tor Erik Ottinsen | February 12, 2023 | |

| REVISION HISTORY | | | |
|---|---|---|---|
| NUMBER | DATE | DESCRIPTION | NAME |
| | | | |

# Contents

# Chapter 1

# Tools

## 1.1 Tools - An extension for AMOSPro

```
                                                  AMOSPro Tools Extension v1.00

     Release Date: 26.02.97
```

## 1.2 Introduction

The AMOSPro Tools extension was originally developed for personal use only.
As time passed it grew as I needed more commands. I think some of the
commands might be of use to others as well, so i wrote this doc and
released it to the public domain.

My apologies for any bad english you might find :)

## 1.3 Installing Tools

The following files should be in the archive:

```
  AMOSPro_Tools.Lib
  AMOSPro_Tools.Guide
  AMOSPro_Tools.Guide.info
```

To install the Tools Extension for AMOS Professional, do the following:

 - Copy AMOSPro_Tools.Lib to AMOSPro_System:APSystem/

 - Load AMOSPro, and select "Set Interpreter" from the config menu.
   Load the default configuration and click on loaded extensions.
   Enter 'AMOSPro_Tools.Lib' into slot 23 and save the default
   configuration.

 - Quit and restart AMOSPro.

## 1.4 Commands to deal with arrays

When you dimension an array in AMOSPro, you have three choices. ↩
Either you
make it a string array, an array of longs, or an array of floats. This is ok
for most uses, but what if you need to dimension a very large array
containing normal numbers. Such arrays are often used in map based games.
Often, the numbers stored in the array never get any bigger than 255 and
could in fact fit into a byte. If such is the case, you are using four times
the memory actually needed. This is where the array commands come into use.
They allow you to create a two-dimensional array consisting of bytes, not
longwords as in AMOSPro arrays. The array is stored in a memory bank.

Commands to deal with arrays:

```
Set Array Bank
 -  Choose an arraybank

= Array Bank
 -  Return the currently used arraybank

Array Dim
 -  Dimension an array
```

```
                 Array Set
                  -  Set an element of the array

                 = Array Get
                  -  Get an element of the array
```

## 1.5 Choosing which bank to use

```
                Set Array Bank BANK
```

This command tells the Tools extension which memorybank should be used by
the other Array Commands. By changing the array bank in the middle of the
program, you can in fact have two or more arrays.
The default is bank number 23.

See also:

```
                 = Array Bank
```

## 1.6 Getting the currently used array bank

```
                = Array Bank
```

Returns the number of the currently used array bank.

See also:

```
                 Set Array Bank
```

## 1.7 Dimensioning an array

```
                Array Dim SX, SY
```

This command reserves a memory bank to use for an array. SX and SY refers
to the size of the array. The AMOSPro Array equalient of this command would
be Dim _ARRAY(SX,SY). Which memory bank to use can be altered with the
Set Array Bank Command. The default is bank number 23.

```
NOTE: If the memory bank already exists,
      it will be erased before the array bank is created!
```

See also:

```
                 Set Array Bank
```

## 1.8 Setting an element of the array

```
                Array Set X, Y, DATA

Sets the element at position (X,Y) in the current array to DATA.
DATA should be a positive number between 0 and 255.

See also:


                 Array Dim

                 = Array Get
```

## 1.9 Getting an element of the array

```
                = Array Get (X, Y)

Returns the data at position (X,Y) in the current array.

See also:


                 Array Dim

                 Array Set
```

## 1.10 Commands to move data to or from memory

```
                The commands in this section move data (bytes, words, longwords,  ←
                   strings)
from variables to memory and vice versa. Using these commands to store
data in a memorybank, then saving it to a file, represents a much better
way of storing data than using Print # and Input #.

Commands to deal with data in memory:


                 Set Pos
                  -  Setting the current memory position

                 = Get Pos
                  -  Getting the current memory position

                 Add Pos
                  -  Incrementing the current memory position


                 Set Byte
                  -  Storing a byte in memory

                 Set Word
```

```
 -  Storing a word in memory

Set Long
 -  Storing a longword in memory

Set String
 -  Storing a string in memory

Set Crypt
 -  Storing an encrypted string in memory


= Get Byte
 -  Getting a byte from memory

= Get Word
 -  Getting a word from memory

= Get Long
 -  Getting a longword from memory

= Get String
 -  Getting a string from memory

= Get Crypt
 -  Getting an encrypted string from memory
```

## 1.11 Setting the current memoryposition

```
Set Pos ADDRESS
```

This command will set the
current memory position
to ADDRESS.
You should always set the current memory position before using any of the other Set and Get commands. Not doing this may crash the computer, as data will be written or read from random memory addresses.

See also:

```
= Get Pos

Add Pos
```

## 1.12 Getting the current memoryposition

```
= Get Pos
```

Returns the
current memory position
.

See also:

Set Pos

Add Pos

## 1.13 Incrementing the current memoryposition

```
Add Pos INCREMENT
```

Increments the current memory position with INCREMENT. To decrement the current memory position, use a negative INCREMENT. This command can be useful for skipping data.

See also:

Set Pos

= Get Pos

## 1.14 Storing a byte in memory

```
Set Byte BYTE
```

Set Byte will store BYTE at the current memory position. After the byte is stored, the current memory position will be incremented by 1.

See also:

Set Pos

= Get Byte

## 1.15 Storing a word in memory

```
Set Word WORD
```

Set Word will store WORD at the current memory position. After the word is stored, the current memory position will be incremented by 2.

See also:

Set Pos

= Get Word

## 1.16 Storing a longword in memory

```
Set Long LONGWORD
```

Set Long will store LONGWORD at the current memory position.
After the longword is stored, the current memory position will be incremented by 4.

See also:

Set Pos

= Get Long

## 1.17 Storing a string in memory

```
Set String STRING$
```

Set String will store STRING$ at the current memory position.
After the string is stored, the current memory position will be incremented by the length of the string + 2.

See also:

Set Pos

= Get String

## 1.18 Storing an encrypted string in memory

```
Set Crypt STRING$
```

Set Crypt will store STRING$ at the current memory position.

However before doing this, the string is encrypted making it unreadable.
The algorithm used for encryption isn't very secure, so please do not
store any sensitive data with this command.
After the string is stored, the current memory position will be
incremented by the length of the string + 2.

See also:

Set Pos

= Get Crypt

## 1.19 Getting a byte from memory

= Get Byte

Get Byte will return the byte stored at the
current memory position
.
After retrieving the byte, the current memory position will be
incremented by 1.

See also:

Set Pos

Set Byte

## 1.20 Getting a word from memory

= Get Word

Get Word will return the word stored at the
current memory position
.
After retrieving the word, the current memory position will be
incremented by 2.

See also:

Set Pos

Set Word

## 1.21 Getting a longword from memory

```
                = Get Long

Get Long will return the longword stored at the
                current memory position
                .
After retrieving the longword, the current memory position will be
incremented by 4.

See also:


                 Set Pos

                 Set Long
```

## 1.22 Getting a string from memory

```
                = Get String

Get String will return the string stored at the
                current memory position
                .
After retrieving the string, the current memory position will be
incremented by the length of the string + 2.

See also:


                 Set Pos

                 Set String
```

## 1.23 Getting an encrypted string from memory

```
                = Get Crypt

Get Crypt will return the string stored at the
                current memory position
                .
After retrieving the string, the current memory position will be
incremented by the length of the string + 2.
The string will be decrypted before it is returned.

See also:


                 Set Pos

                 Set Crypt
```

## 1.24 The current memory position

The current memory position is where the Set commands (Set Byte etc.) will store their data. The Get commands will also get their data from this address. All Set and Get commands will increase the current memory position by the length of their data after setting or getting. In other words, Set Byte and Get Byte will increase the current memory position with 1, Set Word and Get Word with 2, and so on.

## 1.25 Internal Interface Commands

The Tools Extension has got a number of interface commands used by my so far unreleased GUI System. These are internal commands of no use for anybody except me. I therefore choose to leave them undocumented.

## 1.26 Commands for various tasks

```
                Miscellanous commands:


                 = Range
                  -  Limit a number to a certain range


                 Encode
                  -  Encode a piece of memory

                 Decode
                  -  Decode a piece of memory
```

## 1.27 Limit a number to a certain range

= Range (A, MIN To MAX)

This command is a somewhat optimized version of the Range command in the Shuffle Extension. If the number A lies between MIN and MAX, A is returned. If A is less than MIN, MIN is returned. If A is greater than MAX, MAX is returned.

See also:

## 1.28 Encode a piece of memory

```
                Encode START, END, PASSWORD$
```

Encode will scramble the contents of memory between address START and END. PASSWORD$ is a password used for encryption. To unscramble, use the command

Decode with the same password.

See also:

                 Decode

## 1.29 Decode a piece of memory

```
                Decode START, END, PASSWORD$
```

Use decode to unscramble something which you have already scrambled with
Encode. START and END specifies the memoryarea you wish to decrypt.
PASSWORD$ should be the same password you used for encryption.

See also:

                 Encode

## 1.30 History

```
 Version 1.00     (27.02.97)

    \textdegree{} First public release
```

## 1.31 Disclaimer

Disclaimer:

I, the author, nor anybody else will take any responsibility whatsoever
for anything any of the files in this package might do to you, your computer,
or anything else. Use this product entirely at your own risk.

All files are PUBLIC DOMAIN, which means you might spread them all you want,
as long as you keep the archive complete, and do not modify any of the files.

## 1.32 Contacting the author

Please let me know how you like this extension, and if you have any problems.
I would also appreciate any comments or suggestions you might have.

To reach me you can use one of the following addresses:

```
S-mail:
  Tor Erik Ottinsen
  Stovnerlia 33
  0983, Oslo
  NORWAY

E-mail:
  ottinst@pc.iu.hioslo.no
```