PEEKs+POKEs

Simon N Goodwin

COLLABORATORS

	TITLE : PEEKs+POKEs	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Simon N Goodwin	October 30, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

PEEKs+POKEs

1.1 main

```
                    ******************************************************
          **  Structured PEEKs and POKEs for the Amiga Shell!  **
          ******************************************************
```

INTRODUCTION

Would you like to be able to read analogue joysticks, switch the audio filter, check library and device versions, reset MIDI devices, blink the power LED, check your Amiga chip set or CPU type, start and stop floppy drive motors or change the sex of the narrator device - all from the comfort of your shell or a Script file?

All these and much more - including numerous ways to crash the machine - are now available via structured PEEK and POKE extensions for the Amiga Shell.

These new, small, commands are either a great leap backwards or a small lunge forwards for the Amiga. They are freely distributable with stand-alone source in assembler, and run on any machine with at least Workbench 2.0.

SECTIONS IN THIS GUIDE

1.2 overview

OVERVIEW

Like old-fashioned BASIC PEEKs and POKEs, you can read or (if you must) write the bytes at any memory address. Unlike most BASICs, you can also read and write words (16 or 32 bits wide) and read null-terminated strings from RAM. The 'structured' extensions mean that addresses can be specified relative to the base of hardware or software resources, identified by name - so addresses can be relative to the start of CUSTOM chip or CIA registers, or any library, device or resource you choose to name.

Numeric values can be read and written in decimal, hexadecimal or binary bases. Decimal parameters may be signed or unsigned values. Indirect addressing is available, and comes in handy when looking through pointers in a structure.

These commands were written for three purposes:

(1) To allow low-level access to hardware and memory, from the CLI.
(2) To prevent the need to write little bits of code for each POKE.
(3) To experiment with some non-trivial ReadArgs parameter parsing.
(0) To annoy people who believe that POKEs are a thing of the past.

1.3 examples

SIMPLE EXAMPLES

These examples show the flexible format of various acceptable parameters:

```
POKE ciaa 0 2                   - Turns the audio filter off
POKE CIAA 0 0                   - Turns the audio filter on
PEEK L 4                        - Returns the EXEC library base address
PEEK CUSTOM 18                  - Controller 0 left paddle position
PEEK CUSTOM 19                  - Controller 0 right paddle position
PEEK Res potgo WORD 20          - Version number of POTGO.RESOURCE
peek word lib dos 22            - Revision number of DOS.LIBRARY
peek lib exec long 42           - Contents of EXEC cold-capture vector
```

```
 POKE CUSTOM WORD 48 511          - Sends MIDI RESET to the serial port
 POKE WORD dev="narrator" 48 90   - Sets speech rate to 90 words/minute
 peek LONG lib "graphics" 50      - Address of current Copper List
 POKE W DEV "narrator" 54 1       - Narrator attempts a female voice
 peek lib="exec" 62 long          - Returns the top of Chip Memory
 peek 62 word lib graphics        - Returns the current screen display mode
 poke w Dev narrator 62 16        - Set -12 dB Narrator speech volume
 PEEK W lib "graphics" 232        - Microseconds per scan line * 256
 peek lib graphics.library 236    - ChipSet revision, e.g. OCS/ECS/AGA
 POKE CIAB 256 119                - Start motor of DF0, stop other drives
 poke 256 129 ciab                - Stop all floppy disk drive motors
 PEEK long library "exec" 276     - Returns base address of this task
 peek LIBRARY="exec" w 296        - Returns ATTN_FLAGS (CPU type)
 POKE LIB "exec.library" 297 1    - Pretend this CPU is a 68010!
 Peek LIB "exec" 530              - Returns the vertical blanking frequency
 PEEK 531 lib exec                - Returns power supply frequency in Hertz
 PEEK LONG Library "exec" 568     - Number of timer 'E clocks' per second
 PEEK CIAB 2048                   - Low byte of 24 bit scan-line counter
 PEEK dev=parallel string long 10 - "parallel.device", or "pit.device" if
                                    MapDevice parallel 0 to pit 0 is active.
```

See Mapping The Amiga, The Amiga Guru Book and the Include files for lots more interesting and dangerous offsets. The 'best' and most hazardous ones are secret. If you're not sure, PEEK first and POKE afterwards (perhaps)! If you find any interesting or useful examples, please tell us about them.

Authors

1.4 warnings

IMPORTANT WARNING

Library, device and resource offsets can vary between software releases, but all these have worked on all Amigas tested so far - with the exception of the narrator device POKEs which are version dependent. If in doubt, check the documentation (include files) for the devices (etc.) you are using, and use some precautionary PEEKs to determine the version you've got before you POKE into it.

POTENTIAL PITFALLS

There is deliberately no check for uneven-addressed words or long words. These cause an odd address guru on 68000/68010 systems, but work fine on later processors including all the current 32 bit Amiga models.

Attempts to access non-existent memory on a Zorro 3 Amiga cause a bus error after a delay of a fraction of a second, unless the relevant area is remapped by the MMU. Utilities such as WarpKick can do this, if required.

Some debugging utilities are designed to prevent direct access to memory, restricting the usefulness (and potential for problems) of PEEK and POKE. These include Enforcer and phase 5's equivalent CyberGuard. If these are active when PEEK is used, access to protected addresses will return zero

and cause a 'hit' to be reported. POKE will cause a hit and no value will be stored.

1.5 arguments

DETAILED ARGUMENT SYNTAX

PEEK and POKE support (and may engender) lots of arguments, but almost all are optional. The simplest case is PEEK 4, which returns the byte from RAM address 4. PEEK LONG 4 returns the 32 bit contents of the address ExecBase. POKE 4 255 crashes the system by changing the first byte of that pointer!

First, a summary for those who grok Regular Expression templates:

```
PEEK       W=WORD/S,L=LONG/S,LIB=LIBRARY/K,DEV=DEVICE/K,RES=RESOURCE/K,
           C=CUSTOM/S,CIAA/S,CIAB/S,AT/S,ADDRESS/A,VALUE/A

POKE       W=WORD/S,L=LONG/S,LIB=LIBRARY/K,DEV=DEVICE/K,RES=RESOURCE/K,AT/S
           C=CUSTOM/S,CIAA/S,CIAB/S,H=HEX/S,BIN=BINARY/S,STRING/S,ADDRESS/A
```

Human beings may find it easier to understand the options by examining the examples, above, or by experimentation - the way most PEEKs and POKEs are traditionally worked out. What follows is a discussion of some less-obvious details of the way arguments are handled.

The Device option always selects unit zero, with 0 in the 'flags' field.

Numeric values may be specified in decimal, binary with a % prefix, or hexadecimal if prefixed with $, 0X or 0x. Excessive values are reduced modulo the transfer size, e.g. POKE 0 $123 stores $23 and POKE WORD 0 $123456 stores $3456. Thus conventional (decimal only) ReadArgs numeric parsing cannot be used for value and address arguments, so the /N does not appear in the argument template. Nonetheless, VALUE and ADDRESS arguments are numeric.

By default PEEK returns a decimal number, but it can be a string up to 32 bytes long (stopping after 32 bytes or at the first null) returned in quotes, if the STRING switch is supplied, a value in binary, or in hexadecimal, indicated by the HEX or H switch and $ prefix before the number for base 16. The BIN or BINARY switch sets base 2 and a % prefix.

The AT parameter indicates indirect addressing. The long word at the specified address is read and used as a pointer to the required data. Absolutely no validity checks are performed.

The STRING parameter expects that the argument is the offset of a pointer to the required string rather than the offset of the start of the string.

Numeric parameters are 32 bit signed or unsigned decimal values. Results are unsigned bytes or words, signed long words. The default data size is BYTE - this is not a switch. Words and bytes are reduced modulo 65536 and modulo 256 respectively. Device, Resource and library names are case- sensitive and the .suffix is assumed if not explicitly presented. All other keywords are case-insensitive. Quotes, verbose qualifiers and equals signs are optional. PEEK returns an ASCII string followed by a trailing space and a newline, or

nothing if an error occurs.

You may specify the transfer size and device/resource/library/hardware base in any order, so PEEK WORD CUSTOM 18 means the same as PEEK 18 WORD CUSTOM or PEEK W 18 CUSTOM or PEEK CUSTOM WORD 18. The choice is yours. Unfortunately qualifiers (LIB, LIBRARY, RES, DEVICE etc) must appear just before the name of the library, resource or device which you want to use, so you need to say PEEK LIB exec 530 as PEEK exec LIB 530 will not work. This limitation is imposed by Commodore's READARGS function.

Combinations of switches are additive, so PEEK LIB exec CUSTOM 2 reads the byte at ExecBase + $DFF000 + 2, - which is unlikely to be very useful!

1.6 scripts

SCRIPT EXAMPLES

PEEKs and POKEs can be useful in scripts, where they may be combined with conditional tests and arithmetic, using the other features of the Amiga shell. They can also be used from ARexx when the existing memory-access functions of ARexx are inadequate.

PEEK lib exec 297 returns a byte identifying the processor on the current computer. You can redirect this to an environment variable and use its value later in your shell script. For instance, this code checks the current CPU and writes an appropriate message if it's a 68040 and 68060:

```
PEEK lib=exec 297 >env:CPU
IF $CPU equ 127
  ECHO "68040 processor"
ELSE
  IF $CPU equ 255
    ECHO "68060 processor"
  ENDIF
ENDIF
```

A similar script on the author's machine selects a '68040' or '68060' banner when the machine is started, depending on the processor installed that day.

EVAL can be very useful with PEEK environment variables. For instance this sequence finds the start address of any Commodore Kickstart ROM:

```
PEEK LONG $FFFFEC >env:ROMsize
EVAL 65536*256-$ROMsize
```

The result from EVAL can be re-directed to an environment variable, as in the first line, if further processing is required.

1.7 files

FILES

One source file is used to implement both PEEK and POKE commands, depending on the setting of the symbol PEEK (1 for PEEK, 0 for POKE). Most of the comments in this document are also included in the assembler source, with lots of relatively trivial information about the precise implementation.

Name	Bytes	Version	Date
PEEK	1532	37.6	28 March 1997
POKE	1056	37.6	28 March 1997
PEEK+POKE.ASM	23917	37.6	28 March 1997

No include files are needed to re-assemble the code as all necessary constants are in the single source file.

The documentation is supplied in two forms (three if you count the assembler source) - a flat file, for people who like to print out their documentation, PEEK+POKE.TEXT, and this AmigaGuide (PEEK+POKE.GUIDE) in the fashionable heirarchical structure (actually it's still fairly flat, but prettier).

1.8 authors

AUTHORS

PEEK and POKE was written by Simon N Goodwin (simon@studio.woden.com) with help from Tadek Knapik (tadek@student.uci.agh.edu.pl). YOUR suggestions for extra features or improvements are very welcome. In fact any response at all will be gratefully received. This is not cardware or shareware, but we'd still like to know if anyone (besides us) is using it. Please get in touch.

AMIGA FOREVER - the greatest home computer ever made!

1.9 index

INDEX TO SECTIONS