

PC-Scheme/Geneva 4.02 log

Larry Bartholdi

June 1, 1999

1 Short Takes

- This document is there as a placeholder for all modifications done. As a principle items are not suppressed or modified; new entries are added in chronological order.
- The current version is PCS 4.02, dating 1993-nov-11. A Patch level number will be increased each time a bug is found and fixed.
- PCS is distributed freely in binary and source code forms. You're allowed to use it for any purposes but military.

We are working hard on documentation. . . but please be indulgent. As PCS/GE as much in common with TI's 3.03 release, we suggest you turn to Texas Instruments' documentation. You may order TI's manual at the MIT press. It's called:

PC Scheme: Users Guide and Language Reference Manual, Trade Edition, TEXAS INSTRUMENTS, **416 pp**, **1990**, ISBN 0-262-70040-9.

- The authors assume no responsibility in case of damage due to a bug in PCS. Bug hunting and fixing is nevertheless an important activity for them. If you find a bug, please send a message to the E-Mail address below. You will be rewarded by an authentic Swiss present. . .
- The official FTP server is "cui.unige.ch". PCS is there in directory "pub/pcs". Different file formats are present:
 - PCSCHEME.EXE** The big archive, containing everything except the source code. It is an archive made into an executable file for convenience purposes. The compressor used is ARJ, though you won't need it.
 - PCSLIM.EXE** An archive in executable format, too. It contains fewer files: only the executables and docs.
 - PCSCHEME.TAZ** A UNIX tar-file, compressed with gzip. It contains exactly the same files as **PCSCHEME.EXE**.
 - PCSCHEME.FIL** The list of files in the tar archive.
 - SOURCES.EXE** Another big archive, containing source code only. It is also a self-extracting ARJ file.
- PCS has its own mailbox: "schemege@cui.unige.ch". You are asked to use this address, so that developers know at a glance they receive mail concerning PCS. The developers themselves can be reached at "lbartho@cui.unige.ch" and "mvuilleu@cui.unige.ch".
- This work is supported by the University of Geneva, Switzerland.

2 What is PCS?

This program is based, and includes source code, of Texas Instrument's PC-Scheme 3.03 (1988). PCS/GE is the most performant Scheme interpreter for MS-DOS based low-cost 16 bit microcomputers from the iAPX 86,88 series; it also runs on 386/486 computers, but is not as performant as other systems which use specific enhanced hardware capabilities, even if PCS/GE needs indeed much less memory.

2.1 Copyrights

PCS/GE is protected by Texas Instrument's Copyright (see **COPYRIGHT.TXT**), but is free to distribute to everyone. Release 4.xx enhancements, corrections and translated source code are the property of the authors. BGI drivers are (c) Borland International.

3 New features (since 3.03)

- 486 Compatibility (which was not completely achieved by TI PCS 3.03)
- Text mode interface has been extended: every existing video mode (including all graphic modes, EGA 43 lines, VGA 50 lines, SVGA 132 columns...) are supported. The **TEXT-MODE** primitive allows you to set change between modes if this can be done through an unique BIOS-call; the special mode 64 puts the adapter in EGA 43 or VGA 50 lines text if available, like Borland C's **textmode** function. **TEXT-MODE** can do what **SET-VIDEO-MODE!** was doing in TI PCS.
- A block cursor is emulated when necessary; it doesn't clobber the screen and works for all video modes.
- Text attributes are compatible between graphic and text modes: the four high-order bits set the background color, the four low-order bits set the foreground color. The only difference is that MSBit means blinking in text mode and Bright Background in graphic mode.
- In order to achieve transparency between text and graph modes, **WINDOW-SAVE-CONTENTS** and **WINDOW-RESTORE-CONTENTS** have been extended to work in the same way no matter which mode you are in. That means you can for instance save the contents of the *'CONSOLE* window, switch to graphics mode and restore it on the graphic screen as if no change occurred. Note that graphic mode window-saving is done through pattern matching, and might not work if you've drawn pixels on the text. It is also not very fast.
- Edwin has been adapted to use advantages of special video modes. But due to its complex storage system, the screen dimensions are only detected at startup. That means that if you want to use another screen width after window size after having edwin already started, you should call first (**REMOVE-EDWIN**) to unload edwin from memory and let it start up again.
- You are encouraged to use the nice features of window primitives as described in your PCS reference. You might add to the window-manipulation primitives the new one coming from TI PCS 3.03: (**NEW-WINDOW** *string*) where *string* is the title of a new window to be created by the user, using arrow keys.
- PCS/GE also introduces some simplified top-level screen manipulation primitives to help you switching to mixed text & graphic modes: (**FULL-SCREEN**) makes your *'CONSOLE* window as big as possible, reserving the last line for

garbage collecting information, while (`SPLIT-SCREEN n`) makes the `'CONSOLE` window *n* lines big, at the bottom of the screen, which allows you to act interactively on the graphic screen without scrolling up what you've drawn.

- The `%GRAPHICS` primitive has been totally modified; no program should use it directly. All the TI graphic primitives have been removed, and you should use the new Borland Graphic Interface (BGI) instead; if you really need to, it won't be a problem to write a compatibility module since BGI graphics are much more sophisticated and programmer-friendly. See **BGI.TEX**.
- In addition to BGI functions, PCS/GE allows you to customize the coordinate system, linearly or not, and using as much dimensions as desired.
- The `READ` primitive, used for the Top-Level, works now like `DOSKEY`: you can recall previous lines using up and down arrows, go back, insert text and so on. (`PUSH-HISTORY item/list`) allows you to insert a string or a list into the `DOSKEY` history buffer; (`GET-HISTORY`) returns the history as a list of strings. (`CLEAR-HISTORY`) empties the history. These primitives allow you to easily save and restore the `DOSKEY` history buffer.
- `EXIT` now may receive a numerical parameter known as DOS return-code, which can be tested using `ERRORLEVEL` within a batch file, for instance.
- `OBJECT-HASH` and `OBJECT-UNHASH` now give a real hash key, based on a simple additive algorithm, rather than putting all the objects in a single linked list.
- `DOS-GETDIR` has been added for æsthetic reasons (`DOS-CHDIR` hasn't been changed)
- Stack-overflow during I/O and related exceptions don't throw you away as they did before.
- Transcendental functions (trigo & expt) are now built-in, using 80x87 if available or Borland's powerful emulation otherwise. `XLI` has been removed.
- Numerical exceptions are driven by interrupts rather than tests, which is faster and much more reliable.
- The buggy Expanded and Extended versions have been replaced by a single `PCS.EXE` which drives Expanded Memory (LIM 3 spec or later) when available. If you have only Extended Memory on a 386 or 486 computer, use `EMM386`.
- A new type of objects has been added: inline code. It allows you to use your own very short assembler-written functions as any other Scheme lambda-function. A set of macro designed for Scheme objects manipulation from assembly language is given, with documentation and examples. See **INLINE.DOC**.
- As a demo of inline-code, **PEEK.FSL** (which loads **PEEK.BIN**) implements low-level memory and bus input/output primitives. Source code is also dirtibuted (See **PEEK.S** and **PEEK.ASM**).
- `%LOGIOR` and `%LOGAND` undocumented bitwise operators on 16-bits words have been replaced by true Scheme functions, accepting a variable number of arguments, and working on integers of any size: `BITWISE-OR`, `BITWISE-AND` and `BITWISE-XOR`.

- Functions accepting a variable number of arguments such as `+`, `*` and `APPEND` can now receive any number of arguments (even more than 62) if called with `APPLY` (useful for big summations, and so on).
- Original `PCS-MACHINE-TYPE` which recognized only some Texas Computers was enhanced. It is now a list of the form `'((CPU . SPEED) NDP ROMCHKSUM)` where `CPU` is 80x86 or 8088, `NDP` is 80x87 or 0 if no coprocessor found, `SPEED` (in MHZ) is based on a memory-transfers benchmark (the most time-consuming operation for PCS/GE) and `ROMCHKSUM` can be used to recognize a computer (for example the HP95LX to setup a small `'CONSOLE` window and install the non-standard BGI driver; see `SCHEME.INI`).
- PCS can now produce directly fast-load files. Given a program `FOO.S`, just call `(FAST-SAVE-FILE "FOO.S" "FOO.FSL")` and everything's done!
- Mouse...
- ED...

4 Summary of PCS/GE's New Primitives

Some VM instructions changed significantly. Don't forget to rebuild all your own FSL modules from source.



4.1 Text Mode

`(TEXT-MODE n)`

Turns video adapter into hardware mode *n*. This is an enhanced version of the old `SET-VIDEO-MODE!`. Regular values are:

<i>n</i>	Size and Colors	Devices
0	40 × 25, Black & white	CGA and superior
1	40 × 25, Color	"
2	80 × 25, Black & white	"
3	80 × 25, Color	"
7	80 × 25, Monochrome	Hercules and Monochrome
64	80 × 43or50, Color	EGA (43 lines) and VGA (50 lines)

Any other value given in your video adapter user manual can be used. If the resulting mode is a "graphic mode" (in the sense of BIOS), a block-cursor will appear, and display will be slower, but PCS will still consider you are working with text only. If you want to draw and plot, always use `INIT-GRAPH`, `RESTORE-CRT-MODE`, `SET-GRAPH-MODE` or `CLOSE-GRAPH` to set-up your video adapter (see `BGI.TEX`).

`(SPLIT-SCREEN nlines)`

Specify how much text lines should be reserved for `'CONSOLE` window (the standard top-level I/O window). When you turn to graph mode, PCS/GE calls for you `SPLIT-SCREEN` so your drawings won't be scrolled as you type. You can use `SPLIT-SCREEN` to change the number of lines reserved for text.

`(FULL-SCREEN)`

Allocate the maximum number of lines for `'CONSOLE` window. PCS/GE calls it for you when you get back to text mode.

(WINDOW-SET-ATTRIBUTE! *window attribute value*)

This primitive is not new, but the meaning of *value* has been precised; it is not as hardware-dependent as before, and doesn't change from text to graphic mode (assuming that you can display the same number of colors).

Value is always 8 bits: | a | b | c | d | | e | f | g | h | where bits a–d code for background color, bits e–f for text color. Usually, b–d and f–h are the RGB components of the color, while a & e are intensity bits (on some adapters, intense background means blinking character).

(WINDOW-SAVE-CONTENTS *window*)

(WINDOWS-RESTORE-CONTENTS *window contents*)

Recognize these attributes and translate them from text to graphic and vice-versa.

4.2 Arithmetic

(BITWISE-OR [*n*₁ [*n*₂ [...]]])

(BITWISE-AND [*n*₁ [*n*₂ [...]]])

(BITWISE-XOR [*n*₁ [*n*₂ [...]]])

Computes the logical function on all bits of its arguments, which have to be integers.

All floating-point functions have been implemented directly, using the NDP (80x87) when present, instead of using the dirty, memory consuming XLI package. Their syntax remains the same and won't be repeated.

4.3 Hardware-level Primitives

These primitives are not fundamental to scheme and should be loaded using the command: (LOAD (%system-file-name "PEEK.FSL"))

You might play with them (I made them to show students computer architecture) but remember that programming and hacking are two very different things...

(PEEK *type address [length]*)

Reads data directly from memory.

- *type* can be either 'BYTE or 'WORD.
- *address* is a 20-bit address (for instance #xB8000 is address of the main text screen buffer for color adapters)
- *length* is optional. If not given, PEEK returns a single number. Otherwise, PEEK returns a list of contents of consecutive memory locations, starting with *address* and incrementing with size of *type*.

(POKE *type address data*)

Writes data directly into memory (use carefully !)

- *type* can be either 'BYTE or 'WORD.
- *address* is a 20-bit address.
- *data* can be either a single number or a list of numbers to be poked at consecutive memory locations, incrementing with size of *type*.

(IN-PORT *type address*)

Gets a value from an I/O port.

- *type* can be either 'BYTE or 'WORD.
- *address* is the I/O port address (usually in range #x0...#x3FF)

(OUT-PORT *type address data*)

Sends a value to an I/O port.

- *type* can be either 'BYTE or 'WORD.
- *address* is the I/O port address.
- *data* is a number.

4.4 Miscellaneous Functions

(PUSH-HISTORY *string/list*)

Push the string (or the list of strings) into the READ history buffer, as if it was the last line(s) the user just typed; it has no direct effect on the program, but allows the user to recall the string(s) just by pressing up-arrow and modify them (DOSKEY-like interface).

(GET-HISTORY)

Returns the list of last lines typed during READs. In conjunction with PUSH-HISTORY, this allows a program to save and then restore the top-level DOSKEY context.

When you use the (EXIT) primitive to quit Scheme, without giving an ERRORLEVEL parameter or with ERRORLEVEL = 0, a file containing this information named **HISTORY.INI** is always created in the default directory. It contains already a call to PUSH-HISTORY, so you just need to insert the command (load "HISTORY.INI") in your **SCHEME.INI** and the PCS DOSKEY environment will be restored.

(CLEAR-HISTORY)

(DOS-GETDIR)

Equivalent to (DOS-CHDIR ".") but visually better.

(FAST-SAVE-FILE *source [dest]*)

Makes a fast-loadable file from a scheme source file. If *dest* is not given, the source name is taken with an ".FSL" extension instead of the existing one. Note: you don't need COMPILE-FILE any more...

(FAST-SAVE '(PCS-CODE-BLOCK ...) [*dest*])

Assuming the given PCS-CODE-BLOCK is correct (as are those given by COMPILE), FAST-SAVE outputs the fast-load form of the code-block to port *dest* or to the default output-port if no *dest* is given. This is the low-level primitive used by FAST-SAVE-FILE, as COMPILE is the low-level primitive used by COMPILE-FILE.

(CLOCK)

Returns the number of ticks since the start of . A tick is about 55 milliseconds ($1/CLOCK - TICK$). The purpose of `CLOCK` is to measure time deltas. The time in seconds could be determined with:

```
(let ((start (clock)))
  (do-my computation)
  (/ (- (clock) start) CLOCK-TICK))
```

5 Things That Didn't Change For Now

The average speed of the interpreter didn't change significantly; The change from Small to Medium model and the cleaning of some fast-but-ugly tricks are compensated by the more efficient compiler and some optimizations we made on essential VM instructions and arithmetic.

R^4 RS is not yet achieved.



6 Features Removed

- XLI support was removed because it was memory-consuming, and not very performant. PCS/GE object code and makefile will soon be available on the same server; interfacing can be done very easily through the `%ESC` opcode (all arguments are automatically converted to C objects).
- Protected mode Scheme was removed (use SCM if you really need something that doesn't run in real 8086 mode).
- `MAKE_FSL.EXE` doesn't exist any more. It is now directly implemented in PCS by the new `FAST-SAVE-FILE` function, which is much more convenient. There are also some weird things happening with `MAKE_FSL` and large or floating-point numbers.

7 A Look on the Future

- Integrated text windowed environment (including text-editor, environment editor)
- True R4RS `'()` handling, Multiple values
- Hardware Interrupts Scheme procedures (complement to engines)
- Music Driver as example of Interrupt procedure
- Integration of the Scheme Utilities written by Clyde Camps (TI)

8 Ideas and Questions

Which ones of the suggestions in previous section you think are the more important, interesting and/or urgent? Please give us your Hit-Parade...

There is a very complete Virtual Machine Low-Level Debugger, not documented and intentionally not included the distributed version, since it was developed for VM elaboration only. But if you consider it from a theoretical point of view, in association with the auto-compiler written in scheme, it might be a quite good support for a course on Compiler Systems. Ask us for it if you are interested.

9 Bug fixes

Here bug fixes are described in reverse correction order (that is, the most recent corrections come first).

1993-oct-24 Fixed bug with `port?`: when called with a symbol, returns garbage.

Fiddled with directory structure, ready for distribution.

1993-oct-13 Added negative bases to `integer->string`. Created new opcodes for `divide` and `modulo`. Fixed a problem with `read-atom` when symbols and strings of more than 256 characters are read.

1993-oct-4 Changed (`runtime`) so that it returns ticks, not hundredths.

1993-oct-2 Fixed a bug with `remove-edwin` (symbols not removed). Set up plans for a desktop environment. Fixed some problems with `Brief`: `^A` yielding multiple completions, `TAB` indenting the line even if the cursor is not at the start, and a bug in the indentation algorithm: “`(test_`” followed by `CR` indents at +6.

1993-sep-30 Added `MOUSE` package with primitives.

1993-sep-28 Added symbolice constants to `set-file-position`. Fixed a bug with files greater than 64K.

1993-sep-28 Added string support to `get/set-file-position`.

1993-sep-28 Fixed bug with `dos-file-size`: `zopen` did not set all 32 bits.

1993-sep-27 Fixed (`length 1`) so that it accepts only proper lists.

1993-sep-27 Modified `Window-reverse-text!` so that it yields the complementary color in fore- and back-ground.

1993-sep-27 Fixed a bug with `bitwise-and`, traced to `read.s:alloc,nt : leading0's should be suppressed when allo`

1993-sep-11 ~~1993-sep-11~~ **Fast-load** crashes when too many nested files are loaded. The exception is now correctly handled, and the limit has been increased to 8.