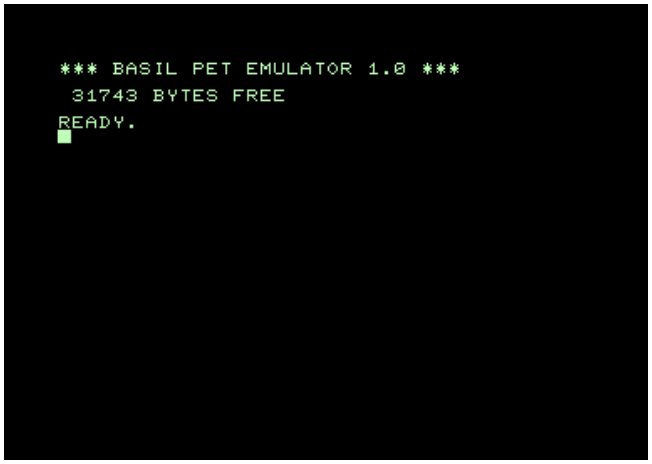
BASIL PET EMULATOR

Version 1.0

Basil is a PET Emulator for Commodore 64, inspired by Jim Butterfield's One-Line PET Emulator shown in Robin Harbron's 8-Bit Show And Tell



Basil emulates CB2 sound, both PET character sets, various keyboard models, shift key register, joystick, jiffy clock, screen retrace bit, disables color control codes and remaps the tape buffer. Basil runs patched C64 KERNAL ROM and patched copy C64 BASIC, remapping many zeropage POKE's and PEEK's to corresponding C64 locations (such as cursor location, key buffer, memory pointers, USR function address).

Basil even adds that one extra space before each line number in LIST, just like PET does. First half of PET BASIC ROM is included as a memory image, because many programs check bytes in memory range $c000-$c3ff to detect PET ROM version.

Historical games currently working include Bonzo, Android Nim, Bat, Bets, Blackjack, Checkers, Dungeon, Everest, Lander, Ambush, Capture, Racer, Lawn, Tank, Ouranos, Rescue, Star Wars Train, Pinball, Frogs, Space, Afo, Alligator 2, Jackpot, Debris, Ski, Volcano, Millipede, Tron Journey, Meteorites, Canyon, Demon, Nab and many more. Currently, Space Ace and Microchess 3.0 almost work, but crash after a while or when doing certain moves.

EMULATOR FUNCTION KEYS

F1	Show one-line help
F2	Reset PET
F3	Keyboard (151) mode
F4	Shift (152) emulation on/off
F5	Key repeat on/off
F6	Change text color (light green, green, amber, light gray, white)
F7	Sound on/off
F8	Tape buffer patch on/off

PATCHED POKES & PEEKS (BASIC)

PET	C64	Description
216	214	Cursor line number
198	211	Cursor column
158	198	Number of chars in keyboard buffer
205	212	Quote mode
40-53	43-56	BASIC memory pointers
0-2	784-786	USR() function jmp and address

How does it work? In Basil, POKE0,96 actually writes 96 to 784. Similarly, PEEK(0) reads from 784. Magic! Obviously, this translation only works in BASIC programs.

TAPE BUFFER PATCH (BASIC)

PET has two tape buffers, the first starting at $027a. It's a common place for short machine code subroutines, such as the Cursor Magazine joystick and keyboard reading routine. This area clashes with many C64 KERNAL screen editor variables.

Basil captures any POKE and PEEK statements accessing area $0200-$02ff and changes the address to $0300-$03ff. SYS to $0200-$02ff triggers a relocate routine that scans the code at $0340-$03ff modifying any 16-bit absolute, indexed or immediate addressing instructions and jumps there.

The relocation is also triggered when writing the MSB of USR() instruction to location 2 (mapped to 786 by Basil). Relocate code also changes any JMP $d278 and JMP $d26d instructions to the C64 equivalent JMP $b391 (convert integer to floating point) - a common and reasonable way to exit an USR() function.

JOYSTICK EMULATION (IRQ)

PET	Description
59471	Bit 0 up, bit 1 down, bit 2 left, bit 3 right. Fire: up+down

Joystick in port 2 is translated to PET joystick directions at $e84f (59471) every 5 ms. Bit value goes zero when a joystick direction is active. Fire button zeroes both bits 0 and 1 (thus making it impossible to detect simultaneous up/down + fire).

There's the high nybble for another joystick port, but emulating joystick port 1 is not implemented, because I didn't find any two player games for testing this. Cursor Magazine joystick reading subroutine combines (ands) input from both nybbles.

KEYBOARD EMULATION (IRQ)

PET	C64	Description
151	197	Keyboard input
152	653	Shift key (bit 0)

C64 keypresses from 197 are translated to PET keys in location 151 using one of the four available keyboard modes. Press F3 to change mode:

GRAPHICS	Graphics keyboard
BUSINESS/NUMPAD	Business keyboard, numbers mapped to PET numeric pad
BUSINESS/TOP ROW #	Business keyboard, numbers mapped to PET top row
CRTC (ASCII)	PET with CRTC chip, partially implemented

Note: This selection doesn't have any effect when reading keys using GET or INPUT statements in BASIC, or when using (C64) KERNAL routines or the normal screen editor. The mapping only makes a difference when the PET program is reading keyboard directly from location 151. One such program is the game Bat, which requires GRAPHICS keyboard mode (the default) to work.

CRTC (ASCII) mode is only partially implemented. For now, it only knows the unshifted keys. Shifted keys set bit 7, which works fine for shifted cursor keys, but not for other keys, some of which I believe should return a totally different ASCII character when shifted. CRTC chip is included in all 80 column PET's, but also in some later 40 column models. In which? I don't know. I didn't find any software for testing this.

Known issue: Shift key from bit 0 in C64 register 653 is reflected to the equivalent PET location 152. This clashes with the C64 memory location for

number of files open, so if your PET program accesses multiple files, you may need to turn off Shift emulation by pressing F4. Again, this setting only makes a difference when reading the state of shift directly from location 152, not when using GET, INPUT, (C64) KERNAL or the screen editor.

CHARACTER SET SWITCHING (IRQ)

PET	Description
59468	Bit 1 selects uppercase & graphics or upper- & lowercase set

POKE59468,12 or PRINTCHR$(142) to display uppercase + graphics
POKE59468,14 or PRINTCHR$(14) to uppercase + lowercase

To conserve memory, only the non-reversed chars are stored in memory and the reversed characters are generated when switching sets. Thus, there is a brief 50 ms pause in emulation when switching.

Note: You're likely to find some older PET programs that have uppercase and lowercase letters swapped. Those programs were written on the early PET ROM revision, where uppercase and lowercase were swapped in char ROM. An option to emulate this may be added in a future version of Basil.

SOUND EMULATION (IRQ)

PET	Description
59467	Set bit 4 to enable sound (looping shift register mode)
59464	Sound frequency
59466	Shift register pattern (1-bit waveform)

PET documentation suggests using bit patterns %00001111 (15), %00110011 (51) and %01010101 (85). Each defines a 50% pulse wave on different octaves.

Basil tries to imitate any bit pattern to the extent what is playable on a single SID voice, so coarse pulse width modulation is possible, just like on the real PET. For example, bit patterns 1, 3, 7, 15, 31, 65, 127 produce SID pulse widths $02, $04, $06, $08, $0a, $0c, $0e on the lowest octave. Also the same bit pattern rolled to any of its 8 positions gives the same octave and pulse width. Pulse width can be also modified on the second octave, but at half the accuracy ($04, $08, $0c). Highest octave is always square ($08).

Non-SID compatible bit patterns (such as %11011011) play a compromised tone on the second octave with pulse width of 2 x (number of 1-bits). Notes that are too high for SID (fundamental frequency around 4000 Hz) are muted.

SCREEN RETRACE BIT (IRQ)

PET	Description
59456	Bit 5 screen retrace: 1 = 'vblank', 0 = 'drawing screen'

Entire C64 KERNAL is copied to RAM and the memory area $e700-$e8ff is freed for an readable/writeable image of PET hardware registers. Corresponding KERNAL code is moved to $c700-$c7ff and any accesses elsewhere in KERNAL are patched pointing there. This enabled reading the joystick directions from $e84f and polling the screen retrace bit in $e840 (bit 5).

Basil IRQ is not currently synchronized to screen refresh (like it is on PET), but it's running on the regular KERNAL 60 Hz CIA timer divided by 4. Keyboard scan is done every fourth call.

There seems to be some differences what screen retrace actually means depending on PET model (whether it has a CRTC chip or not). I couldn't verify whether a zero value always means the raster scan is outside visible screen lines or does it in older models actually indicate the hardware vertical blanking period.

Basil approximation is that $e840 / 59456 bit 5 is set for 25% of the time and cleared for 75% of the time. This is somewhat consistent to whatever I had in my current VICE XPET emulator settings.

TIMERS AS RANDOM SOURCE (IRQ)

PET	Description
59460	Timer 1 LO
59461	Timer 2 HI
59465	Timer 2 HI

Timers are not emulated, but because many programs potentially use timers as a source of (somewhat) random numbers, Basil fills these three timer registers with (somewhat) random numbers every 5 ms.

For example, without this feature the Meteorites game dropped constant meteorites falling in one direction only.

(Joystick and sound emulation are also running at 5 ms.)

JIFFY CLOCK EMULATION (IRQ)

C64 jiffy clock - a human-readable 3-byte software timer of 1/60 seconds, seconds and minutes - is mirrored to PET jiffy clock locations $8d-$8f.

Since PET IRQ is synchronized to screen refresh instead of a cycle-accurate timer, there's another counter for jiffy clock correction at $99-$9a. On PET this counter counts 1/60th seconds up to 623, then resets to zero and skips one 1/60th second in $a0, for the purpose of keeping the clock in time.

Some software such as Microchess 3.0 use the jiffy clock correction counter as a 'stop timer', zeroing location $99 and waiting for the value there reach >128, producing a ~2 second wait. Basil enables this type of short delay loops by incrementing the value in $99 every frame. High byte $9a is not emulated.

FUTURE EXPLORATION

- Chain PET interrupts to C64 IRQ handler
- Move entire C64 IRQ to NMI to enable some PET programs that do SEI
- Patch KERNAL ROM and move C64 "number of files open" to another location
- Option to block unknown POKE's for less crashes in untested BASIC programs
- Study programs that almost but not quite work yet and see what classic or new machine code games could be supported with reasonable effort

IMAGE CREDITS

PET photo, Wikimedia Commons CC BY 2.0 Michael Dunn
PET logo remake, CC-BY-SA Dj152
Basil photo, Wikimedia Commons CC BY 3.0 Frank Huber

APPENDIX: CB2 SOUND

HOW DO I MAKE SOUND ON MY PET?

This process sets the PET's shift register in a free-running state where the signal is used for sound generation. By adjusting the pattern of the output and the frequency you can produce a wide variety of sounds, and even music!

Three pokes are required to make sound:
POKE 59467,16 (turn on port for sound output use 0 to turn it off*)
POKE 59466,octave (octave number, see below)
POKE 59464,frequency (0 for no sound)

After setting 59467 you can adjust 59466 and 59464 to get any sort of sound, but to get music you need to set them with specific values, here is a three-octave note table:

Note Table:

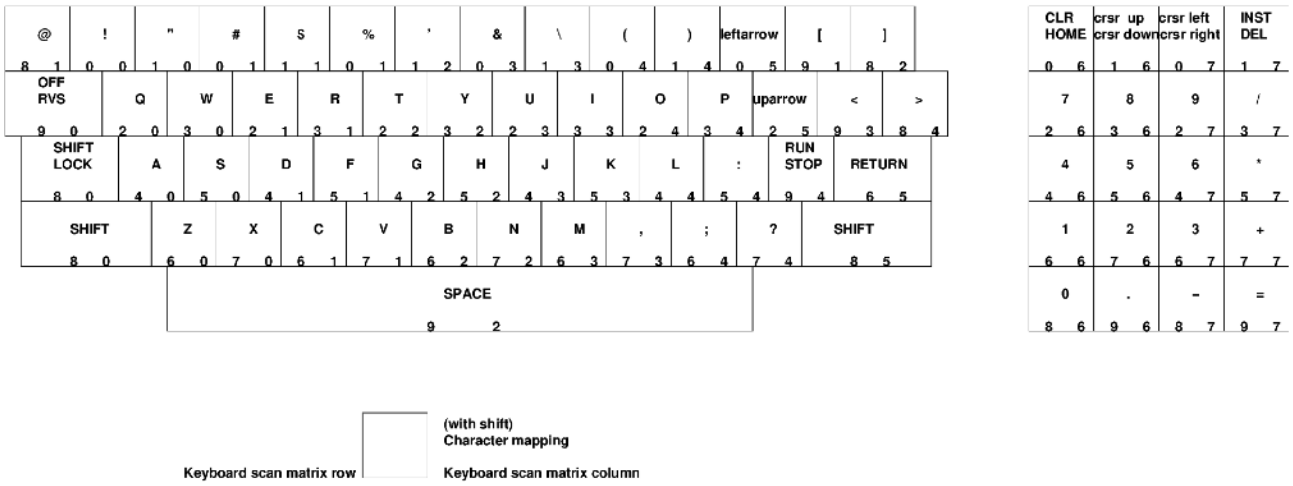
	octave=15		octave=51		octave=85	
Note Freq	Oct.0	Oct.1	Oct.1	Oct.2	Oct.2	Oct.3
B	251	125	251	125	251	125
C	238	118	238	118	238	118
C#	224	110	224	110	224	110
D	210	104	210	104	210	104
D#	199	99	199	99	199	99
E	188	93	188	93	188	93
F	177	88	177	88	177	88
F#	168	83	168	83	168	83
G	158	78	158	78	158	78
G#	149	74	149	74	149	74
A	140	69	140	69	140	69
A#	133	65	133	65	133	65

Set 59466 with octave range desired and play notes by setting the frequency in 59464. To stop any sound use POKE 59464,0.

* Note, due to a hardware bug, leaving the shift register in free running mode will cause problems when attempting to use the datasette so always POKE 59467,0 before attempting to use any tape commands.

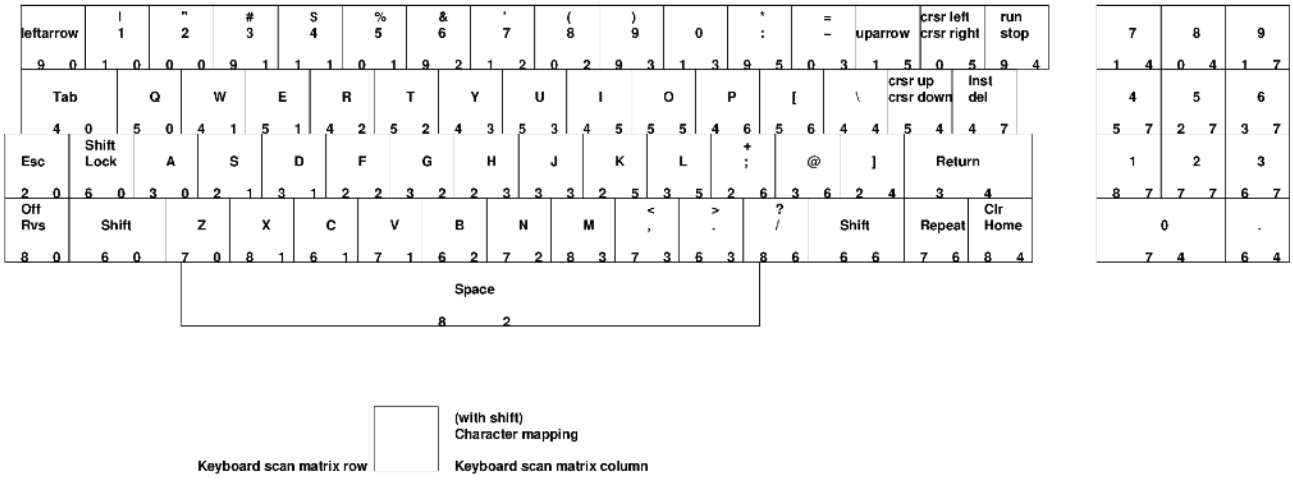
APPENDIX: PET KEYBOARDS

PET graphics keyboard



PET business keyboard

United Kindom / Europe / 50Hz



Source: http://www.6502.org/users/andre/petindex/keyboards.html