Programming In Forth on the Vectrex – Phillip Eaton 2018

What is a Vectrex?

https://youtu.be/k8GiErP6Nfc

Circuit board

- CPU: Motorola 68A09 @ 1.5 MHz
- RAM: 1 KB (two 4-bit 2114 chips)
- ROM: 8 KB (one 8-bit 2363 chip)
- Cartridge ROM: 32 KB
- MOS 6522 Versatile Interface Adapter (VIA)

Sound

- Sound: General Instrument AY-3-8912
- 3-inch electrodynamic paper cone speaker

Design

European release Vectrex playing the built-in game Minestorm, without overlay

My Background

- Spent 90s programming Z80 SBCs with MPE Forth for SCADA applications
- Collected a lot of classic video arcade games: Space Invaders, Asteroids, Defender
- Spent 2000's in London and Zurich on financial systems
- 2 years ago, acquired a dead Vectrex and fixed it

What can I do with it?

- Vibrant home brew community, some amazing programs, hardware hacking
- Memory map and cartridge port simple and open
- I could put CamelForth onto the bare metal 😀
- Challenges: no serial port, don’t know 6809 assembler, don’t know Vectrex BIOS, don’t know low-level Forth

Define Goals

- Get Forth running on Vectrex with interactive terminal
- No Vectrex hardware modification allowed (can’t swap out the BIOS)
- Must provide Forth API to the BIOS
- Must be comparatively fast compared with assembler and C, not a toy
- Must be accessible to potential new developers

Step 1

- Configure CamelForth For Vectrex and cross compile
- No DOSBox – convert cross compiler from F83 to.... Gforth
- No block source files, need to tweak parser – took a lot of thinking about!

Setting up camel forth memory map

README.09 - Notepad2

File Edit View Settings ?

```
the on-board I/O to the space 7C00-7FFF hex.  As distributec
only uses RAM locations 6000-7BFF hex, allowing the use of e
(6264), 32K (62256), or 128K (628128) static RAM.

        0000 +--------------------+
             |       unused       |
             |                    |
        6000 +--------------------+
             |Forth RAM dictionary|
             |                    |
        7980 +--------------------+
             |        TIB         |
        7A00 +--------------------+
             |     user area      |
        7A80 +--------------------+
             |  parameter stack   |
        7B00 +--------------------+
             |   HOLD,PAD areas   |
        7B80 +--------------------+
             |    return stack    |
        7C00 +--------------------+
             |    on-board I/O    |
        8000 +====================+
             |       unused       |
             |                    |
        E000 +--------------------+
             |    Forth kernel    |
             |                    |
        FFF0 +--------------------+
             | 6809 reset vectors |
        FFFF +--------------------+

This memory map is controlled by screen 61 of the CHROMIUM.S

        HEX 0E000 FFFF DICTIONARY ROM   ROM     ( A )
            7A00  EQU UP-INIT                   ( B )
            7A    EQU UP-INIT-HI                ( C )
            6000  EQU DP-INIT                   ( D )

Line A specifies the lower and upper ROM limits.  Thus, to u
32K ROM, you would change '0E000' to '8000'.  Do not change
```

Ln 1 : 193 Col 1 Sel 0 | 9.09 KB | ANSI | CR+LF | INS | Default Text

https://roadsidethought | Grant's 6809 computer | Phillip

Secure | https://roadsidethoughts.com/vectrex/vectrex-memory-map.htm

Memory Map

The 68A09 can address memory between $0000 to $FFFF (65Kb).

The ports of the 6522 programmable interface adapter (PIA) have been mapped into the memory space beginning at $D000. A PIA port is selected by the setting of address bits 3 to 0 (the least significant address byte). The setting of address bits 11 to 4 (the two middle bytes) are ignored. As a result, $D12F, $D6AF or $DFFF is the sam as $D00F - only bits 15 to 11 (which must be to a hex D) and bits 3 to 0 matter.

The Vectrex memory has been mapped as follows:

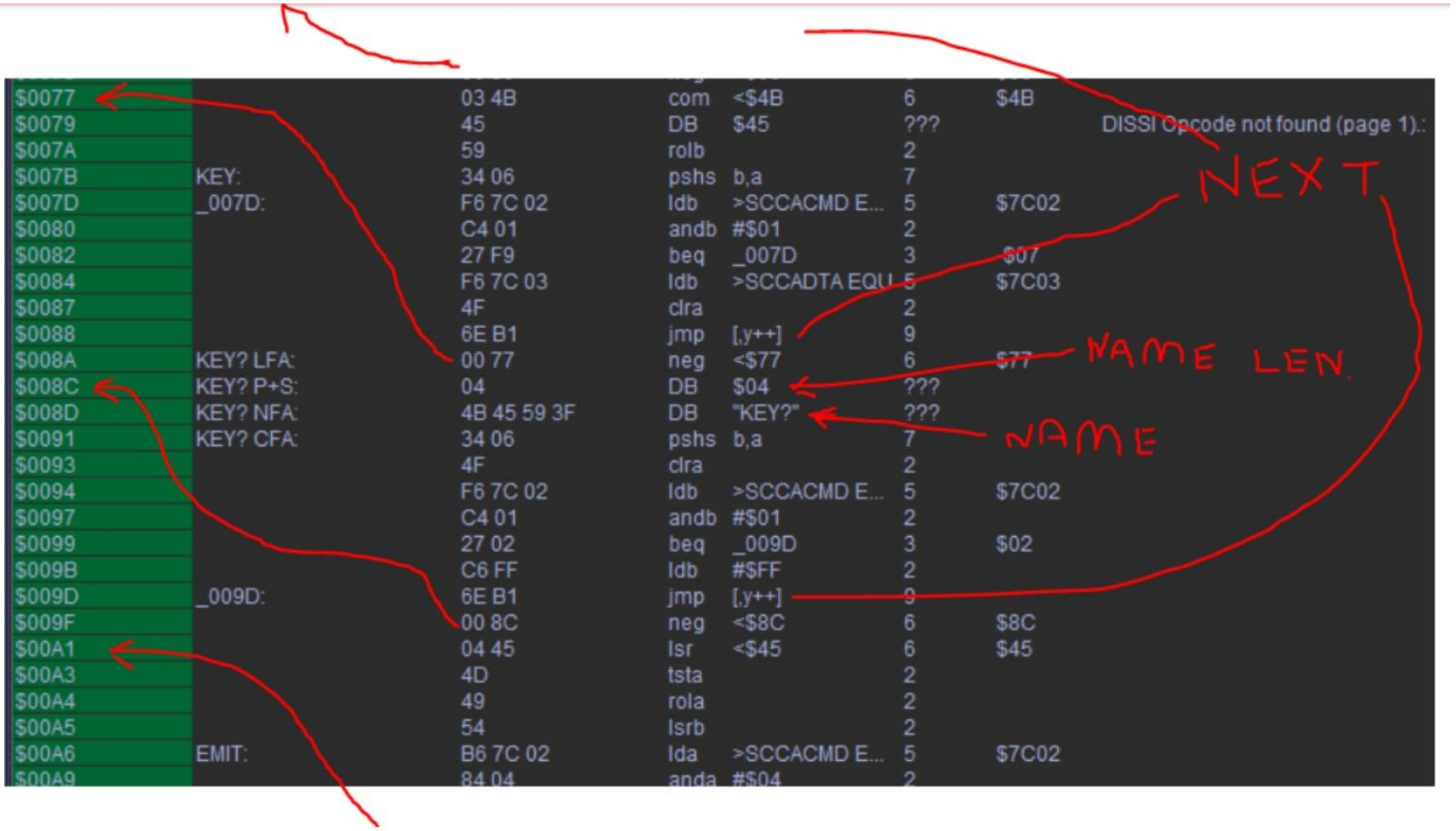
Address Range	Description
$0000 - $7FFF (32kb)	Reserved for game ROM (read only)
$C800 - $CBFF (1kb)	Static RAM (read or write)
$C800 - $C87F (128b)	Reserved for RUM
$C880 - $CBFF (896b)	Reserved for game logic
$Dxx0 - $DxxF (16b)	The Programmable Interface Adapter (read or write)
$E000 - $EFFF (4kb)	Reserved for Mine Storm (read only)
$F000 - $FFFF (4kb)	Reserved for the RUM (read only)

The 68A09 has reserved the highest 16 bytes of memory for its various vectors:

Address Range	Description

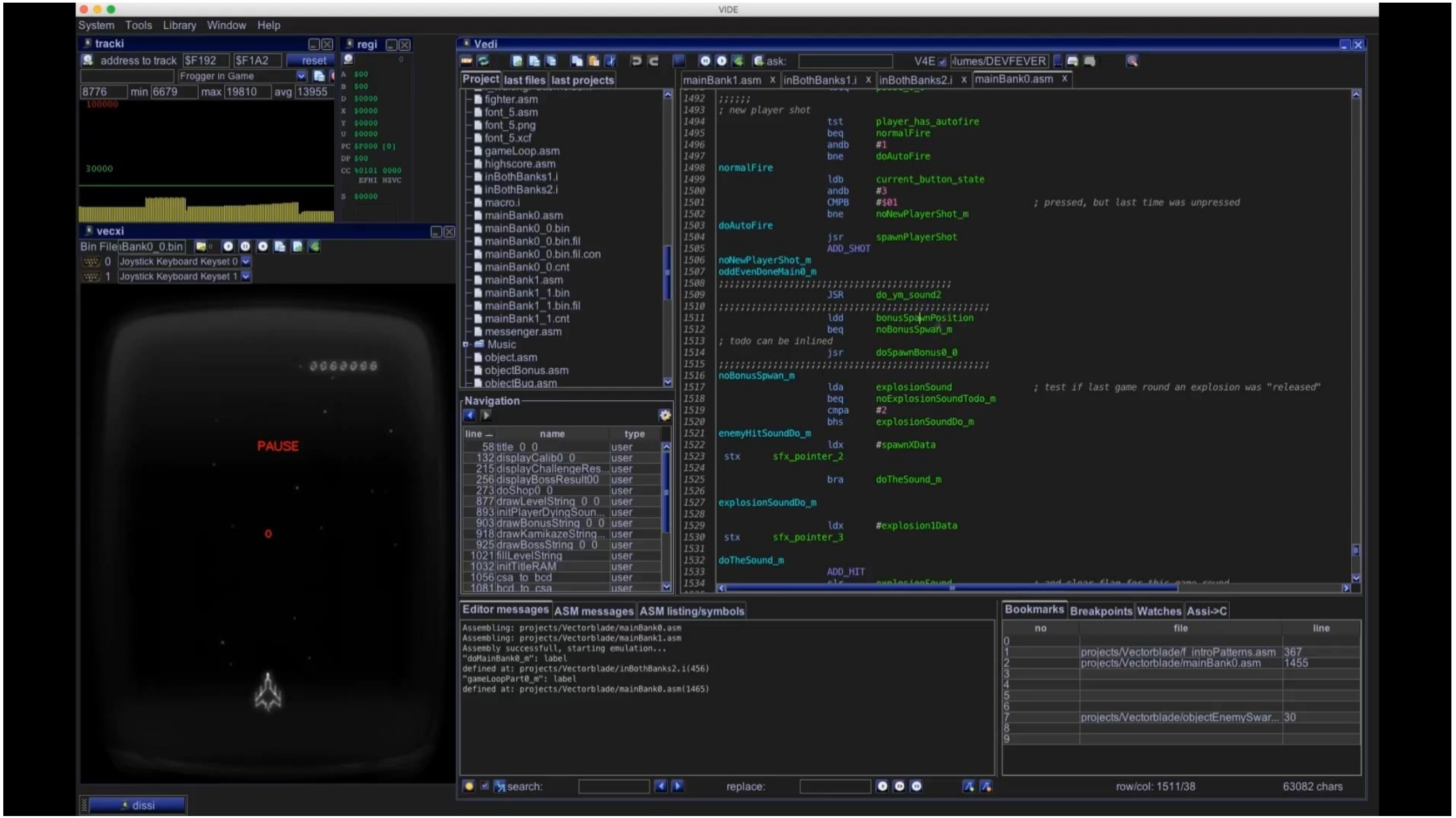
Step 2

- Debug in VIDE emulator
  - Create label file for debugger
  - Use Starting Forth to learn how code is compiled
  - Will it clash with BIOS?
  - Hack COLD to write to display via BIOS



https://youtu.be/t4lwoWBXPhA

Vectrex IDE


VIDE
System Tools Library Window Help
tracki
address to track $F192 $F1A2 reset
Frogger in Game
8776 min 6679 max 19810 avg 13955
regi
vecxi
Bin File mainBank0_0.bin
0 Joystick Keyboard Keyset 0
1 Joystick Keyboard Keyset 1
PAUSE
dissi
Vedi
Project last files last projects
fighter.asm
font_5.asm
font_5.png
font_5.xcf
gameLoop.asm
highscore.asm
inBothBanks1.i
inBothBanks2.i
macro.i
mainBank0.asm
mainBank0_0.bin
mainBank0_0.bin.fil
mainBank0_0.bin.fil.con
mainBank0_0.cnt
mainBank1.asm
mainBank1_1.bin
mainBank1_1.bin.fil
mainBank1_1.cnt
messenger.asm
Music
object.asm
objectBonus.asm
objectBug.asm
Navigation
line name type
58 title_0_0 user
132 displayCalib0_0 user
215 displayChallengeRes... user
256 displayBossResult00 user
273 doShop0_0 user
877 drawLevelString_0_0 user
893 initPlayerDyingSoun... user
903 drawBonusString_0_0 user
918 drawKamikazeString... user
925 drawBossString_0_0 user
1021 fillLevelString user
1032 initTitleRAM user
1056 csa_to_bcd user
1081 bcd_to_csa user
mainBank1.asm
inBothBanks1.i
inBothBanks2.i
mainBank0.asm
1492 ;;;;;;
1493 ; new player shot
1494 tst player_has_autofire
1495 beq normalFire
1496 andb #1
1497 bne doAutoFire
1498 normalFire
1499 ldb current_button_state
1500 andb #3
1501 CMPB #$01 ; pressed, but last time was unpressed
1502 bne noNewPlayerShot_m
1503 doAutoFire
1504 jsr spawnPlayerShot
1505 ADD_SHOT
1506 noNewPlayerShot_m
1507 oddEvenDoneMain0_m
1508 ;;
1509 JSR do_ym_sound2
1510 ;;
1511 ldd bonusSpawnPosition
1512 beq noBonusSpwan_m
1513 ; todo can be inlined
1514 jsr doSpawnBonus0_0
1515 ;;
1516 noBonusSpwan_m
1517 lda explosionSound ; test if last game round an explosion was "released"
1518 beq noExplosionSoundTodo_m
1519 cmpa #2
1520 bhs explosionSoundDo_m
1521 enemyHitSoundDo_m
1522 ldx #spawnXData
1523 stx sfx_pointer_2
1524
1525 bra doTheSound_m
1526
1527 explosionSoundDo_m
1528
1529 ldx #explosion1Data
1530 stx sfx_pointer_3
1531
1532 doTheSound_m
1533 ADD_HIT
1534
Editor messages ASM messages ASM listing/symbols
Assembling: projects/Vectorblade/mainBank0.asm
Assembling: projects/Vectorblade/mainBank1.asm
Assembly successfull, starting emulation...
"doMainBank0_m": label
defined at: projects/Vectorblade/inBothBanks2.i(456)
"gameLoopPart0_m": label
defined at: projects/Vectorblade/mainBank0.asm(1465)
Bookmarks Breakpoints Watches Assi->C
no file line
1 projects/Vectorblade/f_introPatterns.asm 367
2 projects/Vectorblade/mainBank0.asm 1455
7 projects/Vectorblade/objectEnemySwar... 30
search: replace:
row/col: 1511/38
63082 chars

Step 3

- No serial port. Time to get hands dirty now...enter VecFever
- Rewrote EMIT, KEY?, KEY for soft UART
- Unhack COLD
- Try it out...

https://youtu.be/FhHfR9zPggg

```
285  : BZ \ -- ;
286  \  2 RND +! \ New seed for Random
287    INIT
288    0 BOMBY C!
289    BEGIN
290      0 9F \ FF
291      DO \ y axis
292        FF 0
293        DO \ x axis
294          CR ." Stk:" .S  ."  T2-Hi:" D009 C@ U.
295          _Wait_Recal _Intensity_5F
296          -7F -7F _Moveto_d_7F
297          7F 20 CITYVL _Draw_VL_ab
298          \
299          _Reset0Ref
300          I 80 - FF AND J 80 - FF AND _Moveto_d_7F
301          20 4 PLANE _Draw_VL_ab
302          \
303          BOMBY C@ 0 =
```

Game main loop – not optimized or factored!

Forth interface to Vectrex BIOS – no optimization!

```
154 CODE _Intensity_7F \ -- ;
155       8 # ( DP) PSHU,             \ -- ; Save DP
156       D0 # LDX,    X DPR TFR,     \ -- ; DP to D0
157       6 # ( D) PSHS,    Intensity_7F JSR,    6 # ( D) PULS,
158       8 # ( DP) PULU,             \ -- ; Restore DP
159      NEXT ;C
160
161 CODE _Print_Str_d \ x y c-addr -- ; Print single string to screen
162      8 # ( DP) PSHU,             \ -- x y c-addr ; Save DP
163      D0 # LDX,    X DPR TFR,     \ -- x y c-addr ; DP to D0
164      D U EXG,                    \ -- x y U-addr ; String addr to U, save U to D
165      S 2 , LDX,    S 2 , STD,    \                 ; Stack -ROT (2 lines)
166      S 0, LDD,    S 0, STX,      \ -- U-addr x y ;
167      A B EXG,    S ,++ ADDD,     \ -- U-addr yx  ; Combine x and y
168      Print_Str_d JSR,            \ Call Vectrex BIOS subroutine
169      6 # ( D) PULS,              \ -- U-addr     ; Drop TOS
170      D U TFR,    6 # ( D) PULS, \ -- ; Restore U, drop TOS
171      8 # ( DP) PULU,             \ -- ; Restore DP
172      NEXT ;C
173
```

Other little videos

- City Bomber – the basics of a game

https://youtu.be/wbV4a56reNA

- Interactive test to discover what BIOS Wait_Recal function does

https://youtu.be/yWUVZyadA0w