https://wildbitscomputing.com/

VCF East 2026 - Classroom

Intro to the Wildbits Jr2 plus SuperBASIC with character graphics

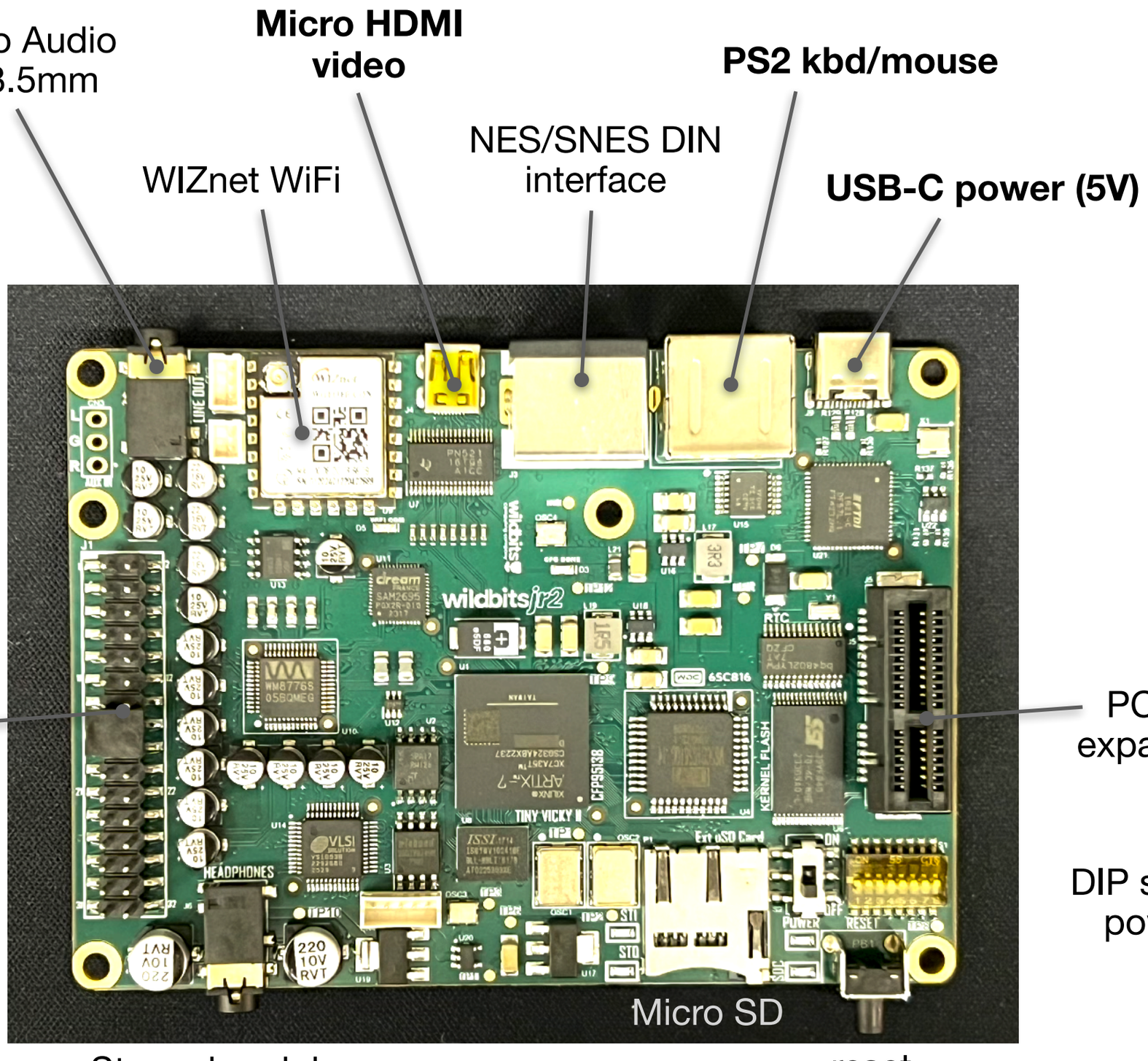
Content developed by
Michael Weitman - @EMwhite

Introducing the Wildbits Jr2

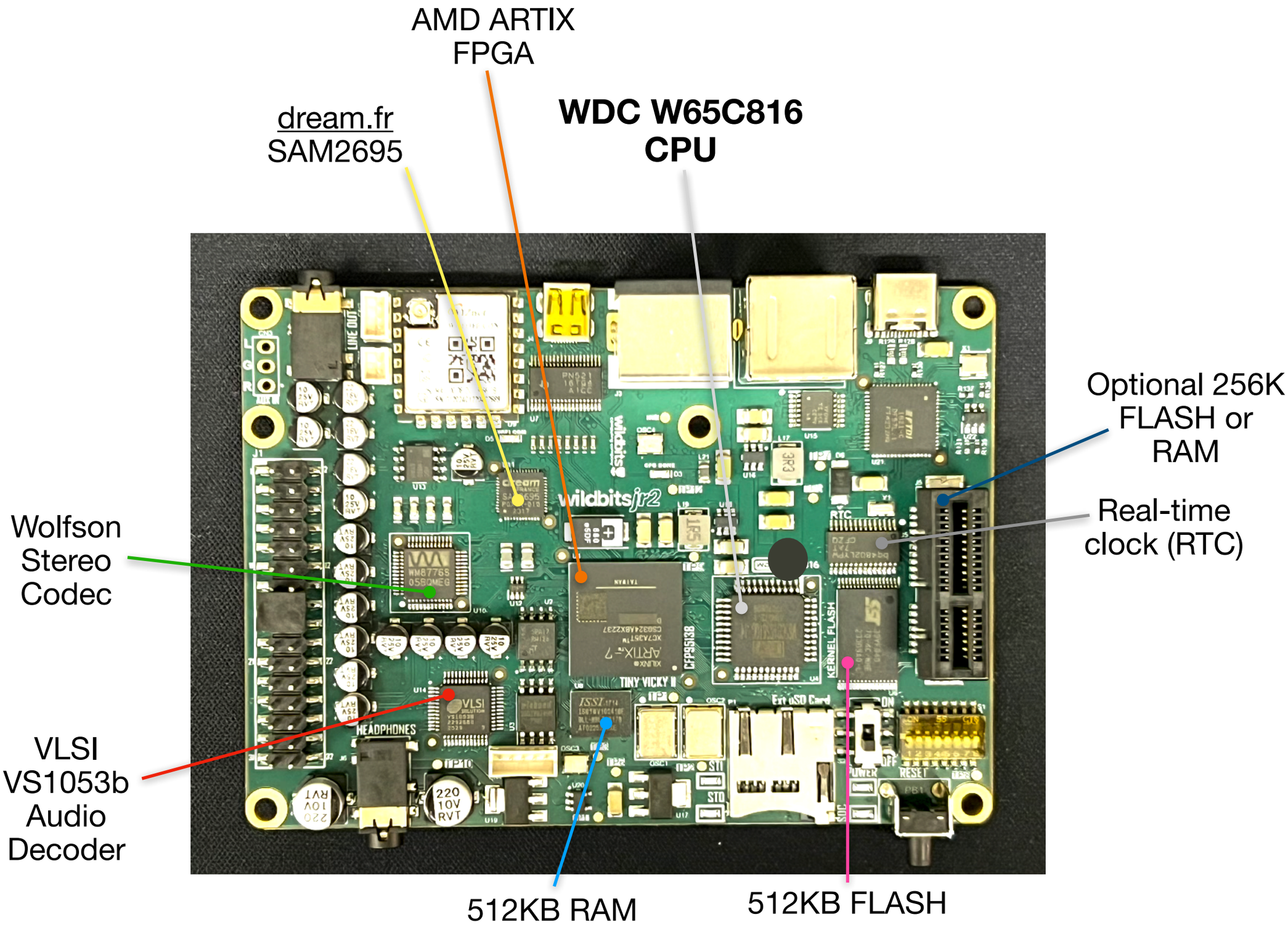
The Wildbits Jr2 is:

- An **easy-to-use** and powerful low-cost home computer with astounding graphic, audio, and computational capabilities
- Ideal for vintage enthusiasts, developers wishing to port code from classic **6502 based** platforms, and most of all, a fun environment to learn how to program or explore as a tool for creative outlet
- Runs the same code as the Wildbits K2 (keyboard based); feature-for-**feature parity**, aside from physical ports & case features
- Based on a WDC 65C816 microprocessor, **a real, physical CPU** that you can see and touch. (And it is 100% 6502 compatible; in fact, it starts up in 6502 mode just like its Apple II, ATARI 8-bit, BBC Micro, Commodore, and Ohio Scientific ancestors)
- Supported by a **thriving community** of hobbyists and developers

JR2 Anatomy (connections)



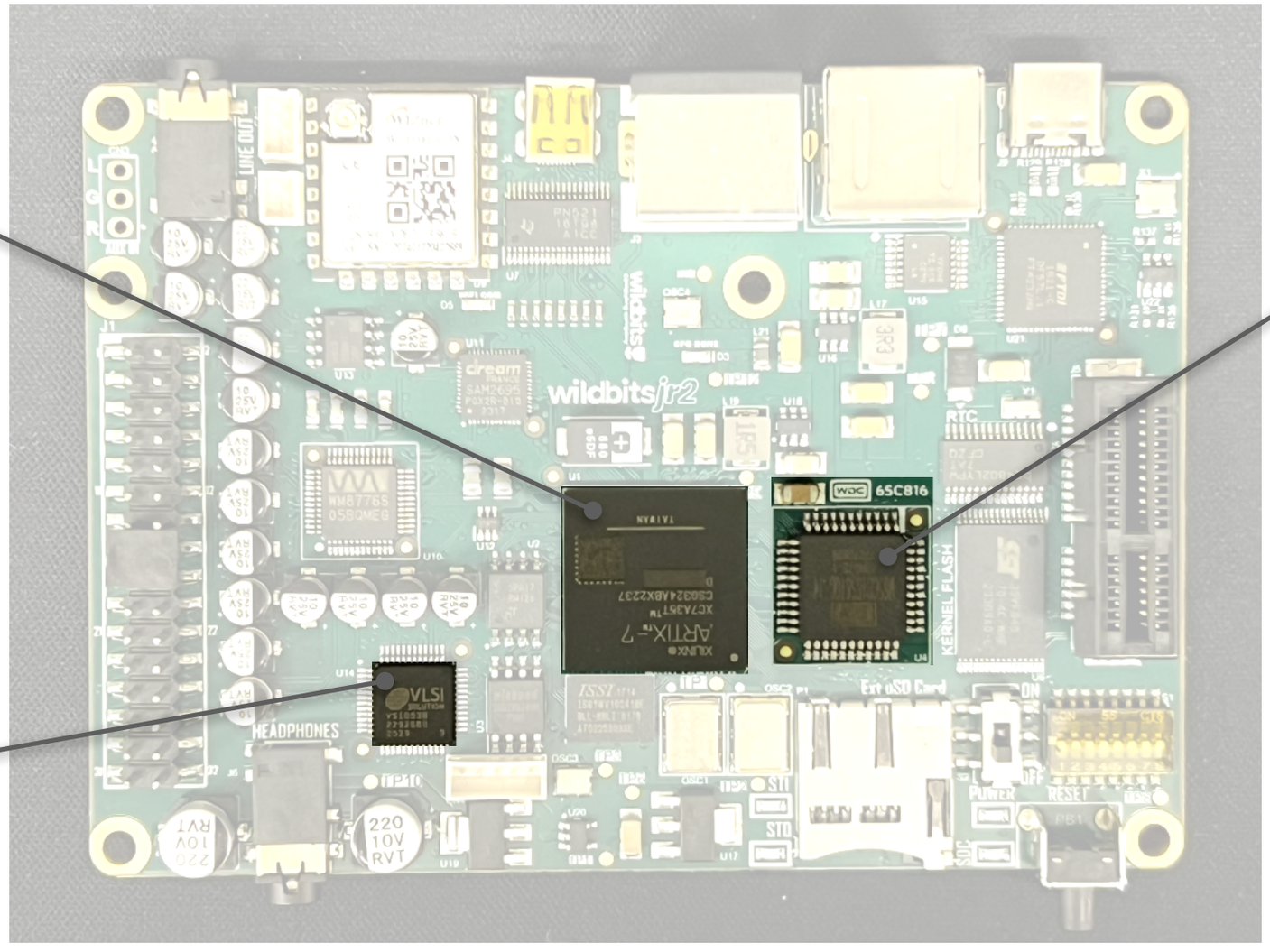
JR2 Anatomy (components)



JR2 Anatomy (3 processors)

Are they really processors? Processing domains is a better description.

ARTIC 7 FPGA
- Video
- Audio
- Timers
- Math co-processor
- MMU
- Acceleration

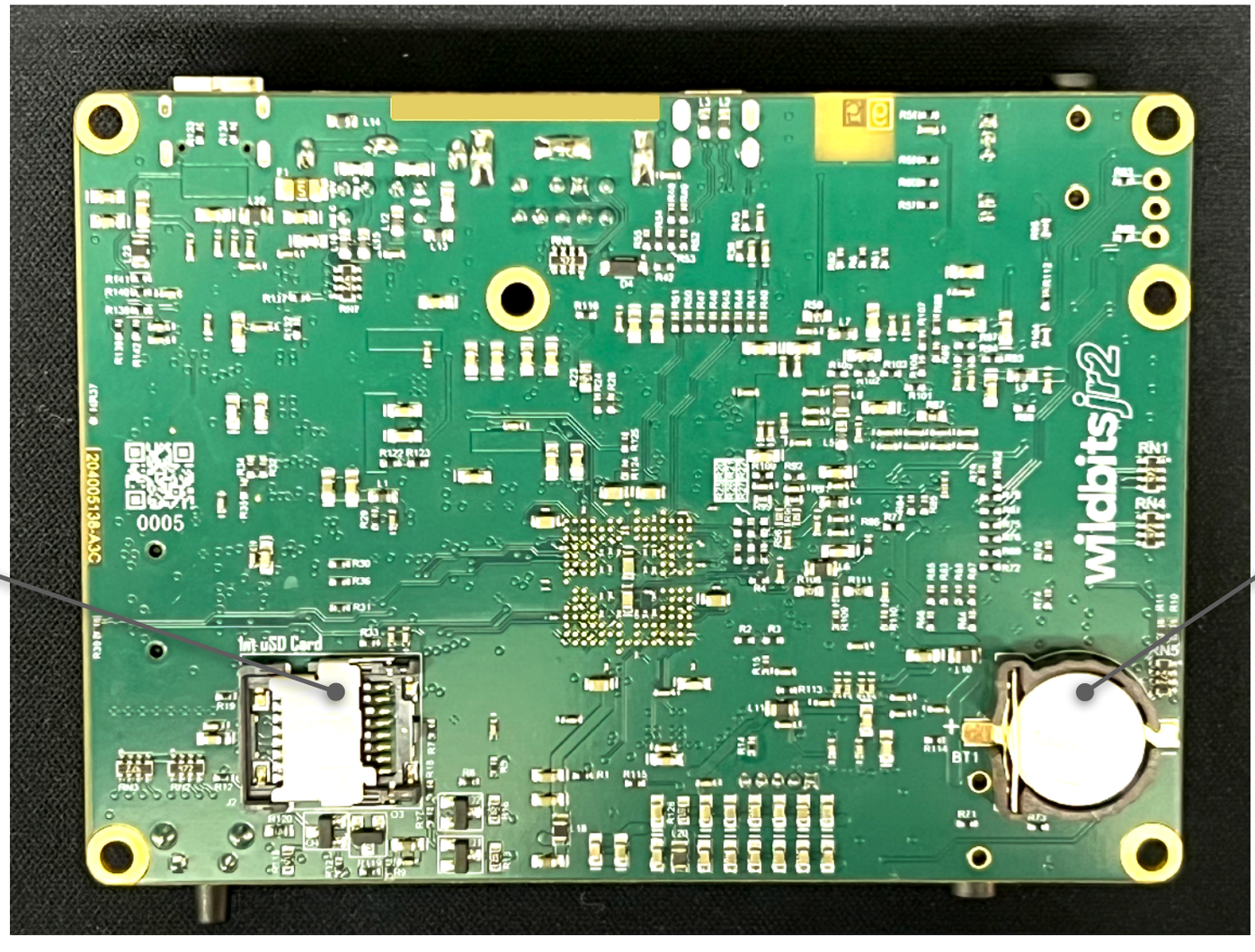


WDC 65816 CPU
- kernel
- BASIC interpreter
- your programs

VLSI VS1053b
- Media decode
- DSP pipeline processing

JR2 Anatomy (back view)

Not much to see here, except for increasingly smaller and smaller passive components. Steve Jobs would say “impossibly small”.



JR2 Anatomy - FPGA

What is it?

- Software-defined hardware (HDL)
- A 'chip*' with hundreds of connections
- Astonishing performance and capabilities

What's Inside?

- Glue logic (otherwise know as memory bus logic and clock domain arbitration)
- Video 'card'
  - Character sets and graphic engines
- Audio 'card'
  - SN76489
  - Two Gideon SID instances
  - Yamaha OPL3

Future Expansion

- Features & Bug fixes
- Alternate CPU and OS

* Less of a 'chip' in the traditional sense; it is a modern 324-pin BGA (ball-grid-array) component

Screen Editor

<u>Five things to know</u>:

1. **It is your primary interface to the computer**
2. **You may freely navigate in all directions, but your commands will not be evaluated until you press the ENTER key, sometimes referred to as the RETURN key**
3. **Programs may be:**
   - **written, listed, executed**
   - **saved, verified, loaded**
4. **You can list the contents of the SD card without disturbing the program in memory, using** `dir`
5. **There are several *special* keystrokes (but it is not necessary that you know or use them all). You should, however, memorize these two:**

`ctrl-c`	to break a program or listing or dir
`shift key (alone)`	to pause a listing (program or dir)

BASIC Introduction

<u>Three things to know about the history of the BASIC language</u>:

1. In 1964, DARTMOUTH University first released the BASIC Programming Language. It is not called 'BASIC' because it is *basic*. It is the:

   <u>B</u>eginners <u>A</u>ll-purpose <u>S</u>ymbolic <u>I</u>nstruction <u>C</u>ode

   The original version had 15 keywords and 10 math functions. Shortly after release, the `INPUT` keyword was added. This opened the door for interactive programs.

2. MICROSOFT nearly went out of business in the mid-1970s before licensing Microsoft BASIC to Commodore (for a flat fee of $25,000).
3. Within a few years, home computer wars led to manufacturers creating extensions to the language. SuperBASIC has well over 100 keywords and operators. It is rich, mature, and continues to evolve.

<u>Two artifacts to review</u>:

Original vintage documentation from DARTMOUTH (<u>click here</u>)

A 40-minute video recapping the genesis of DARTMOUTH BASIC (<u>click here</u>)

Welcome to SuperBASIC

- SuperBASIC is an evolution of the interpreter that shipped in ROM on the famous Acorn BBC Micro in 1981
- Structurally, it *feels* like Microsoft BASIC, but is modern, with support for Wildbits hardware
- Is deemed '**Super**' because of system extensions which support proprietary features such as sound, graphics, tiles, sprites, memory management, and more
- The SuperBASIC environment is simultaneously an interactive workspace where expressions can be entered to perform scratch calculations or examine files present on storage devices, *and*, a screen editor for entering and modifying BASIC programs
- Navigation is similar to *early* commercial home computers such as the ATARI 8-bit systems and the Commodore PET/VIC-20/64. However, you will notice that the size of the screen is much larger and you will see that the editor is more powerful
- As of version 1.2, SuperBASIC supports programs at least as large as 128K with a string and array heap and graphic objects and data managed elsewhere

Exercise #1: Load a full-sized program into memory for examination

```
load "toolkit/font.bas"
```

↖ Subdirectories are supported, but there is no `cd`, `chdir`, or `pwd` command; instead, we use fully qualified filenames wrapped in double-quotes

SuperBASIC: 20 keywords to know

> 120 keywords and functions in total !!

- `cls`
- `dir`
- `list` [*from*, *to*] (or by *procname*)
- `load` *filename*
- `run`
- `bload` *filename*, *addr*

Immediate mode operational commands

- `print` [`at` *y*, *x*]
- `for` and `next`
- `chr$` (*value, var, or iter*)
- `int()`
- `rnd()`
- `peek`
- `poke`
- `read` and `data`
- `goto`
- `gosub` and `return`

[mostly] standard Microsoft BASIC keywords

- `cprint` [`at` *y*, *x*]
- `random()`
- `proc` and `endproc`
- `sound` (also `zap`, `shoot`, `ping`, `explode`)

Wildbits SuperBASIC extensions

Listing and Printing

Exercise #2: Listing a program is accomplished by typing the keyword `list` which optionally supports several different forms beginning with the `{from}, {to}` form.

`list 20,100`	<press RETURN>	List a range of lines, inclusive
`list ,200`	<press RETURN>	List from the lowest numbered line to 200
`list 120,`	<press RETURN>	List starting with line 120 through the last line

Exercise #3: Printing text strings using simple formatting. Try a) *implied* new-line; b) tab stop and c) *suspended* new-line forms. You should type “new” before embarking on this exercise.

```
10 print "Hello Wildbits!"
20 goto 10
```

If you type `run` and press the RETURN key, you’ll encounter an endless loop and will need to either type CTRL-C or press RUN/STOP (if you are using a K2) to break the running program.

```
10 print "Hello Wildbits!",
20 goto 10
```

A trailing comma will move the cursor to the next ‘tab-stop’

```
10 print "Hello Wildbits! - ";
20 goto 10
```

A trailing semi-colon will ‘suspend’ the normal new-line behavior

Advanced Listing and Printing

Exercise #4: Listing a procedure* can be accomplished by typing the word `list` followed by a valid (defined) procedure name followed by empty parens, as follows:

`list loadfont()` <press RETURN> List just the 17 line `loadfont` proc

Exercise #5: List scroll-back* is a new feature that lets you pull back text that has scrolled off the screen. It can also do the opposite, scrolling upwards from the bottom of the screen.

Try it: HOLD the START key (or '*F*' or 'WLD' key) while pressing CURSOR UP (or DOWN)

Exercise #6: Printing cursor control or color commands uses the standard `print` keyword along with a `chr$(`*value*`)`. In the case of foreground and background colored text, characters printed will be rendered with the most recently printed color codes. See the appendix page for a handful of cursor commands and a color chart.

```
print chr$(128);chr$(147);"Hello Wildbits!"
```

Exercise #7: Printing at a specific screen location can be accomplished with the `at` directive to `print` or `cprint` as follows:

```
print at 10, 20;"The next thing is true. The last thing was false"
```

* To make the most out of these exercises, reload the "`toolkit/font.bas`" program

SuperBASIC: Graphic chars

Exercise #8: Printing graphic characters with `cprint`

Your Wildbits computer uses an enhanced version of the ASCII character set and includes dozens of graphic characters or 'glyphs'. Each is unique, and can be referenced by number ranging from 1 to 255. The BASIC keyword modifier `chr$(x)` allows you to print otherwise untypeable characters by referring to them by value. Here are some:

card suit chars

To print these, we use the `cprint` keyword, *not* the normal `print` keyword:

```
10 for x = 1 to 255:cprint chr$(x);:next:print
```

note the `'c'`

In examining the character set, you might notice some patterns:

- groupings of horizontal and vertical lines
- 'shaded' or dithered checkerboard patterns
- quarter block characters and various arrows and bullet shaped glyphs

Below, we will make this output more useful and later, we will redefine a few chars for a fun diversion.

Let's prefix the Wildbits glyphs with their numeric values and improve the formatting. Enter and `run` the following, but mind the spacing and punctuation on line 20:

```
10 for x = 1 to 255
20   cprint x;": ";chr$(x),;
30 next
40 print
```

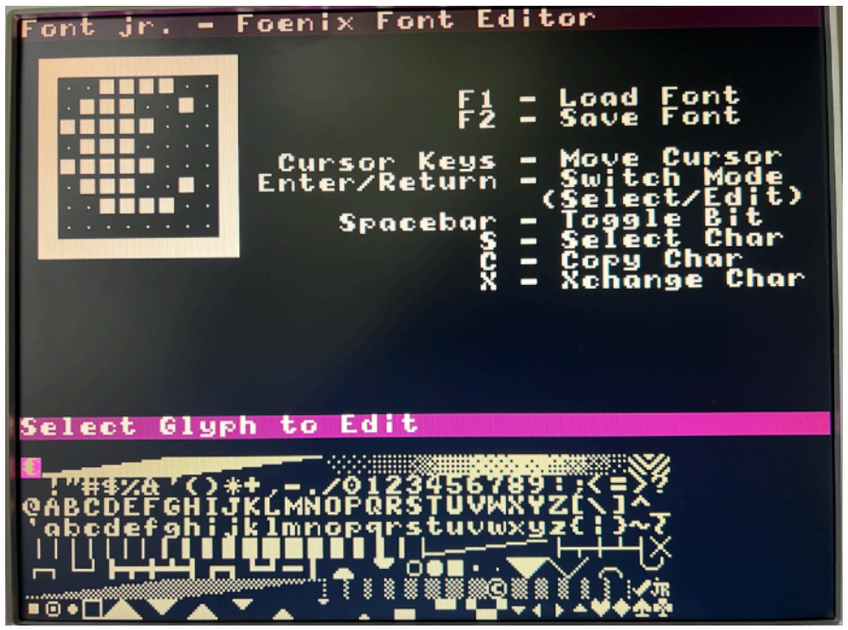
a multi-line, slightly enhanced version of the single-line example program is easier to read. Examine the differences (they are subtle). The output is improved with little additional code.

Wildbits Font Editor

Font Editor is part of the Wildbits Graphics Toolkit. You can load it / run it from either “/FM” (followed by `xgo`) or by typing the following in SuperBASIC:

`load "toolkit.bas"` <press RETURN> and after loading is complete, `run` the program

Press “3” to load the Font Editor:



Exercise #9: Have fun with the Wildbits Font Editor. You might notice that after you quit the program, or if you press the reset button, changes you made will persists. You can examine the new custom character set by typing in the example on the prior page.

To reset the font to the default set, power-off your Jr2, count to 5, then turn it back on again.

Demo program: Invaded.bas

Exercise #10: Have fun running and modifying a SuperBASIC program

`invaded.bas` is a short program that you can use to practice some of what we have learned. It uses redefined characters, `chr$(x)`, for screen foreground and background color, `peek` and `poke` commands to manipulate screen memory and to animate movement of an alien from the left side of the screen to the right side, and uses sound effects.

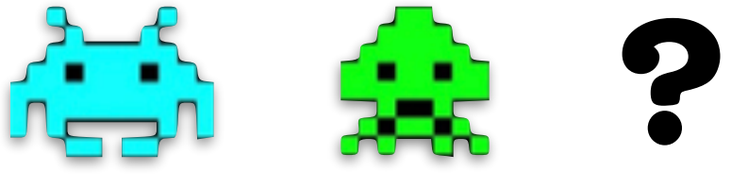
`load "invaded.bas"` <press RETURN> and after loading is complete, `run` the program

Be sure to plug in earpods or a powered speaker so you can hear the audio.

After running, try your hand at modifying the code (change the color, alter the range of motion, or modify the sound effects).

If you are feeling adventurous, try redefining the characters to change the graphic object.

The best way to learn is by experimenting, breaking things, and powering through to find a solution!



Appendix

(Reference data to make the prior examples more meaningful)

Selecting a color

Above, we mentioned there are `chr$` codes to control cursor movement, clear screen, and perform other functions, such as color. Each character may have one foreground color and one background color and there are 16 to choose from occupying `chr$` codes 128 to 160. Here are the highlights:

- Foreground color codes begin at a value of 128; there are 16 foreground colors to choose from (see chart below). You may choose to redefine your own color palette; check the reference guide for additional detail.
- Background color codes begin with a `chr$` value of 144; there are 16 dedicated background colors, as well. While vintage computers had a single background color for the entire screen (sometimes exercising trickery for two), the K2 can be thought of as having a choice of color for each of the 4,800 character positions on the screen (80 x 60).
- To select a green foreground color, we add a value of 3 to 128 to `print chr$(131)`.
- In this example (and in our program), everything printed after the keyword will be green until we change it again.
- To make the background black, we add a value of 0 to the base of 144 to `print chr$(144)`.

Here is the full default color chart. By default, the background and foreground colors are identical, but you can modify this yourself by poking values into the proper memory map location:

0	black	8	dark gray
1	medium gray	9	light gray
2	blue	10	purple
3	green	11	light green
4	magenta	12	pink
5	brown	13	red
6	orange	14	yellow
7	light blue	15	white

A few versions of the famous Commodore “10 PRINT”, single-line BASIC programs (adapted to the K2 / Jr2); try them!

```
10 ch=int(186.5+rnd(1)):cprint chr$(ch);: goto 10
10 ch=int(164+random(5)):cprint chr$(ch);: goto 10
```

A few directional control characters and text examples:

key	val	action	key	val	action
`ctrl-l`	12	clear screen	`ctrl-a`	1	start of line
`ctrl-b`	2	cursor left	`ctrl-e`	5	end of line
`ctrl-f`	6	cursor right	`ctrl-k`	11	kill to EOL
`ctrl-n`	14	cursor down	F3	131	green text
`ctrl-p`	16	cursor up	F4	132	lurid text

The full Wildbits character set in a 16 x 16 grid, including a 15-line SuperBASIC program to print out this table:

```
10  for x = 0 to 15
20    if x < 10
30      if x = 0
40        print x;"   ";
50      else
60        print x;"  ";
70      endif
80    else
90      print x;" ";
100   endif
110   for y = 0 to 15
120     cprint chr$(x*16+y);" ";
130   next
140   print
150   print
160 next
```