Tiny Pascal for the 1802.

John A. Stewart
Ottawa, Canada.

V1.2 9 January 2024
V1.1 6 November 2024
V1.0 8 Sept 2024

Introduction

This Implementation consists of a compiler, and an interpreter.

The compiler compiles "Tiny Pascal" source and produces an intermediate p-code, which is run by the interpreter.

Background

Back circa 1980, I took Wirths' Algorithms + Data Structure book's "PL/0" compiler and interpreter, added code to make the compiler useful, and took the interpreter and moved it to the Intel 8088-based IBM PC. Eventually, this interpreter made it into the "OS" of an IC-tester, for a silicon foundry producing small-batch customer-specific ICs. Everything but my copy of Wirths' book has been lost, or left behind when I changed companies.

Fast Forward to 2024, I still have Wirths' book, complete some 45 year-old notes I pencilled in, and decided revisit the project and target it to the 1802 as a first step.

Current Status

9 January 2025:

The TinyPascal compiler compiles using FreePascal, on OSX and Linux. I do not have Windows, but it should compile just fine.

The TinyPascal compiler takes in a "TinyPascal" program, and produces "PCode" in an A18 1802 Assembler format to run through the PCode interpreter.

The PCode interpreter is written in A18 Assembler, and I run it on MacOS in the Emma-02 emulator, and on the MemberSHIP/MemberCHIP boards produced by Lee Hart.

The PCode interpreter is set to mimic Lee Hart's Membership/Chip cards, with ROM at 0000 and RAM at 8000. There are assembler conditionals in the code to easily allow for code-relocation. I generally just run it located at 0x8000.

The TinyPascal compiler produces A18 assembler code; with conditionals and defines for where the code resides, again, relocatable. Currently, it loads at 0x8800.

An example program or two is provided.

First I load the interpreter to 0X8000, then I load my assembled p-code "TinyPascal" compiled program to 0X8800.

The reason for loading the p-code interpreter first is that, it contains a very small program that does nothing (so, no garbage printed) at 0x8800. Loading YOUR TinyPascal program at 0x8800 overwrites this small do-nothing program.

In the MCANSA monitor provided with Lee Hart's cards, simply type "R8000" after everything is loaded, and your program will run.

Features, or not.

I have added a few Pascal features, and more are planned.

There are a few changes, or missing elements, from standard Pascal:

- "uint16" and "char" variables - right now, no signed integers, sets nor arrays.
- procedures and functions with parameters work.
  - reading the assigned value to a function in the function currently is not allowed. Treat it as a write-only variable.
  - parameters are passed by value, not currently possible to pass by location.
- peek/poke - read/write to absolute memory locations.
- Character input not implemented yet.
- Typing. Hmmm. This is kind of working, but I ran into some gotchas. So, you can assign a uint16 to a character. This is great for "write" statements, as a "char" prints a character, and "uint16" prints a number.
- Numbers. Hmmm.
  - Hexadecimal numbers are allowed. Uses "C" convention, eg, in the maze program, the maze is stored at "0xA000".
  - Right now, numbers are printed in 4-digit HEX format. Yes, HEX format.
- Stack Checking. Right now, if you have a highly recursive program, stack will grow (stack is at high mem) downwards, and eventually overwrite your pcode, and things will crash. Should implement stack checking...

To compile and run the PCode Interpreter:

Some specific notes, intended for follow-along for both you and I.

Step 1) Assemble the interpreter, that runs natively on the 1802, from the "TinyPascalInterp.asm" source.

On a terminal on MacOS, I have a little script called "interp.txt". Here is the contents:

```
/Users/john/Desktop/CDP1802-Desktop/a18/a18 TinyPascalInterp.asm -l
TinyPascalInterp.lst -o TinyPascalInterp.hex
```

The output, "TinyPascalInterp.hex" is generic for all TinyPascal PCode programs; it is what is loaded into EMMA-02 or onto the membership card to run your TinyPascal programs on the 1802.

Step 2) To compile a TinyPascal program, I currently use Linux. It needs FreePascal, which was an easy install on Linux. Check out freepascal.org for a package for your OS.

On Linux in a terminal window:

- install freepascal.
- write a TinyPascal program.
- execute TinyPascal by:

```
./TinyPascal < myprog.pas
```

to convert the TinyPascal program you wrote into p-code which is placed in a ".asm" file. (currently, "**assemblerOut.asm**" in the directory you invoked TinyPascal from)

- take the resulting "assemblerOut.asm" file back to macos. Rename it if you wish.

On a MacOS terminal window:

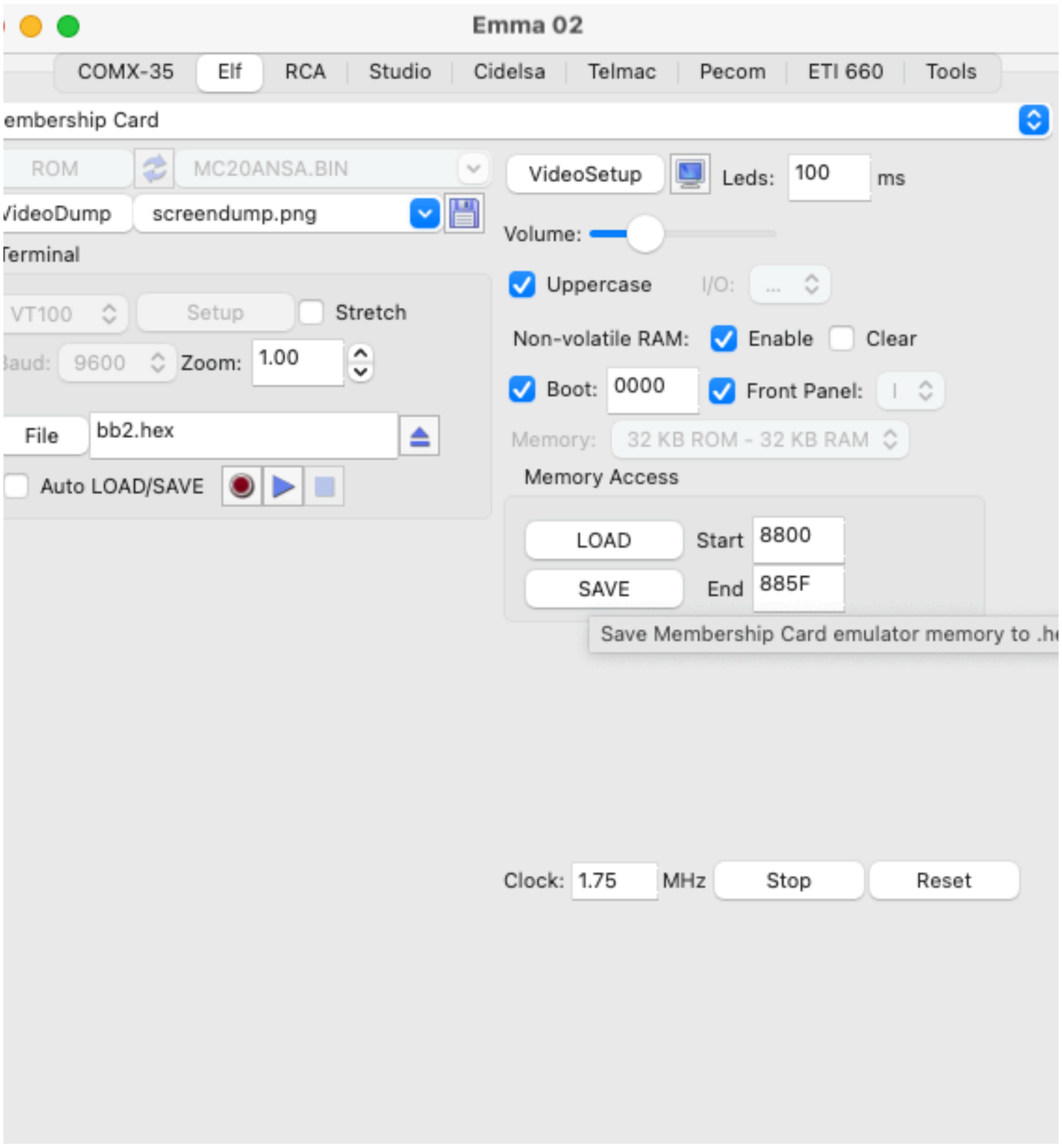
- take the "assemblerOut.asm" and run it through A18
  (as you did with the pcode interpreter above). eg:

```
/Users/john/Desktop/CDP1802-Desktop/a18/a18 maze2.asm -l maze2.lst -o
maze2.hex
```

As mentioned above, the Interpreter loads at 0x8000 in the 1802 memory space, and the pascal p-code assembly file (assemblerOut.asm assembled into assemblerOut.hex) at 0x8800. (These are all able to be changed by defines; these work for me on the MemberCHIP card.)

On Macos, start up Emma-02:

- select ELF as the computer to emulate.
- load in the "PL0Interpreter.hex" file.
- load in the "assemblerOut.hex" file.
- in the Emma "1802 console" window, type R8000 and the program will run.


Emma 02

COMX-35 | Elf | RCA | Studio | Cidelsa | Telmac | Pecom | ETI 660 | Tools

embership Card

ROM: MC20ANSA.BIN

VideoDump: screendump.png

Terminal: VT100, Baud: 9600, Zoom: 1.00

File: bb2.hex

Leds: 100 ms

Uppercase ✓

Non-volatile RAM: Enable ✓ Clear

Boot: ✓ 0000

Front Panel: ✓

Memory: 32 KB ROM - 32 KB RAM

Memory Access

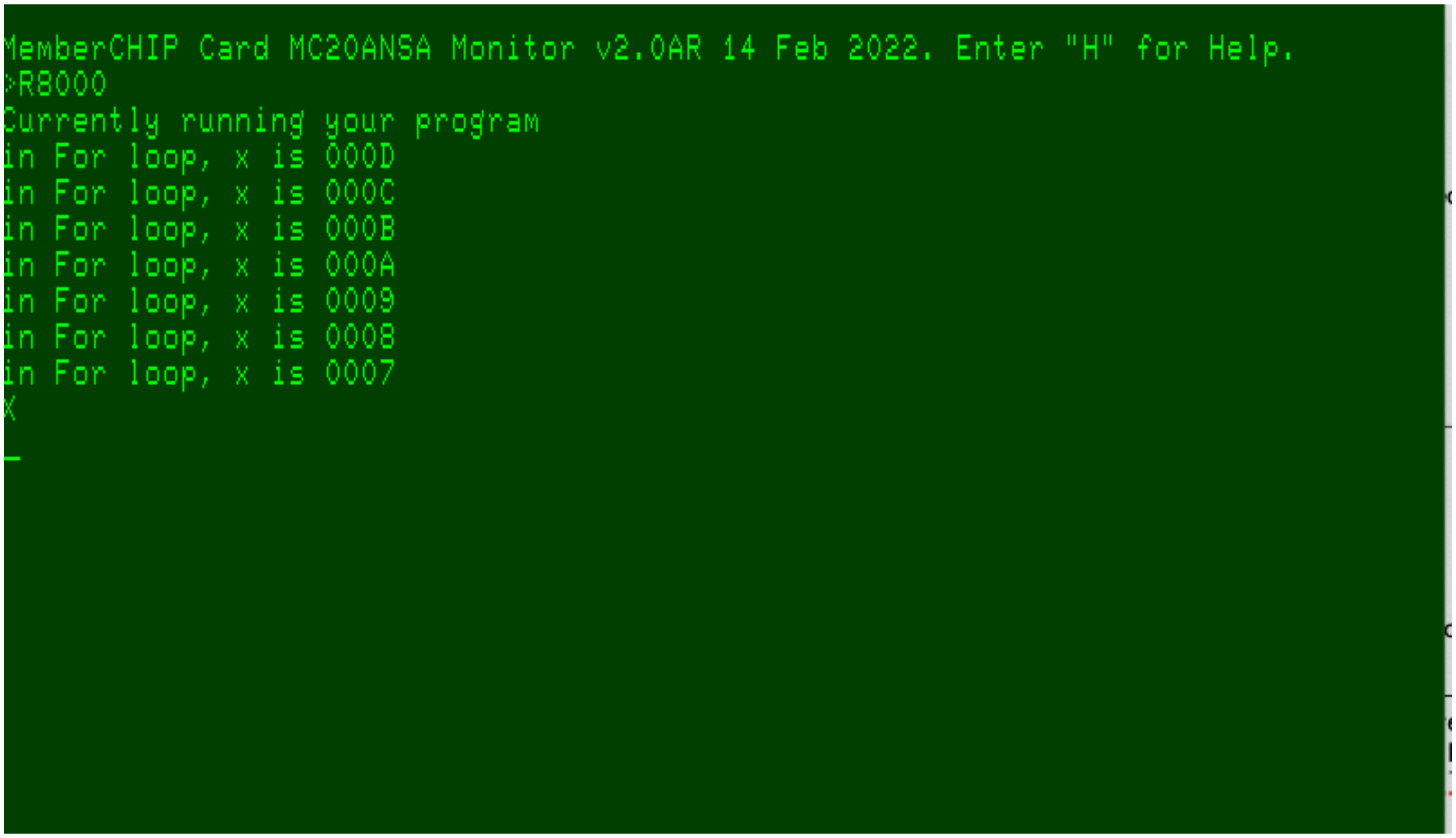
LOAD Start 8800

SAVE End 885F

Save Membership Card emulator memory to .he

Clock: 1.75 MHz

Loading the interpreter at 8000, then (shown) the tinypascal prog at 8800



```
MemberCHIP Card MC20ANSA Monitor v2.0AR 14 Feb 2022. Enter "H" for Help.
>R8000
Currently running your program
in For loop, x is 000D
in For loop, x is 000C
in For loop, x is 000B
in For loop, x is 000A
in For loop, x is 0009
in For loop, x is 0008
in For loop, x is 0007
X
_
```

Ran a program; R8000 to run the interpreter, it runs the program, then prints "X" when Xiting the interpreter. Note here, uint16s write out in hex. Might change in the future!

Running on a MemberCHIP card from Lee Hart:

Using SCREEN in a terminal window on OSX:

1- Open an OS X terminal session (window)

2- Find the right TTY device. Type: `ls /dev/cu.*`
With the USB-Serial adapter plugged in, you'll get a list, including something like this:
ouedraogoabdoulm.@NB100429 ~ % ls /dev/cu.*
/dev/cu.Bluetooth-Incoming-Port /dev/cu.Mohamine-WirelessiAP /dev/cu.S8-JL_SPP **/dev/cu.usbserial-14140**

3- Then type: screen /dev/cu.usbserial-14140 1200
The 1200 at the end is the baud rate. If you read Lee Hart's manual and can add more time at the end of each line, you can speed this up to 2400 or 4800.

4- To quit the screen app, type **CTRL-A**, then **CTRL-\**.
Type `man screen` in Terminal for further information on **screen**. (use 'enter' or 'space' to scroll, and 'q' to quit).

/Users/john/Desktop/CDP1802-Desktop/TinyPascal/TinyPascal-1802Interpreter

```
Troy:TinyPascal-1802Interpreter john$ ls *hex
PL0Interpreter.hex      assembChanged.hex
TinyPascalInterp.hex  assemblerOut.hex
Troy:TinyPascal-1802Interpreter john$

Troy:TinyPascal-1802Interpreter john$
     screen /dev/cu.usbserial-A600e1mx 1200

MemberCHIP Card MC20ANSA Monitor v2.0AR 14 Feb 2022. Enter "H" for Help.
>H
Commands           Description
--------           -----------
H                  Help
B                  BASIC level 3 v1.1
P                  Play Adventureland
L                  Load program or data (Intel HEX format)
V                  View 1802 registers
Daaaa bbbb<CR>     Disassemble Opcodes from aaaa to bbbb
Maaaa bbbb<CR>     Memory read from aaaa for bbbb bytes
Waaaa dd dd..<CR>  Write to memory until <CR>
Saaaa bbbb<CR>     Save memory at aaaa for bbbb bytes (Intel HEX format)
Taaaa bbbb cccc<CR> Transfer (copy) memory from aaaa to bbbb for cccc bytes
Raaaa<CR>          Run program with R0=aaaa, P=0, X=0, Q=1

All commands are UPPERCASE. All numbers are HEX. <ESC> aborts a command.

> L

Ready to LOAD Program
```

cut-n-paste the full TinyPascalInterpreter.hex file contents. You won't see anything on the screen, but the green LED on the Memberchip card will glow green.

cut-n-paste the full hex file of the program.

```
Ready to LOAD Program
File Loaded Successfully
>L

Ready to LOAD Program
File Loaded Successfully
>
MemberCHIP Card MC20ANSA Monitor v2.0AR 14 Feb 2022. Enter "H" for Help.
>R8000
Currently running your program
TinyPascal 2024-11-10 V02
OK!
0009 0103 00FB
0008 0102 00FC
0007 0101 00FD
0006 0100 00FE
0005 00FF 00FF
0004 00FE 0100
0003 00FD 0101
0002 00FC 0102
0001 00FB 0103
0000 00FA 0104
Finish
```

It will do a cold-start, so hitting cr will give you the welcome message again.

Here is a screenshot of the whole process but note, the paste command, the characters are not echoed back by the 1802 monitor, so the hex file contents are not shown.

```
MemberCHIP Card MC20ANSA Monitor v2.0AR 14 Feb 2022. Enter "H" for
Help.
>H
Commands            Description
--------            -----------
H                   Help
B                   BASIC level 3 v1.1
P                   Play Adventureland
L                   Load program or data (Intel HEX format)
V                   View 1802 registers
Daaaa bbbb<CR>      Disassemble Opcodes from aaaa to bbbb
Maaaa bbbb<CR>      Memory read from aaaa for bbbb bytes
Waaaa dd dd..<CR>   Write to memory until <CR>
Saaaa bbbb<CR>      Save memory at aaaa for bbbb bytes (Intel HEX
format)
Taaaa bbbb cccc<CR> Transfer (copy) memory from aaaa to bbbb for cccc
bytes
```

```
Raaaa<CR>           Run program with R0=aaaa, P=0, X=0, Q=1

All commands are UPPERCASE. All numbers are HEX. <ESC> aborts a
command.

>L

Ready to LOAD Program
File Loaded Successfully
>L

Ready to LOAD Program
File Loaded Successfully
>R8000
Currently running your program
TinyPascal 2025-01-04 V04
makeMatrix...
maxcellcount 0320 and we have 0079
within size, lets run!
printMatrix...
row:0001 ##### #####
row:0002 #         #
row:0003 #######   #
row:0004 #     ### #
row:0005 # ### ### #
row:0006 #   #   # #
row:0007 ##  ### # #
row:0008 #   #   # #
row:0009 # # ### # #
row:000A # #   #   #
row:000B ##### #####
solving...
solved
row:0001 #####o#####
row:0002 #    ooooo#
row:0003 #######ooo#
row:0004 #ooooo###o#
row:0005 #o###o###o#
row:0006 #oo #ooo#o#
row:0007 ##o ###o#o#
row:0008 # oo#  o#o#
row:0009 # #o###o#o#
row:000A # #ooo#ooo#
row:000B #####o#####
Finish
```