Tiny Pascal Quick Start

John A. Stewart
Ottawa, Canada.

V1.4 19 April 2025

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.

The compiler runs on your desktop. The Interpreter runs on your single board computer

The interpreter runs on a SBC - a Single Board Computer; I use:
- Lee Hart's 1802 Membership Cards,
- the EMMA - 02 debugger on my desktop.

Emma-02: https://www.emma02.hobby-site.com/

Lee Hart's Computer kits: https://www.sunrise-ev.com/#microcomputers

Q) Why these boards, not XXXX?
A) Because I'm focusing on getting the compiler and interpreter working, once it is good enough, other platforms and operating systems can be targeted.

INCREDIBLY QUICK START

Download three things,
1) the Emma-02 emulator, (URL just above);
2) the Interpreter assembled for Emma-02 (In the "Interpreter" directory)
3) the example maze-solver, "maze3a.hex", in the "Tests" directory)

Install Emma-02, look below in this document for info on loading / running it, and load in the two hex files in the list above. It'll let you see how it works for one example.

There you go!

To compile the TinyPascal compiler.

Do this step if you want to write your own TinyPascal files.

1) Download the `TinyPascal.pas` file.

2) Download a Pascal Compiler. I use the "FreePascal" compiler on my Linux and MacOS computers.

`https://www.freepascal.org/`

Once installed, on Linux, or MacOS, I open a terminal window, "cd" to the directory containing the `TinyPascal.pas` file, and type

```
fpc ./TinyPascal.pas
```

This should result in a working TinyPascal compiler for you.

3) You can compile your own TinyPascal programs, and (once the typos are removed!) **it will run your program** on your desktop computer, so you can see the results.

Take one of the example programs (I use ".mod" to identify them), put it in the directory where you compiled the TinyPascal.pas file. Lets say you have a program named "test1.mod". In the terminal window, type:

```
./TinyPascal < test1.mod
```

NOTE: the "<" tells the compiler where to get the input file, do not forget it! It will try and tell you if you do forget the "<".

The Compiler will compile and run the program, and will produce an "Assembler" file, which you will, using the A18 assembler, assemble, then download and run on your SBC.

To assemble the program for downloading to your SBC; in a terminal window on your Desktop PC, download and install the "a18" assembler. Grab the A18 assembler from Herb Johnson's website, and follow his install instructions.

https://www.retrotechnology.com/memship/a18.html

To *Compile a TinyPascal program.*

Compile your TinyPascal program; eg:

```
./TinyPascal < maze3a.mod
```

Look in your directory for a file called "`assemblerOut.asm`", which is the code for your program.

Assemble this with the A18 assembler. Here's an example I use, type it on a single line:

```
/Users/john/Desktop/CDP1802-Desktop/a18/a18 maze3a.asm -l maze3a.lst
-o maze3a.hex
```

Run the program.

Now you have a choice to run your program on hardware, in the emulator.

Step a) To run this in the Emma-02 emulator, 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.

The Screenshots below should help you to do this.

Of note:

1) Before you can load a program, you have to START the emulator.

2) I chose the "ELF" computer, with the options shown in the screenshot below.

3) Note that the emulator expects it's **internal ROM** to be at 0000H, so that our programs can load at 8000H (the pcode interpreter) and 8800H (your TinyPascal assembled program)

4) in the green "screen window", press the return key to get the first prompt.

5) the example shown the TinyPascal program just looped and printed a few lines.

Step b) Running on a MemberCHIP card from Lee Hart:

Lee has instructions for Windows computers on his build-docs for his boards.

I used to use SCREEN in a terminal window on MacOS, **now I use Minicom on Linux.**

Lets assume all files in our Desktop are in the directory:

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

Here's what's there:

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

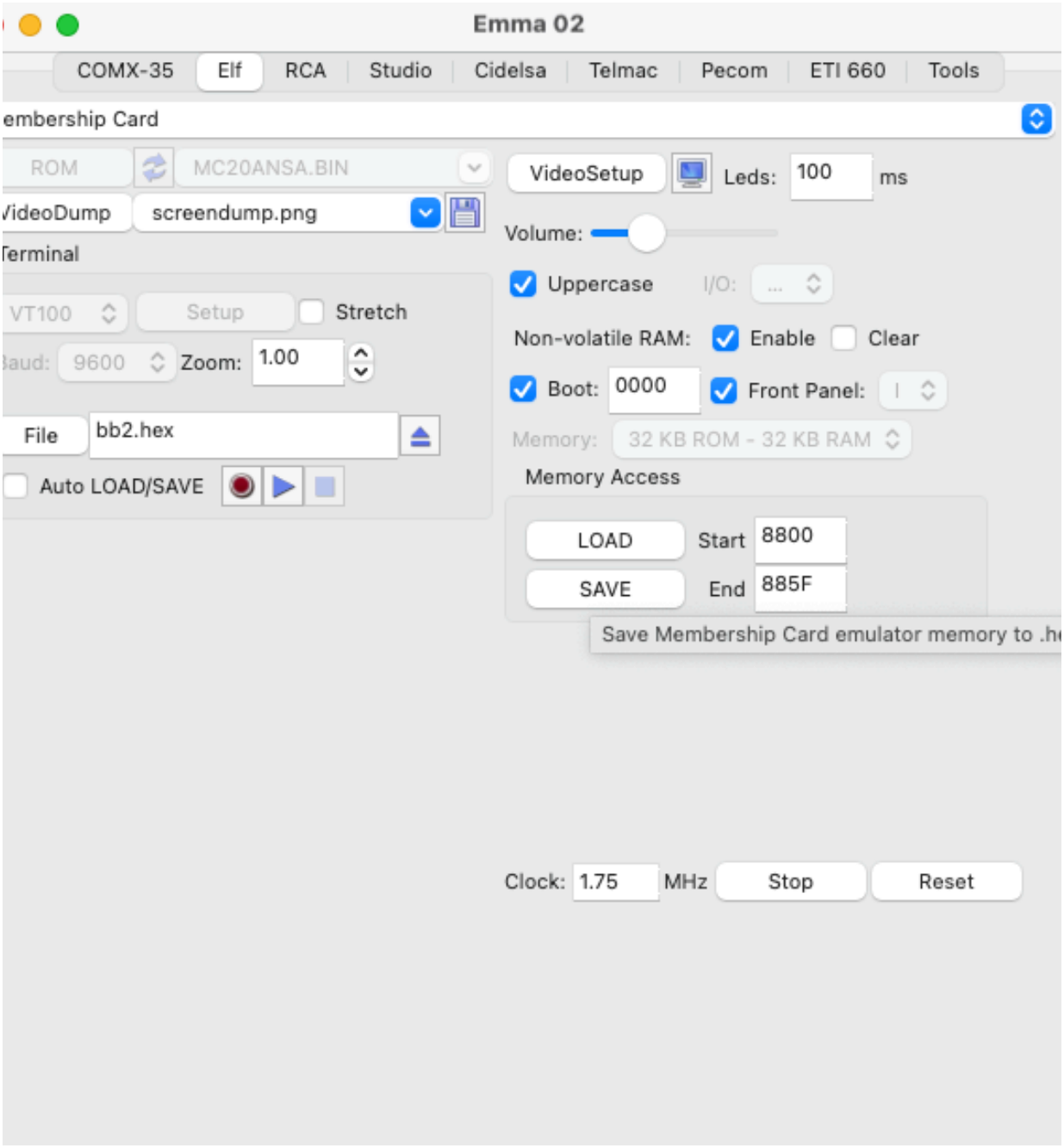
In Minicom, you need to:

a) select the correct port (google this, better instructions than what I have had here before)

b) set the speed to 4800 N 1 (hint in Minicom, type the two keys ctrl-A, release, then the P key.

c) set the Character Tx delays (hint: ctrl-A, release, then the T key) so that the newline delay is 250ms, and the "between characters" delay is 10ms.

Then, type return, and you should have the prompt shown below. (The "H" key gives the help info)



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.
>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
```

copy-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.

copy-n-paste the full hex file of the program. You can do multiple copies and pastes, just remember where you were in the previous copy.

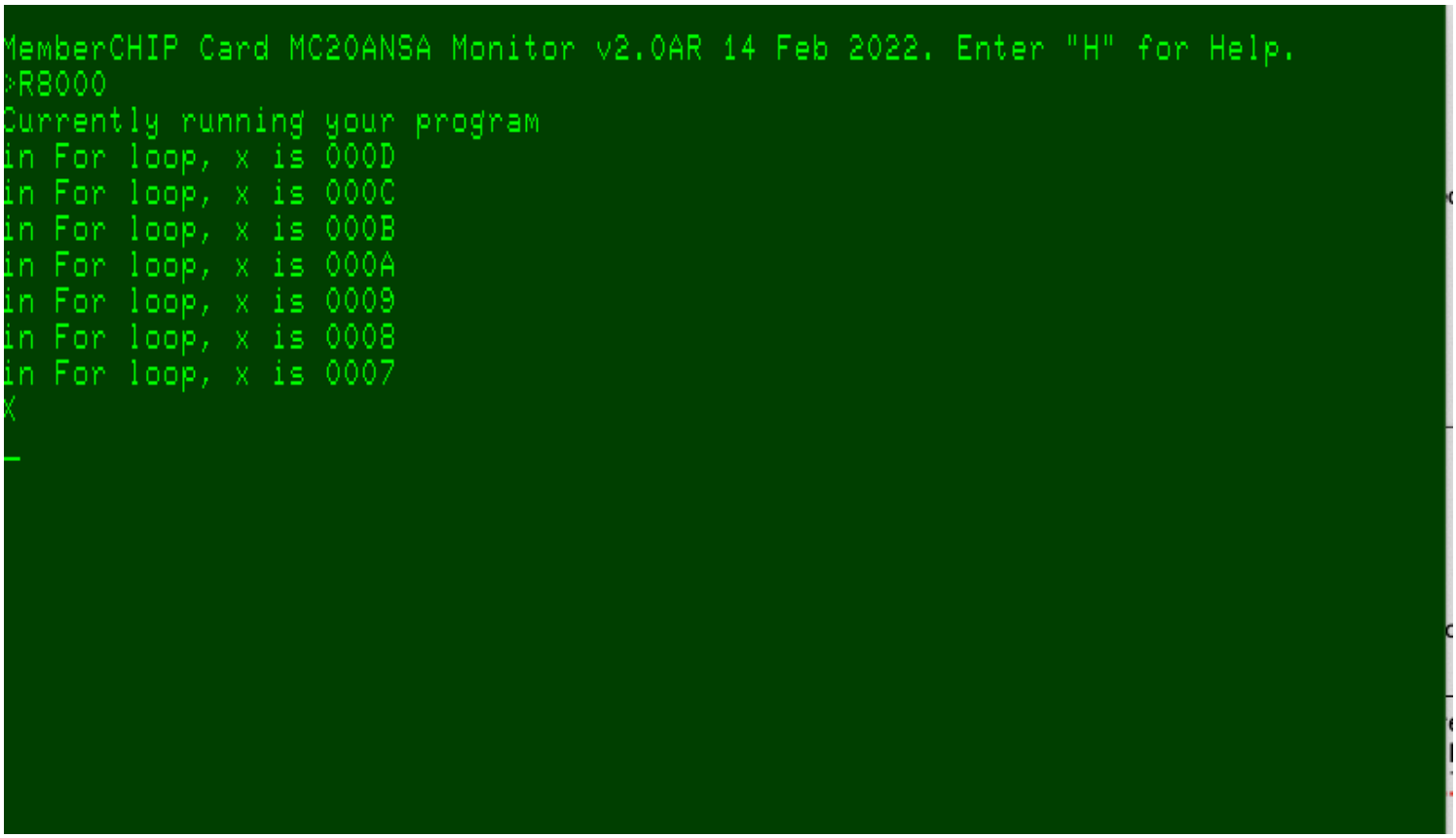
```
File Loaded Successfully
>L

Ready to LOAD Program
```

Do the same cut-n-paste for your test program .hex file.

```
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.



Ran a simple 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!