# Modern TEC Software Development

A twenty first century approach to writing TEC software

By Craig Hart

## The Problem

Writing serious code for the TEC computer has always been difficult.

Key punching hex bytes, looking up opcodes in books, pen and paper, calculating JR offsets, managing memory etc. It was all rather hard.

This approach made anything more than trivial programs of a couple of hundred bytes very, very hard to put together, debug and modify.

As a result, the TEC is often viewed upon as a ‘toy’ and interest is lost since it becomes ‘too hard’ to work on any software of substance.

## A better approach

The obvious solution is to develop code for the TEC on your PC – but how?

A number of tools are required:

* An Editor to write your code in
* An Assembler to compile the code into Z80 binary
* A means to download the code to the actual TEC hardware
* (optionally) A debugger and/or emulator to fault find or if you don’t have a real TEC handy
* Somewhere to save and share your work

Once you have your toolkit put together, the development process becomes quite rapid and straight forward.

You first write the code in Z80 assembly language using regular Z80 instructions e.g. LD A, CP B, JR NZ etc.

The assembler does all the hard work converting these to opcodes, calculating all your JMP addresses, allocating memory locations etc. for you. The Assembler can be told where in memory to build your code for, and to output the result in various formats (binary, intel HEX, source code listing, etc).

Most assemblers can also offer advanced features such as labels, variables, constants, macros and so on which make refactoring code a breeze.

The assembled code is transferred to the actual hardware via a serial link (this can be as easy as 2 or 3 mouse clicks) and then tested on the real machine.

A whole write, assemble, transfer, test cycle can be as short as a minute or two. Since your source code is in easy to manage text file, edits are a breeze.

Suddenly complex software development because far easier, once you are freed from the tedious chores of looking up opcodes, keypunching hex values, doing jump relative math, etc.

### Editor

Text editors are numerous these days; Windows comes with Notepad built in, which works fine.

I personally prefer Notepad++, and there is also VS Code (Visual Studio), to name a few.

In short, any program that saves files as plain text will do, however the third party tools can assist with formatting for readability, syntax checking for typos, code highlighting and colours for readability, etc.

### Assembler

Once Again there are numerous assemblers out there, both free and paid.

I have previously used Oshonsoft’s integrated Z80 development environment, available from [oshonsoft.com](http://oshonsoft.com/) - which offers an editor, assembler, disassembler, debugger and more all in one place.

Today have settled on Telemark [TASM](https://www.ticalc.org/archives/files/fileinfo/250/25051.html) which does the job nicely and has a good feature set. It is DOS based so driven from the command prompt however Oshonsoft can also call TASM directly, making assembly a case of just pressing F9.

I know that John Hardy prefers an all-online approach via [asm80.com](https://www.asm80.com/) which is an online editor and assembler all in one.

### Code Transfer

I have adopted the SC1 hardware as my development environment. The SC-1 includes a serial interface and ROM based transfer routine (Fn 1) that accepts Intel HEX files.

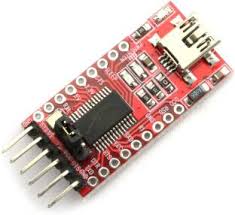
TASM generates this sort of file, so I simply have the file open in Notepad++ (which automatically refreshes onscreen when the file changes), copy and paste the file to the serial port.

I use the well known serial port terminal program PuTTY to keep the serial port open, so uploading new code goes as follows:

* SC: Reset, Fn 1
* PC: Copy .HEX file in notepad++ and   
  paste into PuTTY window
* SC: Fn 0 – code running.

To connect the PC to the TEC I have purchased a cheap USB to Serial board on ebay for a few dollars. Don’t pay $20+ at Jarcar!!

These boards are broadly known under the generic model name ‘ft232’ and are around $2 each. They use a standard USB to mini or micro USB cabe.



John Hardy has an online solution here too, with the Wicked TEC-1 emulator, that also accepts .HEX files.

[Wicked TEC-1 Emulator](https://jhlagado.github.io/wicked-tec1/)

Graphical user interface, text, application

Description automatically generatedCraig Jones has recently produced the TEC-1F which also supports a serial interface; I understand Craig is working on a TEC monitor with serial upload capability also.

### Debugger

Like Z80 assemblers, numerous debuggers exist on the market, but since I have it, I tend to use the Oshonsoft debugger.

There is an amazing Z80 add on for VS code at [Github](https://github.com/maziac/DeZog) which allows direct debugging within the editor.

## Github? What’s that?

Obviously one can simply save one’s work to whatever storage device you prefer (e.g. your C: drive, USB etc.), in the traditional fashion.

One issue here is that of keeping different versions – with software development it’s easy to make a change that doesn’t work – but not always easy to remember how to get back to where you were.

Another issue is – how to share your work and work collaboratively as a team?

Enter, Github.

Now is not the time to promote all of Github’s features, but amongst the major ones, Github addresses these concerns and much more.

Please consider researching how Github can work for you in terms of sharing your work, and keeping your code safe and manageable.

## Summary

Rapid, simple code development is now very achievable at basically no cost. The tools are available freely on the Internet and now, both the TEC-1 and the SC-1 have the necessary interface and software to complete the picture.

Now is also the time to consider upgrading your old TEC-1B or earlier board to a modern TEC-1F or give the SC-1 a go.

Either board offers significant improvements for programmers and will be well worth the expense.

I strongly suggest you try it for yourself.