Project Description

Component Tester Circuit

CTC

Version 1

Peter K. Boxler, October 2019

This docu is part of the GitHub Project Page

Table of Contents

1. Purpose of the Component Tester Circuit 2
2. How it works 3
3. Design of the CTC 3
4. Schematic of the CTC 4
5. Control lines 5
6. Example 5
7. Libraries used 6
8. Conclusion 6
9. Links 8

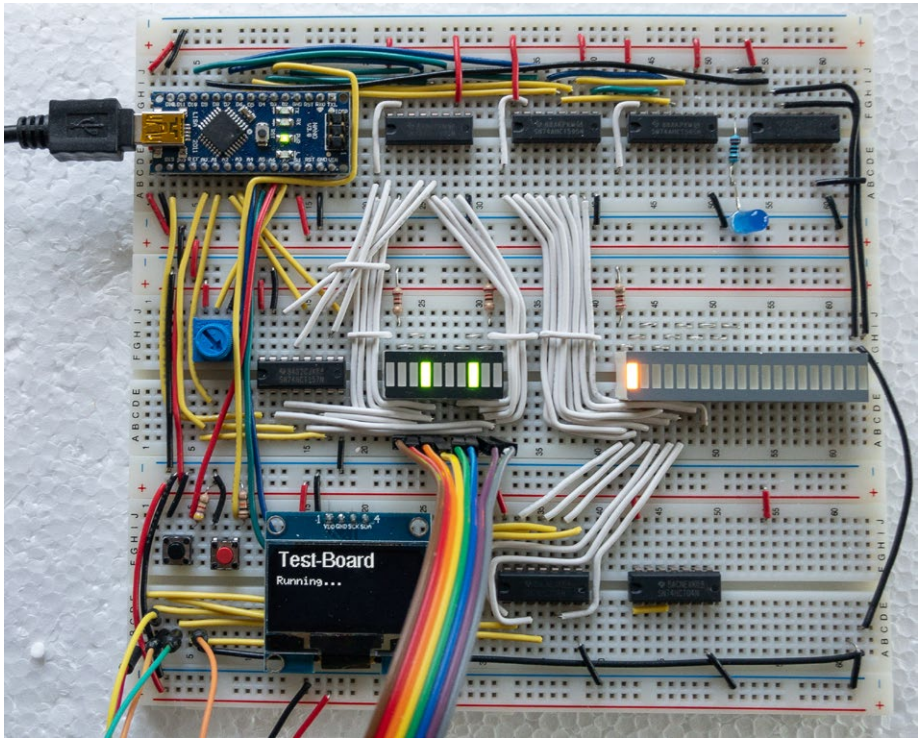
1. Purpose of the Component Tester Circuit

Ben Eater's now almost famous videos on YouTube have inspired others to make their own versions of a SAP computer (Simple as possible, as described by Albert Paul Malvino in his book „Digital Computer Electronics" (c) 1977.)

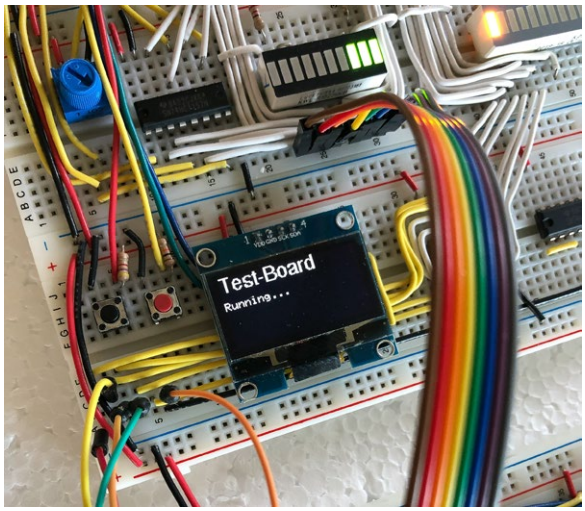
For myself, I plan to follow James Bates videos, whose computer has 8-bit addresses instead of 4-bit addresses used by Ben Eater. Thanks to both for their work and excellent didactic skills in explaining their respective builds.

Manual testing of finished component boards of the 8-bit breadboard computer such as general purpose registers, memory module, ALU, etc. is tedious at best and an Arduino based circuit seems the best way to do this. The Component Tester Circuit (CTC) allows thourough testing of the individual components boards.

The finished CTC circuit looks like this:



Component Tester on Breadbords



OLED display

2. How it works

The function of the CTC ist simple enough: it can assert values to the 8-bit bus and it can also assert control signals. It can also read from the bus. Well, that is all that is needed to test component boards.

Of course, it also needs a user interface to select the desired test and to see the results.

The Ardunio sketch doing all this is not rocket science and one needs just one sub-function for every board to be tested. Integration tests of multiple boards (say a register board and the memory board) is also possible.

For test of a component board do this:

- connect the 8-bit bus
- connect the necessary control line (more on those later)
- select the desired test (or write a new sketch for a new component)
- start the test
- check the result on the display

3. Design of the CTC

The idea of the CTC was inspired by Ben Eater's EEPROM programer, check out his video here

Ben Eater EEPROM

The CTC uses an 8-bit bus (easily expandable to 16-bits) and supports up to 16 control lines (easily expandable to > 16 lines).

The Arduino sketch shifts out the value to be written to the bus into a shift register 74HCT595, after enabling the output, the 8-bits are asserted to the bus.

Likewise for the control signals: since 16 control signals are supported two shift register chips are needed. The control signals are connected to an led bar and hex inverters are used for the active low signals. The control signals instruct the component boards to do their thing. Another Arduino pin is used to output the clock signal.

Since the 74HCT595 chip has a 3-state output we can also read from the bus. To save Arduino pins a selector chip 74HCT157 is used. The lower 4 bits of the bus are connected to input A, the higher 4 bits to

input B. With that arrangement we can read 8-bits from the bus with just 5 Arduino pins. (4 bits and one pin to select A/B input).

The user interface consists of two pushbuttons, a potentiomer (for speed control) and a 128x64 OLED display (i2c).

Pushbuttons are used as follows:

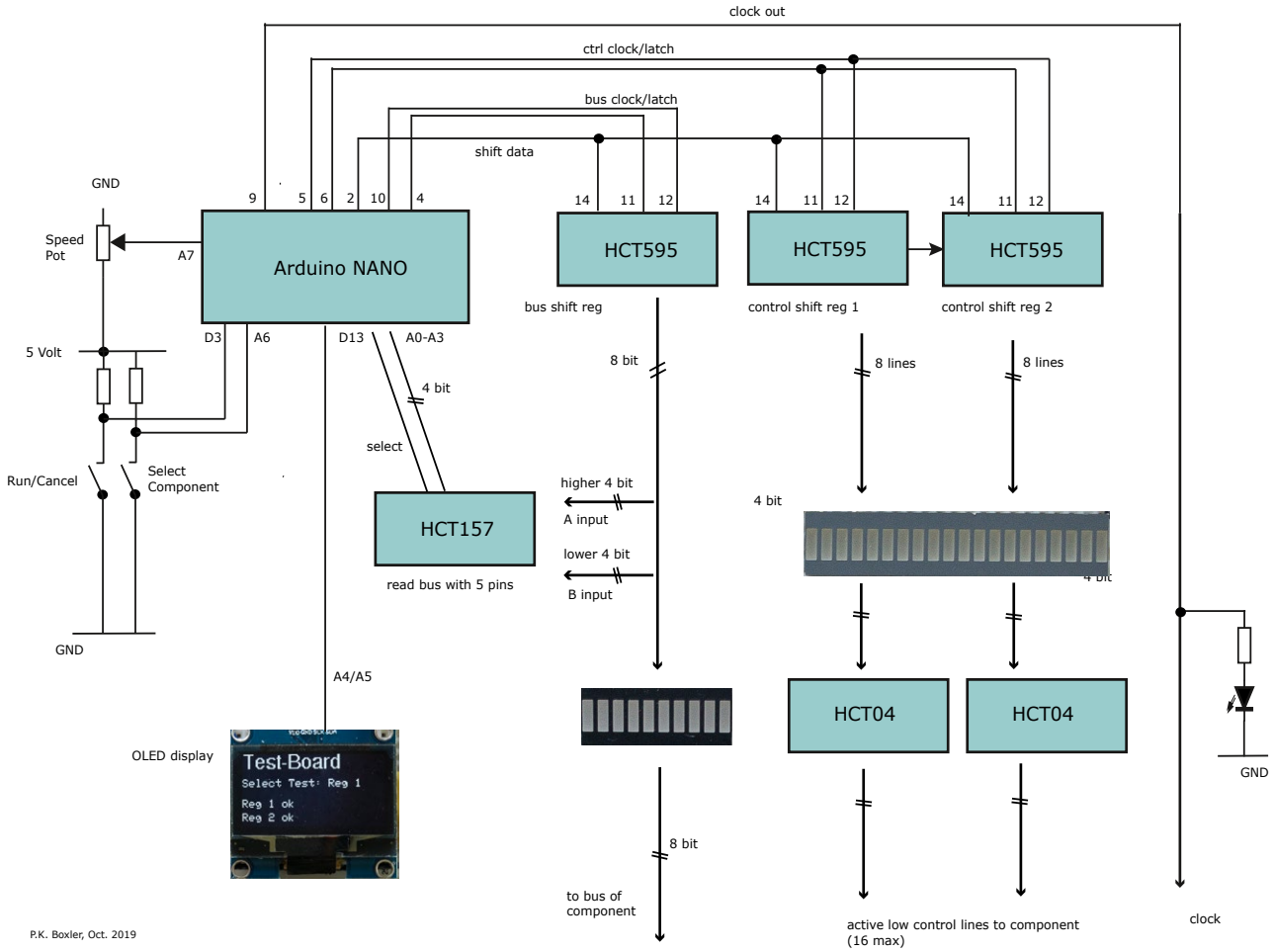
- Pushbutton 1: multiple pushes to select the desired test, this button ist connected to the A7 pin on the Arduino (adc).
- Pushbutton 2: start the selected test or cancel a running test, this button is connected to pin A3 on the Arduino, it triggers interrupt 1.

Here is what the OLED display shows after running a test of a register board with two general purpose registers:

OLED display after running reg test

4. Schematic of the CTC

This is an overview of the CTC. Note the two pushbuttons on the left, the potentiometer on the left , the two led bars for bus and control lines and the OLED display.



Schematic Component Tester

5. Control lines

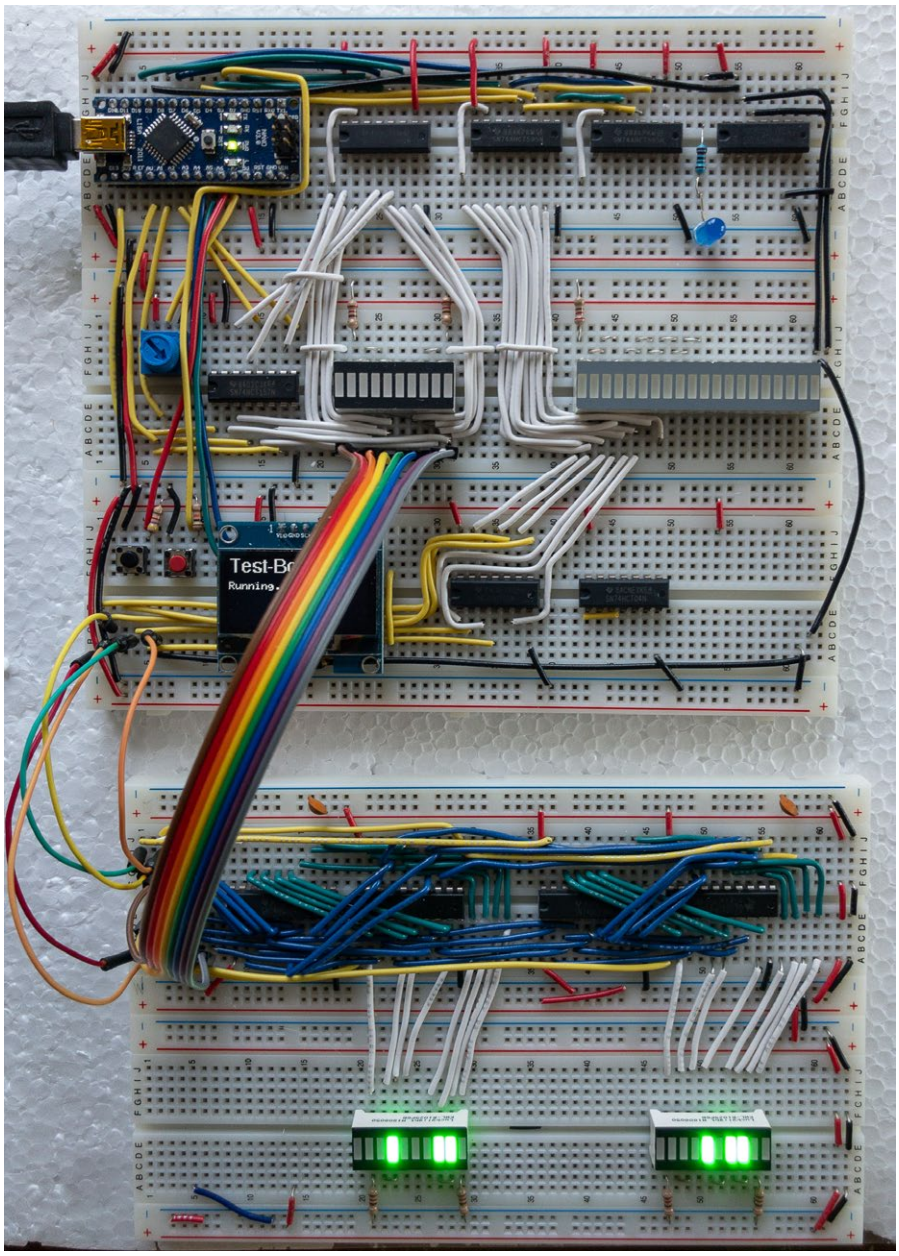
The CTC in its current implementation supports 16 control lines. They are simply numbered 0 to 15 since their semantic depends on the board to be tested. The sketch and the connections to the board under test has to match.

6. Example

This pic shows the CTC hooked up to a board containing two 8-bit general purpose registers. Connections include the 8 bus lines, four control lines and the clock line. The Arduino sketch runs a test on both registers, one after the other and displays the result on the display (see above). After that another test (or the same) can be selected.

The test of one register performs this function (see Arduino sketch **reg_test1.ino**) on GitHub:

it loops over all values from 0 to 255 asserts the value onto the bus, sends the control signal to write register (RxW), then, after a short delay, it sends control signal RxE and then reads the value from the bus. It compares this to what is has written. If all values match the register seems to work ok.



CTC connected to register board

7. Libraries used

The Arduino sketches need the following libraries:

```
#include <stdint.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SH1106.h>
```

and also the precomplier directives for debug output. Author Andreas Spiess, found here. Very useful, but can be replaced by Serial.print().

https://github.com/SensorsIot/Pre-Compiler-Directives

8. Conclusion

Works as designed. Easy enough. More fun than fiddling with wires and looking at leds at the same time.

9. Links

My Raspberry Projecs website
projects.descan.com

CTC on GitHUB

Ben Eater on YouTube
Ben Eaters website
Ben Eater on reddit's

James Bates on YouTube
James Bates on GitHub

James Sharman on YouTube

end of document

This page intentionally left blank (last page)