CMOS 4000 And 7400 Series

Integrated Circuit Tester

Based on the work of Baweja Akshay and expanded by A.S.G

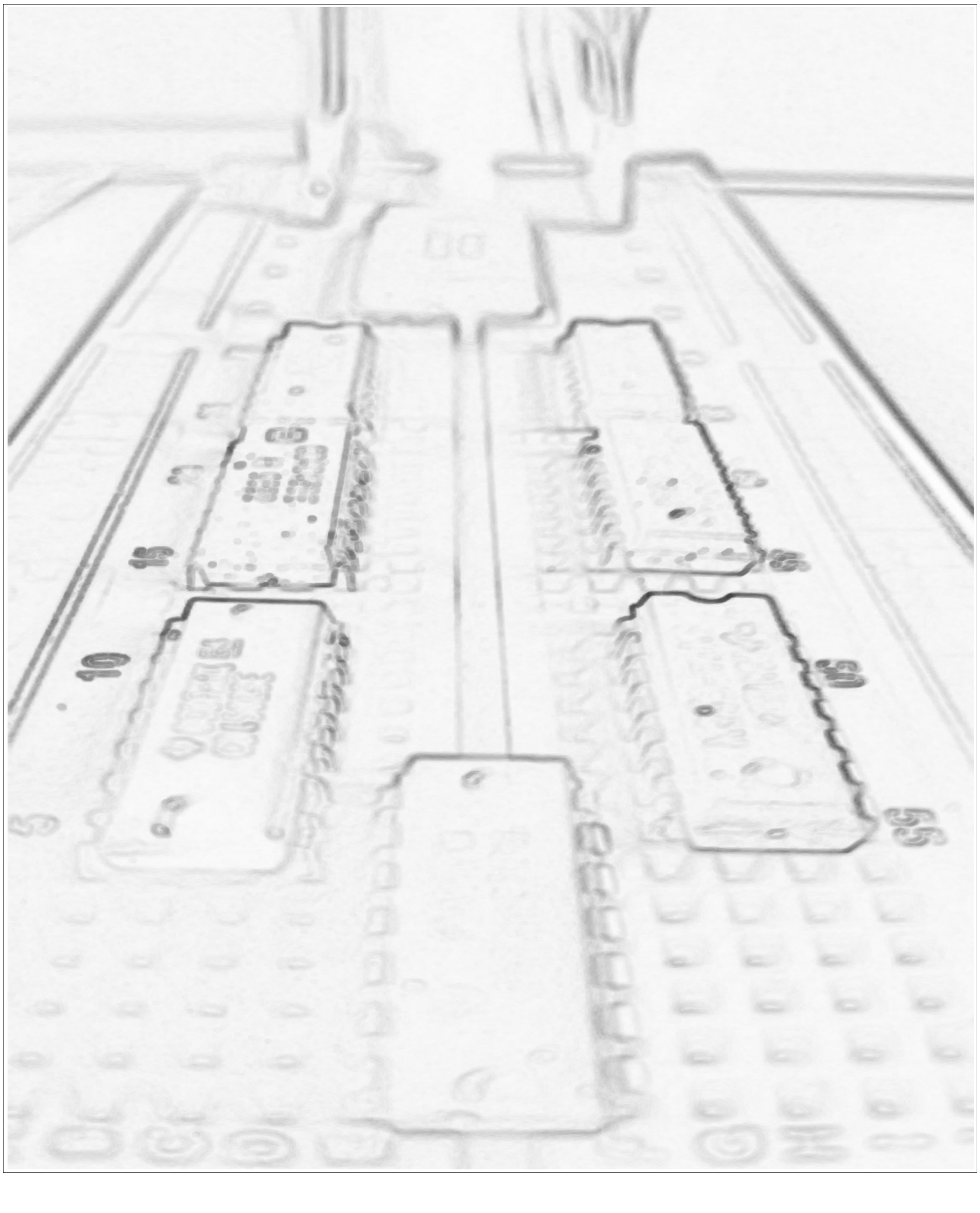
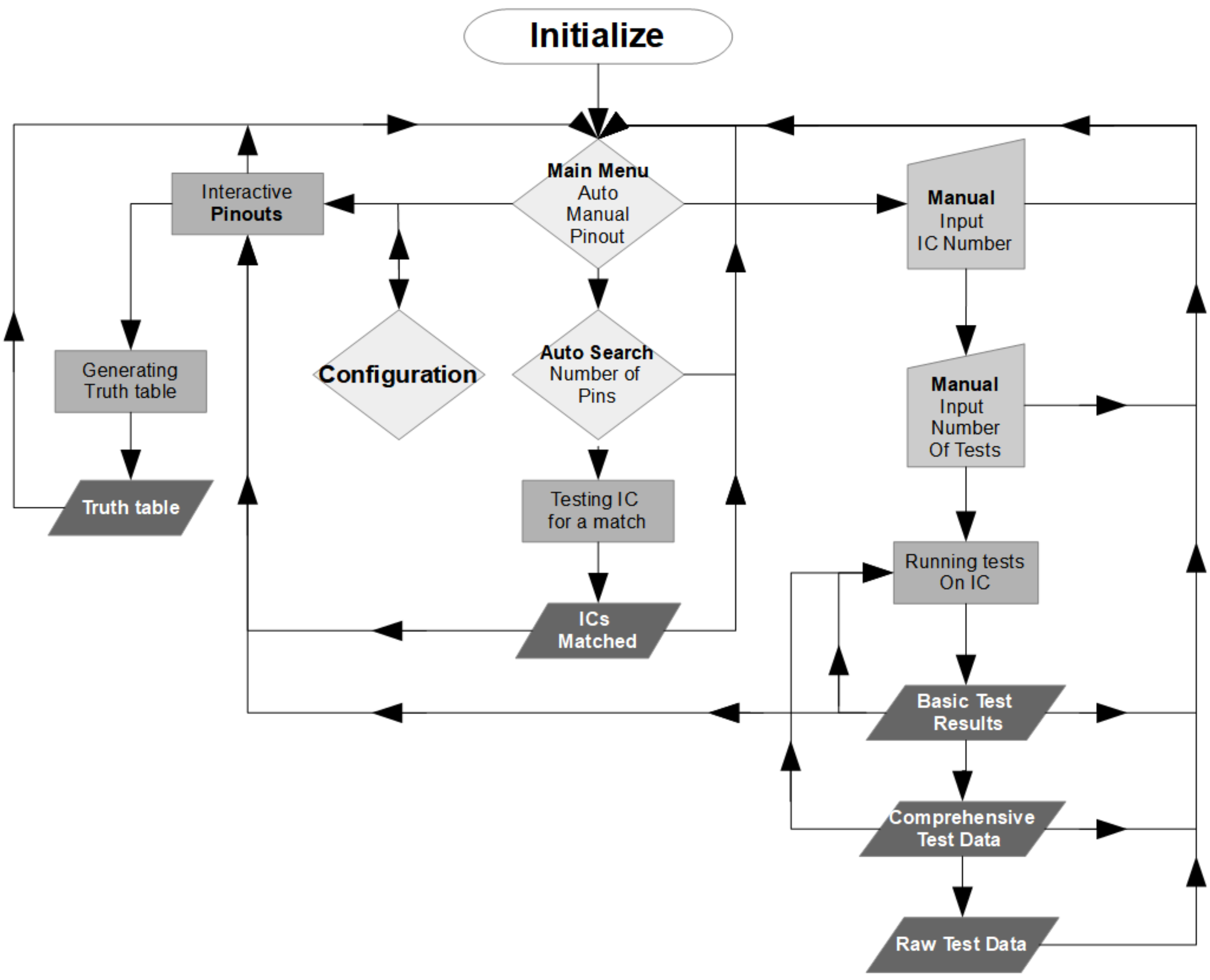


Table of Contents

Improvements and additional features

- No need to reboot the device after each test.
- Interactive graphical layout of Ics which lets the user check the functionality of Ics manually without additional circuitry.
- Truthtable/Timing diagram generation, both manually and automaticaly generated. Saves the truthtables as a CSV file.
- Loop testing available, the program keeps track of both passed and failed tests as well as which pins are the most like to be the culprits.
- Shortcut for Last test routine so the user can quickly test a sum of the same IC types.
- Configuration where the user can tailor the behaviour of the program to suit their testing needs.
- Added tests and pinouts for 20 & 24 Pin ICs

Flowchart / Overview of Modes

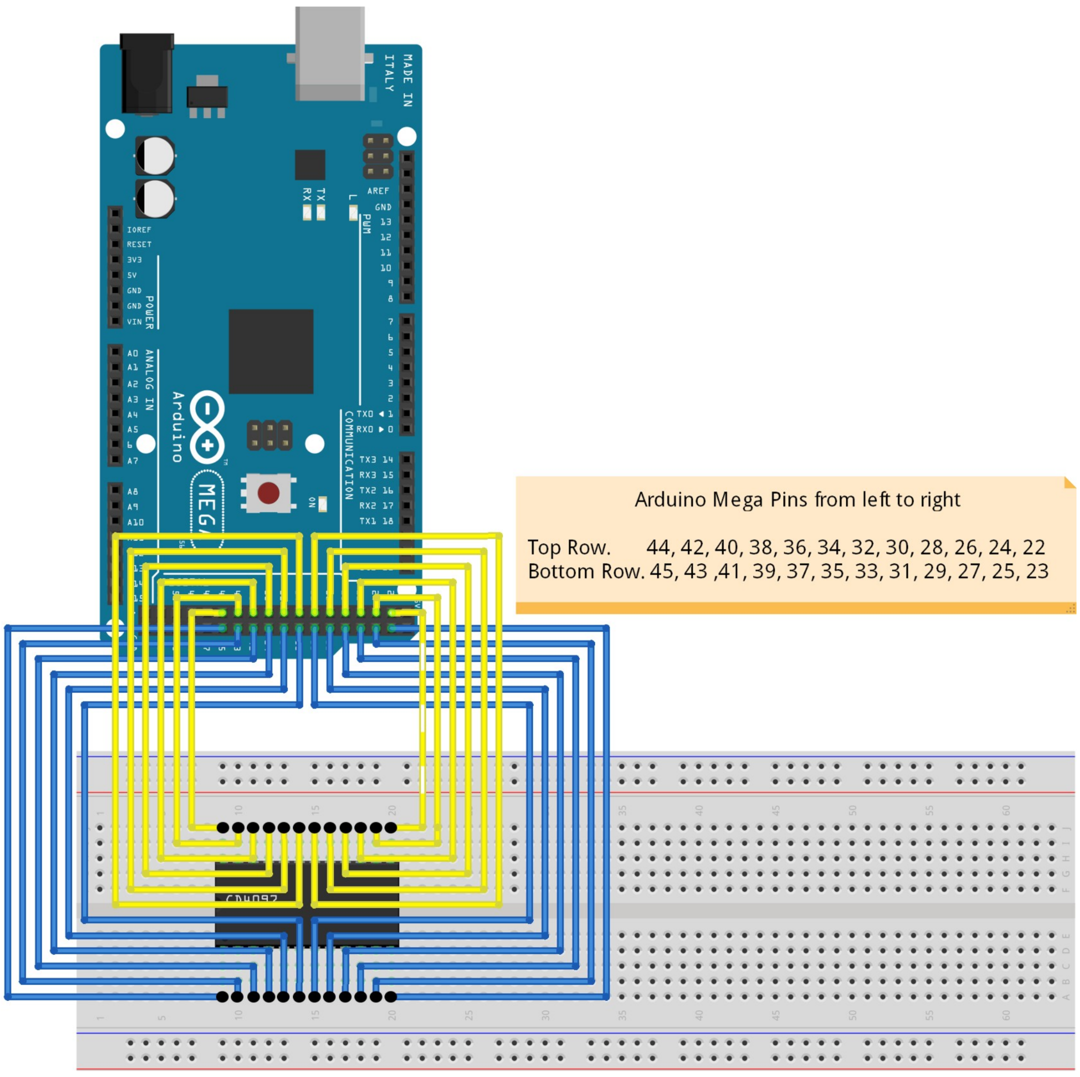


Setup

Bill of Materials.

Arduino Mega 2560.
2,4” TFT Screen with an SD card slot(shield that you can plug directly on top of the arduino Mega see picture on the next page).
Breadboard(24 pin ZIF socket optional).
24 wires.
ICs to test.

Wiring on breadboard



Arduino Mega Pins from left to right

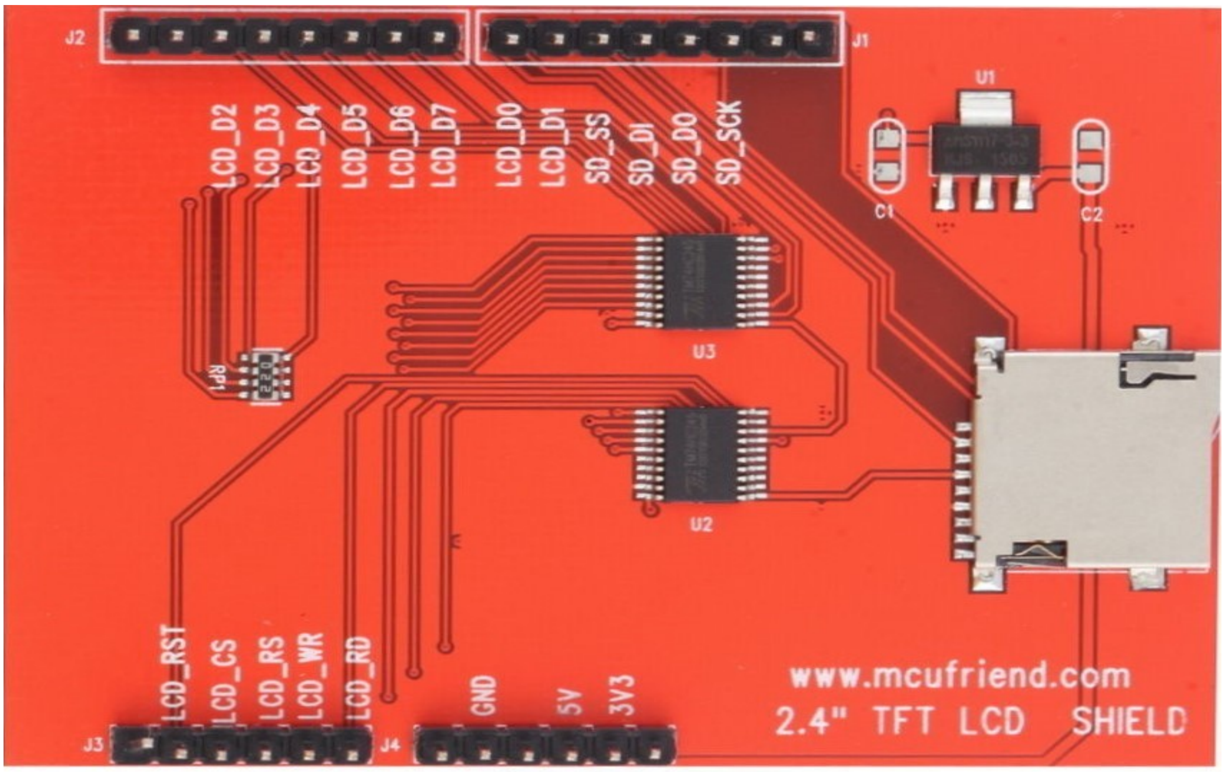
Top Row. 44, 42, 40, 38, 36, 34, 32, 30, 28, 26, 24, 22
Bottom Row. 45, 43 ,41, 39, 37, 35, 33, 31, 29, 27, 25, 23

I haven’t made a proper PCB(shield) yet so this fritzing image of my setup on a breadboard will have to do for now...

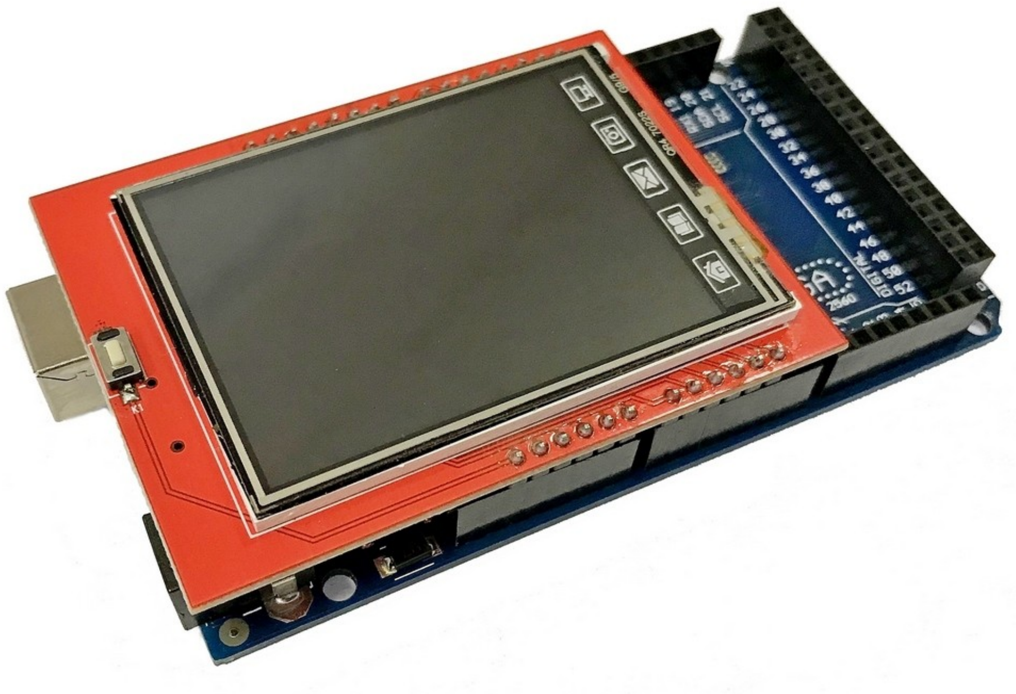
If you find the image hard to read since it has a lot of overlapping wires the IC pins and arduino IO connect as the table below .

IC Pinout	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	**21**	**22**	**23**	**24**
Arduino IO	**45**	**43**	**41**	**39**	**37**	**35**	**33**	**31**	**29**	**27**	**25**	**23**	**22**	**24**	**26**	**28**	**30**	**32**	**34**	**36**	**38**	**40**	**42**	**44**

2,4” TFT Shield Pinout



Shield placement on Arduino Mega2560



Intergrated Development Enviroment

Arduino IDE or other compatible IDE (It was developed on Arduino IDE 2)

Libraries for arduino

All the libraries that the tester uses are in the library folder in my GitHub.

Adafruit GFX or SPFD5408
(depends on the controller for your screen, most of the $10-$20 ones are of the SPFD variant)

Put it into **Documents/Arduino/Libraries**

Kicksort
Sorting library needs to be copied to **Documents/Arduino/Libraries**

SD library
This library needs to be referenced to directly or you need to override the one that comes with the arduino IDE. Placing it into you library folder in documents **isn't enough!!!**

for Arduino IDE 2 on Windows 10 it can be found here **C:\Users\UserName\AppData\Local\Arduino15\libraries**

SD Card Files

Copy **Pinouts.txt** & **Database.txt** which can be found in the main folder of the program onto a SD card and insert it into the TFT shield.

TFT screen troubleshooting

Unfortunately there are plenty of different screens and controllers being sold as the same thing. Same controller and the same pinouts yet they need to be addressed a bit differently

I have only acquired and tested the SPFD5408 variants and even they don't work exactly the same... :[

- If your screen only shows white after you have uploaded the sketch you probably need the other one. As in the adafruit type instead of the SPFD one or vice versa.

- If you have graphics on your screen after uploading but touch doesn't work. try uncommenting lines 63 to 68 in IC_Tester.ino and comment out the lines 54 to 57.

- If the touch is inverted you might need to switch the last values on lines 16 & 17 in Flow.ino depending on how it’s inverted...
```
p.y = map(p.y, TS_MINX, TS_MAXX, tft.height(), 0);
p.x = map(p.x, TS_MINY, TS_MAXY, 0,tft.width());
```

- If the touch is still inverted or more like at two places at once... :/ then you could try to uncomment out line 159 in **SPFD5408_TouchScreen.cpp** that can be found in the SPFD5408 folder

  ```
  return TSPoint(x,y,z);//uncomment this line
  ```

- If then the touch is only inverted then change the line to something of the sort

  ```
  return TSPoint(1023-x, 1023-y, z);
  ```

- Here are some helpful links:
  https://www.instructables.com/How-to-use-24-inch-TFT-LCD-SPFD5408-with-Arduino-U/
  https://create.arduino.cc/projecthub/SurtrTech/interfacing-and-fixing-touch-problem-on-tft-lcd-2-4-shield-fd7738

If an IC fails a test. Don't panic!

There are few things that could be wrong other than the IC being faulty.

- Check the wiring connections if testing is done on a breadboard and traces/connectivity if done on a PCB
- Test a known good device if on hand and see if it passes the test.
- If the good device passes the test, check if the failed IC is from the same manufacturer as the one that passed the test. Sometimes the difference between them is active on rising edge versus falling edge which makes a test for one of them out of sync while the other one passes.(e.g HEF Versus TI) Better yet sometimes a pin is present on an IC from one manufacturer but not from another manufacturer e.g HEF4068(NAND) Vs CD4068(AND&NAND)
- **There might be an error in the test for that type of IC**. I was only able to test about 50% of the 4000 series and less than 10% of 7400 series since I don't own nearly all of them. My speculation is that Baweja Akshay was in a similar boat since I have already fixed few tests myself as added some from scratch as well. So check the datasheet, use Pinout Mode to verify the logic and see if the logic of the test is off . Often its as simple as getting the IC to an “initial state” before beginning the test. Further on that in Database.txt section of the manual
- **Note** that some 4000 & 7400 series IC's can´t be tested without external components e.g 4045 which needs a crystal oscillator to function.

Tips ,Tricks and Miscellaneous Info that didn’t fit anywhere else

- Always have a datasheet to reference to when testing unfamiliar Ics.
- If there isn´t a test available for the type of IC you want to test and you want to roll your own. See if there is a version of it in Pinout Mode. Sometimes the datasheet doesn´t remove all doubt about the sequence of events on the logic of the IC therefore with the aid of Pinout mode you can easily verify the logic of each pin which does help with creating a test for said IC.
- General handling of CMOS chips is touched upon further on the next page
- Pin 1 on the IC under test is always connected to IO 45 of the Arduino <u>no matter if it’s a 14, 16, 20 or a 24 pin chip.</u>
- So far I’ve only tested CMOS chips, 4000 series and the 74HCxx variant of the 7400 series. So TTL chips have not been thoroughly tested nor developed with in mind. Test at you own risk!

Please be advised to never insert or remove an IC unless the program is in Main Menu!
In most other modes there might be voltages present on the pins.

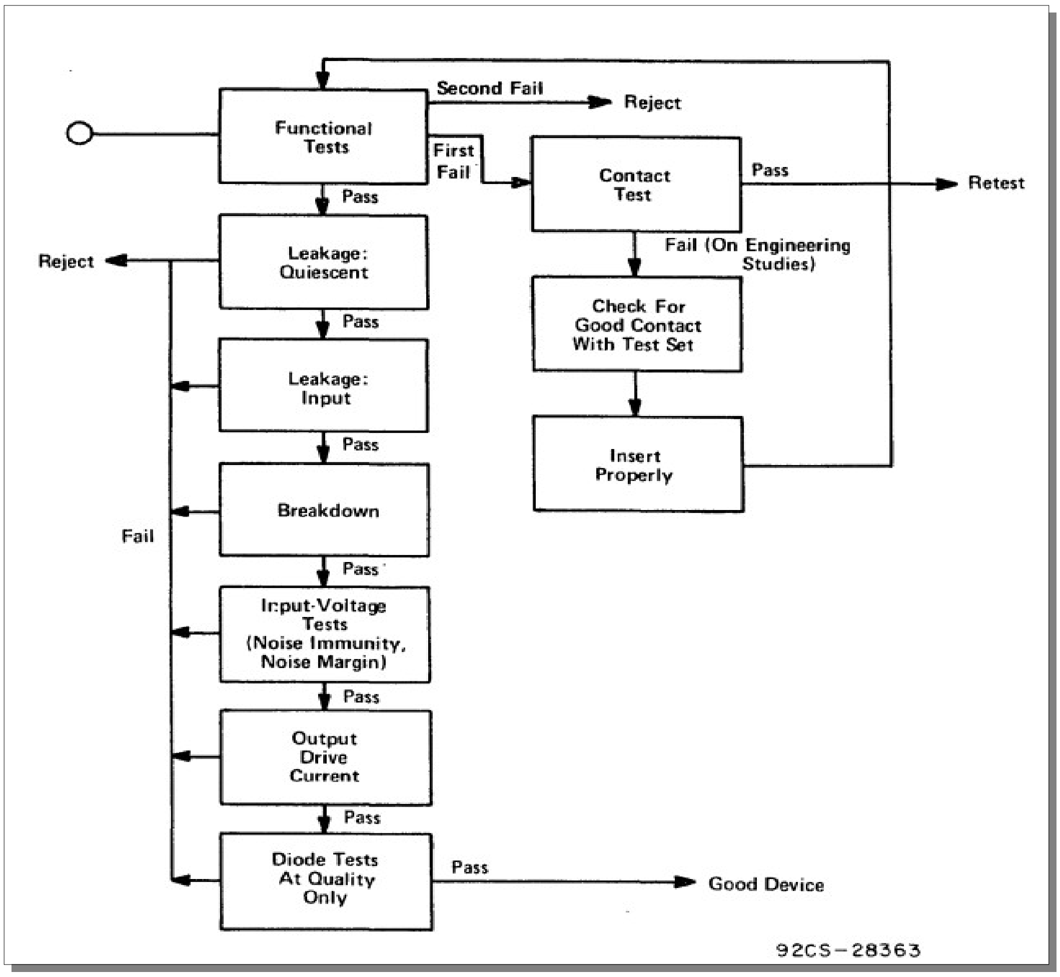
Fundamentals of Testing COS/MOS Integrated Circuits

From 1983 RCA CMOS_Databook: page 740

ICAN-6532

"Users should follow the sequence below when testing COS/MOS devices:

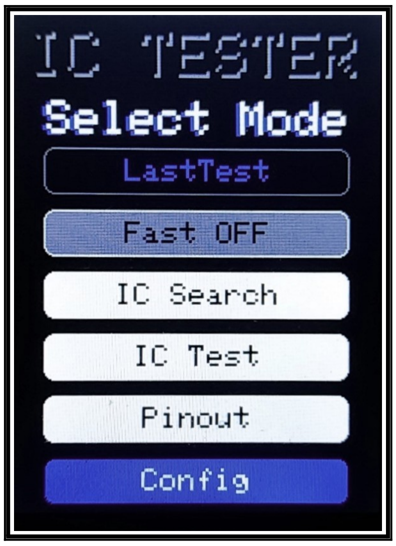
I. Insert the device into the test socket.
2. Apply VDD.
3. Apply the input signal.
4. Perform the test.
5. On completion of test, remove the input signal.
6. Turn off the power supply (VDD).
7. Remove the device from the test socket and insert it into a conductive carrier. COS/MOS, devices under test must not be exposed to electrostatic discharge or forward biasing of the intrinsic protective diodes"



Note*
This tester only tests the logical functions of an IC at a low speed and is therefore only a small indication of a health of an IC. Even if the IC under test passes all the tests on the tester there might be other issues with it that only analog tests could give a truthful image. *See flowchart of a typical test sequence of an IC from **RCA CMOS Databook** above.*

*The Arduino Input pins are high impedense and so are the input pins of CMOS devices therefore if the IC under test is heating up there is probably something wrong with it. Very small current is present in this circuit when the IC works as intented. **The Arduino Mega and the TFT screen generate some heat however so that is to be expected.***

IC Tester Modes



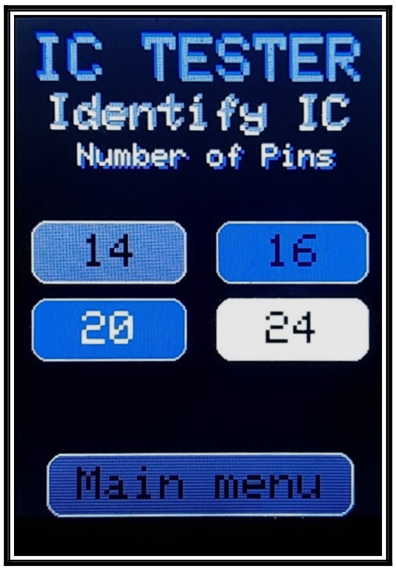
<u>Last Test:</u>

Repeates the last IC test or Truth table/ Timing diagram Cycle.

<u>Fast Searching/Testing ON/OFF:</u>

Disables the update of graphics in IC Search and IC Test which speeds up the process quite a bit.

<u>IC Searching/Identifying:</u>



Goes through a database of tests to determine which IC charachteristics the IC under test behaves as.

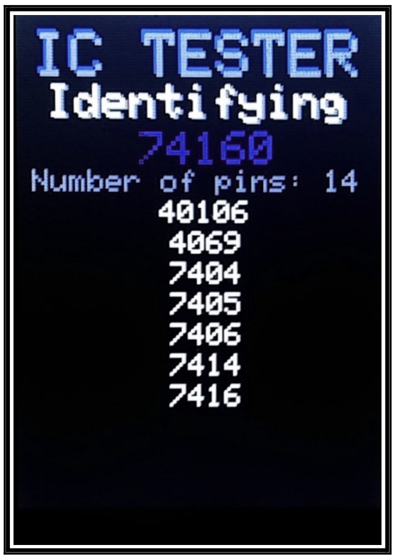
1. Select the number of pins on the IC (14, 16, 20, 24)

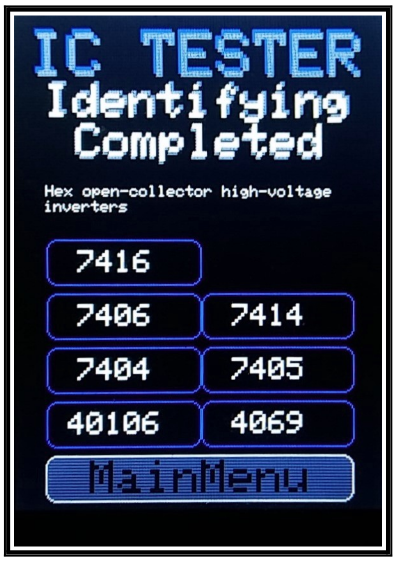
2. Runs tests. If fast mode is OFF then the Ics that are being tested against will be displayed on the screen.

3. Displays the names of IC's that the IC passed the tests.

   Note that often there are few Ics that are very similar in function usually the only differences are buffered versus unbuffered outputs and open collector versus open drain etc. So it's not uncommon for the tester to find more than one type of IC that functions like the one under test.(very common with gate type ICs they come in all sorts of flavors and hex inverters) e.g 40106 tests the same as 4069, 7404, 7405, 7406, 7414 and 7416 which are all hex inverters with the same pinout but different analog properties.

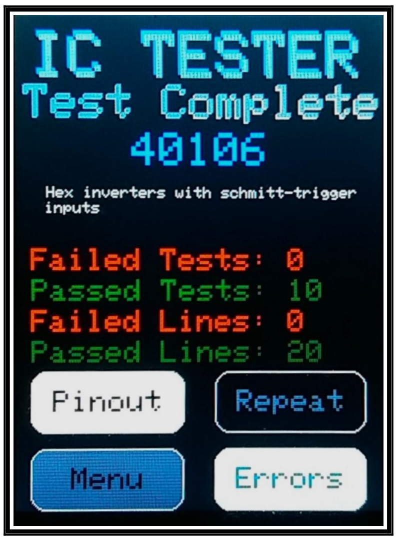
4. User can choose to go to interactive pinout by using the 74xx/4xxx buttons or main menu. **Note** that the further down the pinout.txt file a IC information is stored the longer it takes for the program to load it.





IC Testing:

Runs the IC through series of tests and keeps track of which test the IC failed and/or passed



```
IC TESTER
Test Complete
40106
Hex inverters with schmitt-trigger
inputs

Failed Tests: 0
Passed Tests: 10
Failed Lines: 0
Passed Lines: 20
```

1. Enter the number of the IC under test

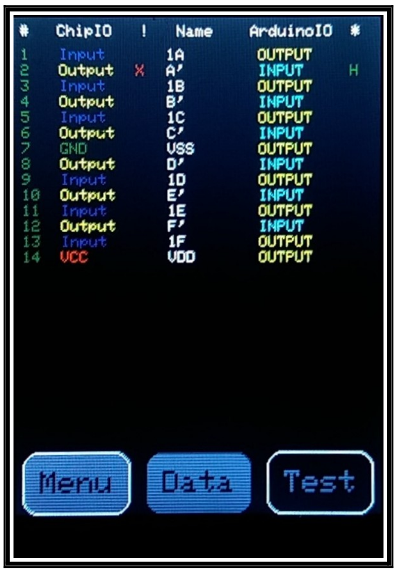
2. Enter the number of tests that the program should execute
   (entering 0 makes the program loop until the user stops the test)

3. While the program tests the IC the user can see how many tests the IC has passed or failed **IF Fast mode is disabled.** <u>*the user can stop the test at anytime by touching the screen for a second*</u>

   The varables that keep track of number of passed, failed tests etc, are 16-bit positive integers(**word** variable) so they cap out at 65536 in which the program rolls them over to 0. This can be easily modified if the user choses so by making them into 32-bit **unsigned long** variables which will roll over after 4,294,967,295(this will however make the number to big for the screen and will wrap around to the other side.

   The variables that keeps track of number of passed & failed tests can be found in the **IC_Test.ino**

4. When testing has finished the number of passed and failed tests is displayed and the user can navigate to the Error screen which has more detailed information about the results.

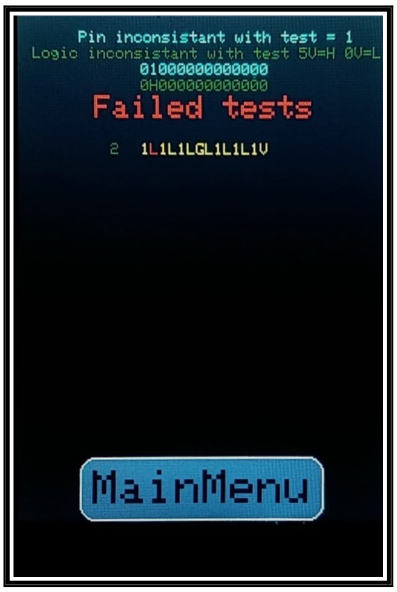
   

   ```
   #   ChipIO   !  Name   ArduinoIO  *
   1   Input       1A     OUTPUT
   2   Output   X  A'     INPUT      H
   3   Input       1B     OUTPUT
   4   Output      B'     INPUT
   5   Input       1C     OUTPUT
   6   Output      C'     INPUT
   7   GND         VSS    OUTPUT
   8   Output      D'     INPUT
   9   Input       1D     OUTPUT
   10  Output      E'     INPUT
   11  Input       1E     OUTPUT
   12  Output      F'     INPUT
   13  Input       1F     OUTPUT
   14  VCC         VDD    OUTPUT
   ```

   a) <u>**Error section**</u>**:**

      1) Pin number
      2) Pin functions
      3) Which pins were most likely the ones that failed the test(if any)
      4) Pin names,
      5) Arduino IO's
      6) The state of the pin that failed a test.

      Note. Since the program has no way of testing(reading) the input pins of the IC with any level of certainty the user must read a bit further into the results to make sure that an **Input** pint isn't creating a false error reading of an **Output** pin. <u>The Pinout section is a great way to manually check the functions of each pins on the IC.</u>

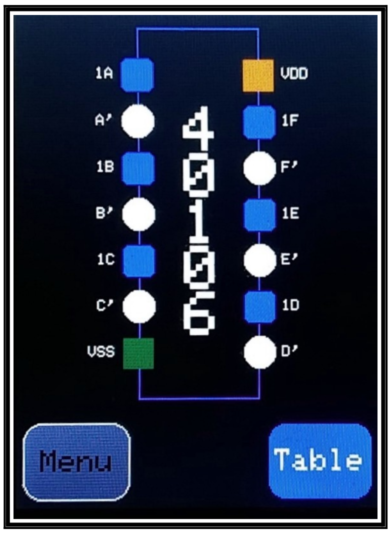
   ```
     Pin inconsistant with test = 1
   Logic inconsistant with test 5V=H 0V=L
           01000000000000
           0H000000000000
            Failed tests

       2   1L1L1LGL1L1L1V
   ```

   b) <u>**Data section**</u>**:** Here the user can see which lines of the test and pins the IC under test had fail. Not only useful to see which pins don't work as expected but also very handy when creating your own tests to be able to spot errors in them and where those errors are.

      The top 4 lines are a leftover from development that serve little purpose after I got the program to pinpoint which pins were failing tests and properly displaying them with the lines that the IC failed the tests. It does however no harm being there so I kept it for time being...

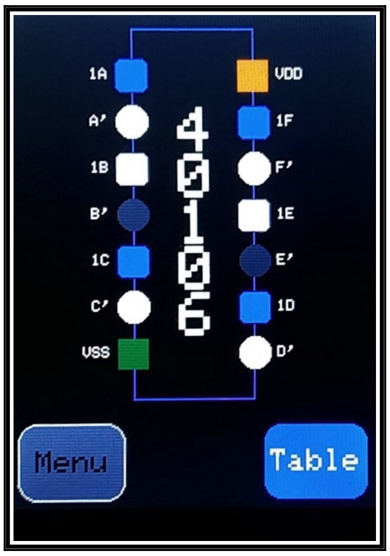
Pinout: → Truth table/Diagram

Interactive graphical representation of the chosen IC. From here the user can go to the Truthtable/ Timing diagram section of the program.



- In the graphical display of the user chosen IC the user can touch/set the inputs and see the readings of the outputs. It can be really useful to see the logic of a chip at the pace that you want.
- If the IC has a clock pin a **Clock button** will be displayed and when activated will similated a clock pulse on that pin.e.g 4040, 4024 amd 4017. **The clock speed can be changed in configuration**.
- If the IC is a Mux/Demux or any IC with bi-directional input/outputs an **In/Out button** will be displayed and when activated will switch inputs with outputs and vice versa.eg. 4053,4067 and 4097.
- The state of inputs are transferred to the truthtable/diagram mode but reset when navigating from the Truth table mode back to Pinout mode.

Pins graphical layout



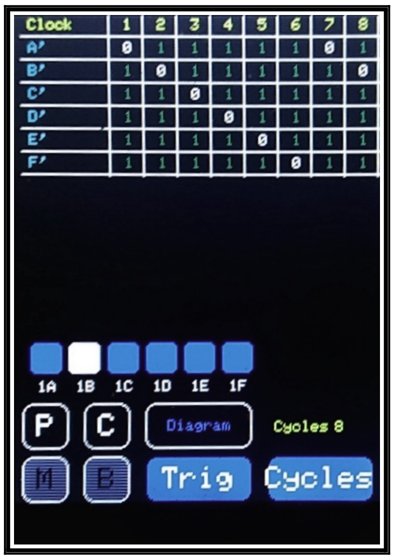
- 0V VSS/GND Power pins are **Green** squares
- 5V VDD/VCC Power pins are **Red** squares
- 0V(Logic Low) Inputs are **Blue** squares
- 5V(Logic High) Inputs are **White** squares
- 0V(Logic Low) Outputs are **Blue** circles
- 5V(Logic High) Outputs are **White** circles
- 0V(Logic Low) Clock pins are **Purple** squares marked with a C
- 5V(Logic High) Clock pins are **White** Squares marked with a C
- 0V(Logic Low) Mux Inputs are **Grey** squares
- 5V(Logic High) Mux Inputs are **White** squares
- 0V(Logic Low) Mux Outputs are **Grey** circles
- 5V(Logic High) Mux Outputs are **White** circles

The naming convention for inputs/outputs etc, on both the 4000 and 7400 series can be a bit inconsistent and even differ between books and datasheets so some of that inconsistancies have transferred into the IC tester. The book I mostly references to regarding the 4000 series is the* RCA CMOS Databook 1983 and for the 7400 series it was the **Texas Instruments Digital Logic Pocket Data Book (REV.B)

To make things even more confusing I had to make the designing choice of only having '8' characters per pin name which left me with shortening many names making it harder to read. If this is a bother it can be „easilly“ rectified by changing the names in Pinout.txt

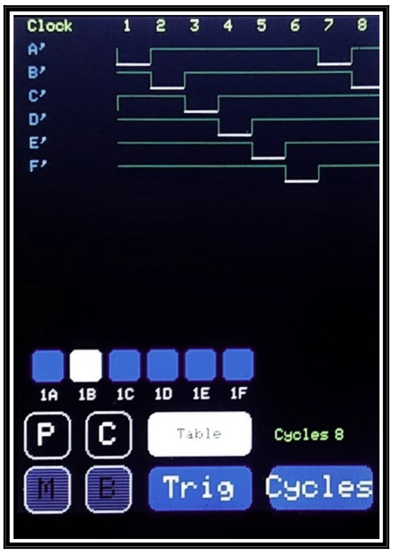
Truth table/Timing Diagram

Here the user can create a truth table / timing diagram.



Clock	1	2	3	4	5	6	7	8
A'	0	1	1	1	1	1	0	1
B'	1	0	1	1	1	1	1	0
C'	1	1	0	1	1	1	1	1
D'	1	1	1	0	1	1	1	1
E'	1	1	1	1	0	1	1	1
F'	1	1	1	1	1	0	1	1

- **Truth table/Timing diagram button** switches between the two.
- **Trig button** switches between rising edge / rising and falling edge for the input logic. The user can manually control the input logic(High or low) using the Input buttons.
- **Cycle button** lets the user select how many clock cycles to generate the truth table. If there is a Clock pin on the IC the user must push the Clock menubutton for the generation to start but if the IC has no Clock pin the program will automatically pulse each Input pin with a High to Low signal for as many clock cycle as the user has chosen. If you wish to stop the cycle press anywhere for a second.
- **P-Button** Will print out on the serial monitor the truth table that has been saved to the SD card in CSV format. See ***Converting CSV into a Timing Diagram section.*** *The serial communication speed is by default* ***115200*** *and can be changed in code.*



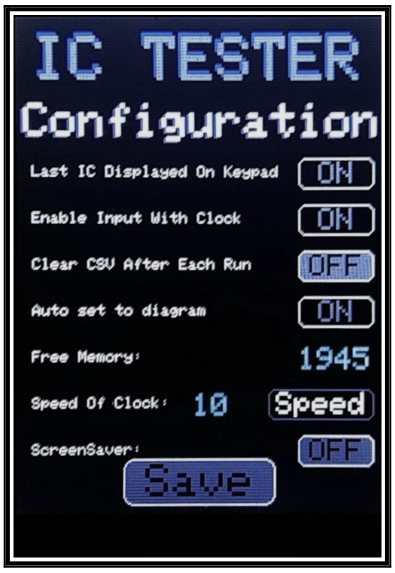
- **C-Button** Erases the truth table file.
- **B-Button** Goes back to Pinout section.
- **M-Button** Goes back to main menu
- **Input Buttons** If you want to manually construct a truth table or diagram you can use these buttons(see trig button) and you can set an Input High and then make the program cycle through the rest of the buttons.

  Sometimes you have to enable a pin before a clock can start which you can do here but in cases where you don't disable ***Enable Input with Clock*** in the configuration menu.

Configuration

Here the user can set some parameters which makes life easier.

- **Last IC displayed on keypad.** Helpfull if the user is repeatedly testing the same kind of an IC.
- **Enable Input with Clock.** Helpfull if the IC under test in truth table generation has a Clock pin and also an Enable/Clear pin that needs logic High signal for the outputs to be active e.g the 74273 IC



IC TESTER
Configuration

Last IC Displayed On Keypad ON
Enable Input With Clock ON
Clear CSV After Each Run OFF
Auto set to diagram ON
Free Memory: 1945
Speed Of Clock: 10 Speed
ScreenSaver: OFF

- **Clear CSV after each run.** Sometimes you would want to record all of your truth table generated other times it would be unnessecery clutter.
- **Auto set to diagram** Helpfull if the user is repeatedly using the diagram over the truth table
- **Free Memory** Here you can see how much of the memory is left on the stack. Testing many different Ics in a session slowly starts eating up the memory and if the tester is acting irrationally then a reboot might be in order.
- **Speed of Clock 1-10.** 1 = slowest, 10 = fastest.
- **Screensaver 1-2.** Set the screensaver On/Off and the time 1 = 30 seconds, 2 = 60 seconds.

Constructing your own tests

It´s fairly easy to make your own tests if your IC is not already in the database as well can you override the one already in place if you find it to be incomplete.

1. Get a hold of a datasheet for the IC. Make sure it´s from the same manufacturer and that the suffixes match the one´s you have on hand.
2. Check if there is already a pinout of said chip in the program. If not, go through the trouble of creating one (see pinouts.txt section above) since it will make it many times easier to make a test if you can verify the sequence of events manually.
3. Reference below on how the tests are constructed.

Database.txt

Tests parameters supported:

0 = Logic Low(0V) applied on Input
1 = Logic High(+5V) applied on Input
L = Logic Low(0V) read on Output
H = Logic High(+5V) read on Output
X = Don't Care read on Output
C = Clock pulse for Ics with Clock inputs ***see 4017 example.***
G = 0V for VSS Pin
V = +5V for VDD or VCC Pin

Basic Example Code

```
$4000                                   //$Name of IC
Dual 3-input NOR gate and inverter      //Description of IC(Only one line)
14                                      //Number of pins on IC
00000HG1LH000V  //First line of test. Applying +5V on Pin 8 and reading High on Pins 10 & 6, Low on Pin 9
00000HG0HH000V
00001LG0HL100V
00010LG0HL010V
00011LG0HL110V
00100LG0HL001V
00101LG0HL101V
00110LG0HL011V
00111LG0HL111V   //Last line of test. Applying +5V on Pins 3,4,5,11,12 & 13 Reading High on Pin 9 and Low on Pins 6 and 10
```

Example with a Clock Pin

```
$4017
Decade counter (5-stage Johnson counter) with 10-output decoder
16
LLHLLLLGLLLH0C1V      //First line of test. Applying +5V on Pin 15(Reset) to clear the counter to zero count, Pin 13(Inhibit) is set to Low
                      //Applied a clock pulse to Clock pin to make sure that the IC resets to zero. All outputs are read Low except Q0 and
                      //Carry Out. Note that in this case and many other the IC needs to be set to an "initial state" first before actual testing
                      //can can begin. Often X(don't care) is adviced on Outputs while/before the IC reaches its "initial state" since the
                      //Outputs could be in a random state when the IC is powered on (very common with shift registers)

LHLLLLLGLLLH0C0V      //Second line of test and the second Clock pulse. Now Output on Pin 2 is High(Q1) and Reset Pin has been pulled
                      //Low
LLLHLLLGLLLH0C0V      //Third line of test and Clock pulse. Pin 4(Q2) and Pin 12(Carry Out) are High
LLLLLLHGLLLH0C0V      //Fourth line of test and Clock pulse. Pin 7(Q3) and Pin 12(Carry Out) are High
LLLLLLLGLHLH0C0V      //Fifth line of test and Clock pulse. Pin 10(Q4) and Pin 12(Carry Out) are High
HLLLLLLGLLLL0C0V      //Sixth line of test and Clock pulse. Pin 1(Q5) is High and Pin 12(Carry Out) is Low
LLLLHLLGLLLL0C0V      //Seventh line of test and Clock pulse. Pin 5(Q6) is High and Pin 12(Carry Out) is Low
LLLLLHLGLLLL0C0V      //Eight line of test and Clock pulse. Pin 6Q7) is High and Pin 12(Carry Out) is Low
LLLLLLLGHLLL0C0V      //9th line of test and Clock pulse. Pin 9(Q8) is High and Pin 12(Carry Out) is Low
LLLLLLLGLLHL0C0V      //10th line of test and Clock pulse. Pin 11(Q9) is High and Pin 12(Carry Out) is Low
LLHLLLLGLLLH0C0V      //11th line of test and Clock pulse. Pin 2(Q0) and Pin 12(Carry Out) are High
LLHLLLLGLLLH1C0V      //12th line of test and Clock pulse. Pin 2 (Q0) and Pin 12(Carry Out) are High
```

Pinouts.txt

Pin Functions supported:

Input: General Input
Output: General Output
Clock: Clock pin, this pin can be clocked by the program
In/Out: Bi-directional pin(mux/demux etc.) by default an Input
Out/In: Bi-directional pin(mux/demux etc.) by default an Output
NC: Not connected(non functional pin on an IC)
VCC: Positive voltage Pin(in our case+5V)
GND: 0V Pin

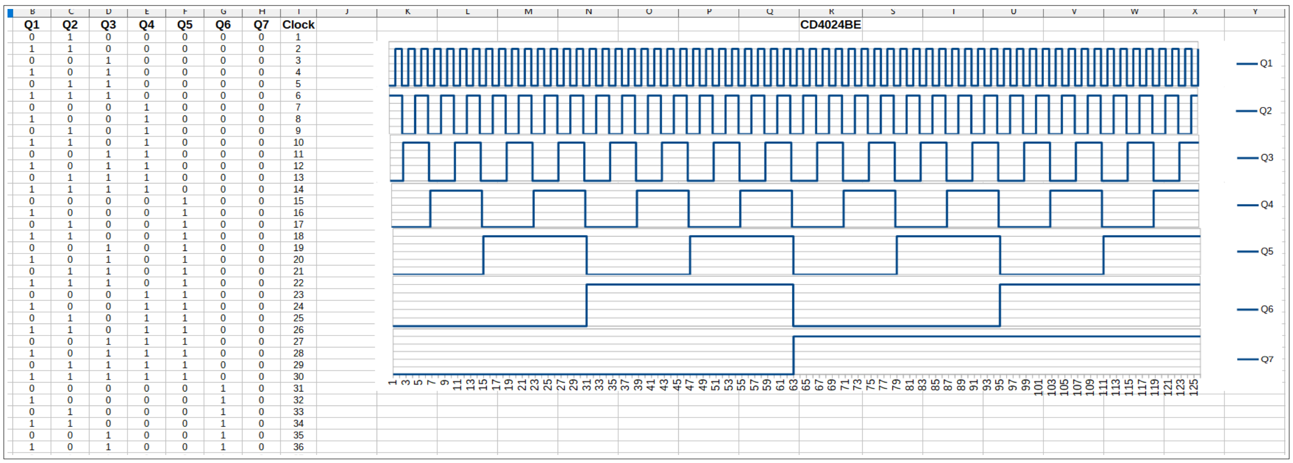
Basic Example Code

```
$4000                                                //$Name of IC ($ should always be in front of the IC name for seperation purposes)
Dual 3-input NOR gate and inverter                   //Description of IC(One line only!)
14                                                   //Number of Pins on IC
NC      %      NC                     Pin 1          //Left side = Pin Function %  Right side = Pin Name
NC      %      NC                     Pin 2          //8 characters %  8 characters
Input   %      1A                     Pin 3          //The pins on the left on the IC must end on the 8th
Input   %      1B                     Pin 4          //character for formating purposes. In this example
Input   %      1C                     Pin 5          //it's the first 7 pins.
Output  %   Out 1                     Pin 6
GND     %      0V                     Pin 7
Input   %2A                           Pin 8          // The next 7 pins are the right side of the IC and their
Output  %Out 2                        Pin 9          // pin names should not exceed 8 characters.
Output  %Out 3                        Pin 10         //Watch out for whitespaces after the pin names they will cause formatting issues
Input   %3A                           Pin 11
Input   %3B                           Pin 12
Input   %3C                           Pin 13
VCC     %+V                           Pin 14
```

Example Code with both Bi-directional Input/Outputs and a Clock pin

```
$4034                                                       // Name of IC
8-stage bidirectional parallel/serial input/output register // IC description
24                                                          // Number of pins on IC
In/Out %     DB8                                    Pin 1   // Bi-directional port set as Input by default
In/Out %     DB7                                    Pin 2   // The ports can be switched in Pinout Mode
In/Out %     DB6                                    Pin 3
In/Out %     DB5                                    Pin 4
In/Out %     DB4                                    Pin 5
In/Out %     DB3                                    Pin 6
In/Out %     DB2                                    Pin 7
In/Out %     DB1                                    Pin 8
Input  %Enable A                                    Pin 9   // General Input
Input  %SerialIn                                    Pin 10
Input  %     A/B                                    Pin 11
GND    %      0V                                    Pin 12 // Ground Pin(0V)
Input  %P/S                                         Pin 13
Input  %A/S                                         Pin 14
Clock  %Clock                                       Pin 15 // Can be clocked in Pinout and Truth table Mode
Out/In %DA1                                         Pin 16 // Bi-directional port set as Output by default
Out/In %DA2                                         Pin 17 // The ports can be switched in Pinout Mode
Out/In %DA3                                         Pin 18
Out/In %DA4                                         Pin 19
Out/In %DA5                                         Pin 20
Out/In %DA6                                         Pin 21
Out/In %DA7                                         Pin 22
Out/In %DA8                                         Pin 23
VCC    %+V                                          Pin 24 // VCC (+5V)
```

Converting CSV into a Timing Diagram



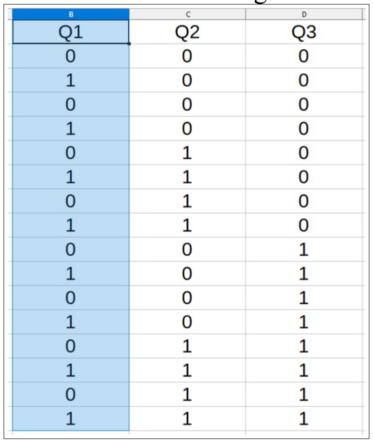
Q1	Q2	Q3	Q4	Q5	Q6	Q7	Clock
0	1	0	0	0	0	0	1
1	1	0	0	0	0	0	2
0	0	1	0	0	0	0	3
1	0	1	0	0	0	0	4
0	1	1	0	0	0	0	5
1	1	1	0	0	0	0	6
0	0	0	1	0	0	0	7
1	0	0	1	0	0	0	8
0	1	0	1	0	0	0	9
1	1	0	1	0	0	0	10
0	0	1	1	0	0	0	11
1	0	1	1	0	0	0	12
0	1	1	1	0	0	0	13
1	1	1	1	0	0	0	14
0	0	0	0	1	0	0	15
1	0	0	0	1	0	0	16
0	1	0	0	1	0	0	17
1	1	0	0	1	0	0	18
0	0	1	0	1	0	0	19
1	0	1	0	1	0	0	20
0	1	1	0	1	0	0	21
1	1	1	0	1	0	0	22
0	0	0	1	1	0	0	23
1	0	0	1	1	0	0	24
0	1	0	1	1	0	0	25
1	1	0	1	1	0	0	26
0	0	1	1	1	0	0	27
1	0	1	1	1	0	0	28
0	1	1	1	1	0	0	29
1	1	1	1	1	0	0	30
0	0	0	0	0	1	0	31
1	0	0	0	0	1	0	32
0	1	0	0	0	1	0	33
1	1	0	0	0	1	0	34
0	0	1	0	0	1	0	35
1	0	1	0	0	1	0	36

The example above was created using Libre Office. It's probably possible to create similar diagrams with other spreadsheet programs as well.

1. **1.** Copy the CSV file from the SD card or even easier method is by printing the data out on serial monitor. Paste it into a text file, save that file as CSV file.

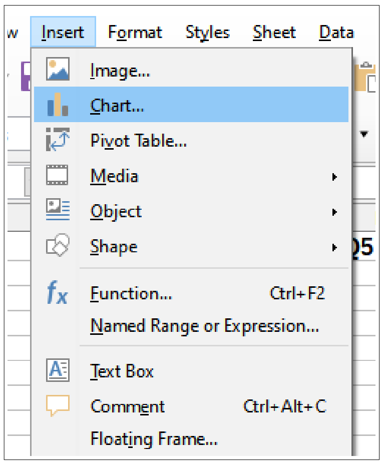
2. **2.** Open it in a program of choice (in this example I'll be using Libre Office)

3. **3.** *Example Data.* Select the row and colums for the first diagram.

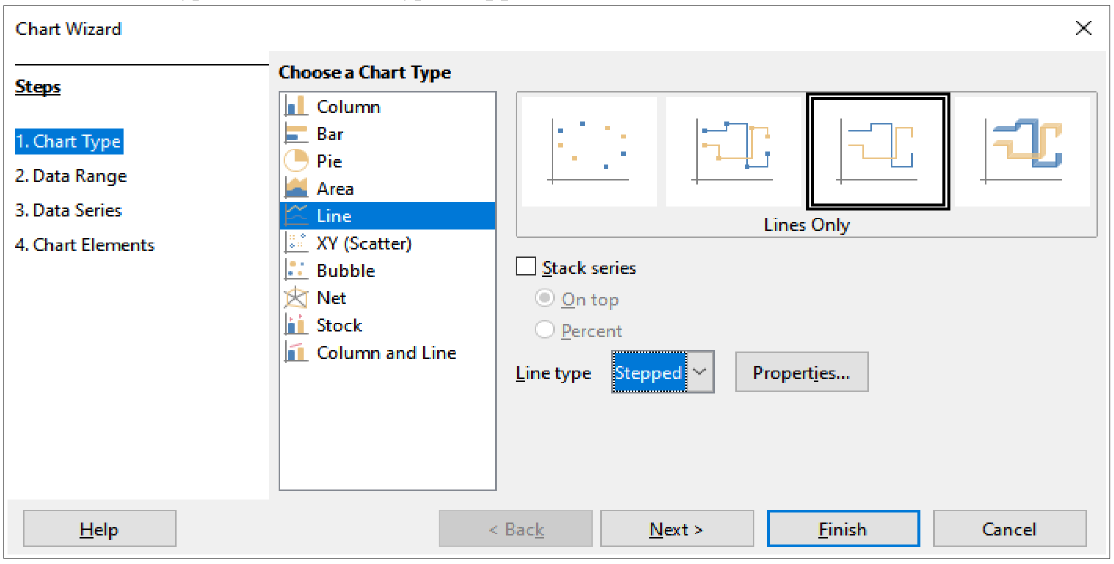


Q1	Q2	Q3
0	0	0
1	0	0
0	0	0
1	0	0
0	1	0
1	1	0
0	1	0
1	1	0
0	0	1
1	0	1
0	0	1
1	0	1
0	1	1
1	1	1
0	1	1
1	1	1

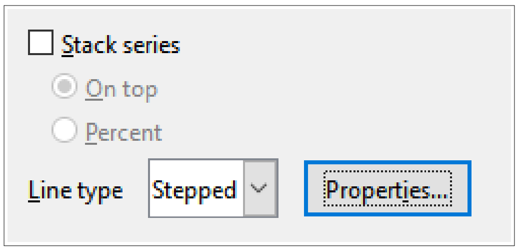
4. **4.** Insert **Chart**.



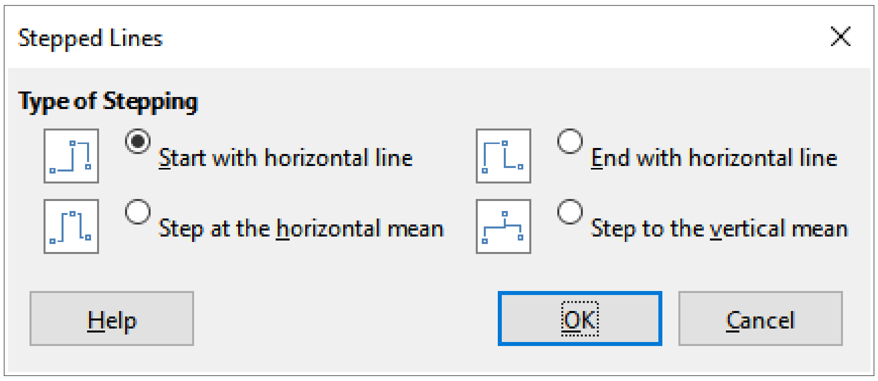
5. Select *chart type* **LINE** and *Line type* **Stepped**



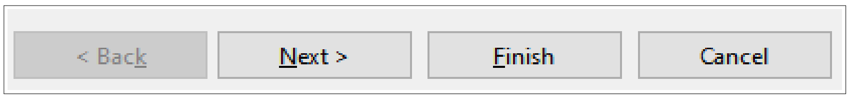
6. Select **properties**



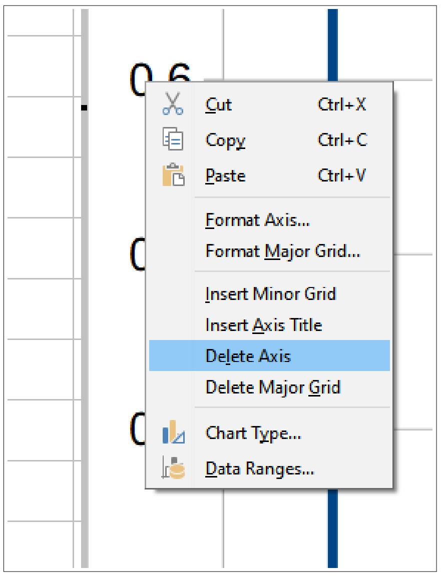
7. Select **Start with horizontal line**


Stepped Lines
Type of Stepping
Start with horizontal line
End with horizontal line
Step at the horizontal mean
Step to the vertical mean
Help
OK
Cancel

8. Click on **Finish**

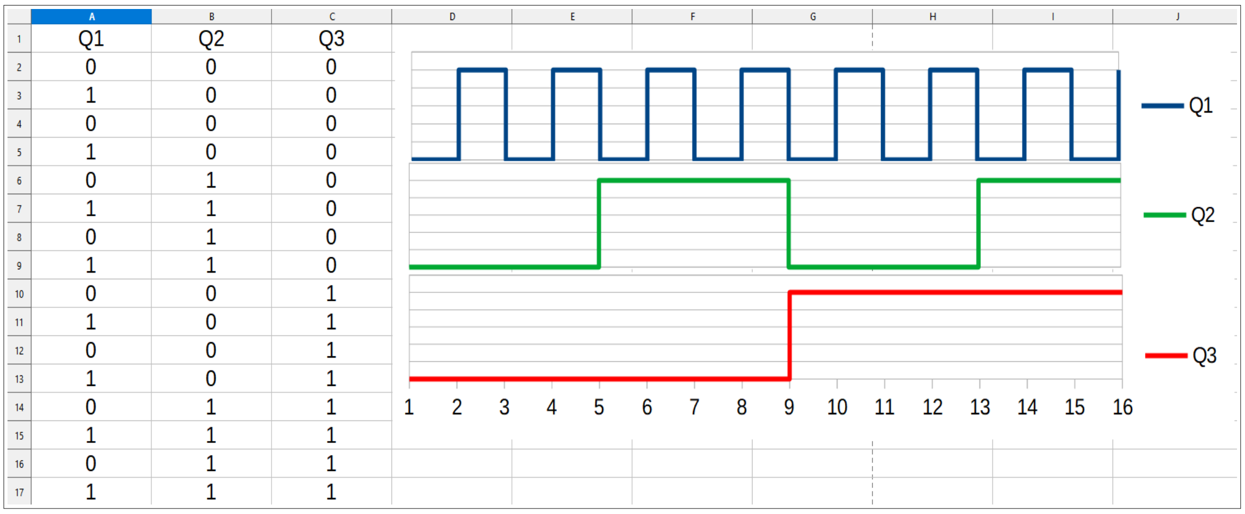


9. Right click on the left side Axis and select **Delete Axis** since we don't really need it.



10. Adjust the size of the diagram and position it as you see fit and repeat the process for the other Outputs. *You can also copy the first diagram and change the **Data Ranges** on the new diagrams to make the process a lot quicker.*

Finished Example



	A	B	C
1	Q1	Q2	Q3
2	0	0	0
3	1	0	0
4	0	0	0
5	1	0	0
6	0	1	0
7	1	1	0
8	0	1	0
9	1	1	0
10	0	0	1
11	1	0	1
12	0	0	1
13	1	0	1
14	0	1	1
15	1	1	1
16	0	1	1
17	1	1	1

ICs Pinouts supported

4xxx Series

4000	**4001**	**4002**	**4006**	**4007**	**4008**	**4009**	**4010**	**40101**	**40102**	**40103**	**40104**	**40105**	**40106**
40107	**40108**	**40109**	**4011**	**4012**	**4013**	**4014**	**4015**	**4016**	**4017**	**4018**	**4019**	**4020**	**4021**
4022	**4023**	**4024**	**4025**	**4026**	**4027**	**4028**	**4029**	**4030**	**4031**	**4032**	**4033**	**4034**	**4035**
4037	**4038**	**4040**	**4041**	**4042**	**4043**	**4044**	**4045**	**4046**	**4047**	**4048**	**4049**	**4050**	**4051**
4052	**4053**	**4054**	**4055**	**4056**	**4057**	**4059**	**4060**	**4062**	**4063**	**4066**	**4067**	**4068**	**4069**
4070	**4071**	**4072**	**4073**	**4075**	**4076**	**4077**	**4078**	**4081**	**4082**	**4085**	**4086**	**4089**	**4093**
4094	**4095**	**4096**	**4097**	**4098**	**4099**	**4502**	**4503**	**4504**	**4508**	**4510**	**4511**	**4512**	**4514**
4516	**4517**	**4518**	**4520**	**4527**	**4532**	**4536**	**4538**	**4541**	**4543**	**4555**	**4556**	**4585**	**4724**

74xx Series

7400	**7401**	**7402**	**7403**	**7404**	**7405**	**7406**	**7407**	**7408**	**7409**	**7410**	**74107**	**74109**	**7411**
74112	**74121**	**74122**	**74123**	**74124**	**74133**	**74136**	**74137**	**74138**	**74139**	**7414**	**74140**	**74145**	**74147**
74148	**74150**	**74151**	**74153**	**74154**	**74155**	**74156**	**74159**	**7416**	**74161**	**74163**	**74164**	**74165**	**74166**
74169	**7417**	**74170**	**74173**	**74174**	**74175**	**74181**	**74182**	**7419**	**74190**	**74191**	**74193**	**74194**	**74195**
7420	**7421**	**74221**	**74240**	**74241**	**74243**	**74244**	**74245**	**74247**	**7425**	**74250**	**74251**	**74253**	**74257**
74258	**74259**	**7426**	**74260**	**74265**	**74266**	**7427**	**74273**	**74276**	**74279**	**74280**	**74283**	**74286**	**74292**
74293	**74294**	**74297**	**74298**	**74299**	**7430**	**7431**	**7432**	**74321**	**74323**	**7433**	**7434**	**74348**	**7435**
74354	**74356**	**74365**	**74366**	**74367**	**74368**	**7437**	**74373**	**74374**	**74375**	**74377**	**74378**	**7438**	**74386**
74390	**74393**	**74395**	**74399**	**7442**	**74423**	**7445**	**74465**	**7447**	**7451**	**74518**	**74520**	**74521**	**74533**
74534	**74540**	**74541**	**74543**	**74561**	**74563**	**74564**	**74569**	**74573**	**74574**	**74575**	**74576**	**74577**	**74580**
74590	**74592**	**74593**	**74594**	**74595**	**74596**	**74597**	**74598**	**74624**	**74628**	**7464**	**74640**	**74641**	**74642**
74645	**74646**	**74647**	**74648**	**74651**	**74652**	**74653**	**74654**	**74657**	**74666**	**74667**	**74669**	**74670**	**74673**
74674	**74679**	**74682**	**74684**	**74686**	**74688**	**74697**	**74699**	**7473**	**7474**	**7475**	**74756**	**74756**	**74760**
74804	**74805**	**74808**	**74821**	**74823**	**74825**	**74827**	**74828**	**74832**	**74841**	**74843**	**7485**	**74857**	**7486**
74867	**74869**	**74870**	**74873**	**74874**	**74876**	**74885**	**7490**	**74990**	**7492**	**74992**	**7493**	**74994**	**7497**

ICs Testing supported

4xxx Series

4000	**4001**	**4002**	**4009**	**4010**	**40106**	**4011**	**4012**	**4013**	**4015**	**4016**	**40161**	**40162**	**4017**
40174	**40175**	**4018**	**4019**	**40192**	**40193**	**4020**	**4021**	**4022**	**4023**	**4024**	**4025**	**4027**	**4028**
4029	**4030**	**4031**	**4040**	**4041**	**4042**	**4043**	**4044**	**4048**	**4049**	**4050**	**4051**	**4053**	**4063**
4066	**4067**	**4068**	**4069**	**4070**	**4071**	**4072**	**4073**	**4075**	**4076**	**4077**	**4078**	**4081**	**4082**
4086	**4093**	**4094**	**4097**	**4098**	**4503**	**4504**	**4510**	**4511**	**4512**	**4518**	**4519**	**4520**	**4529**
4532	**4543**	**4572**											

74xx Series

7400	**7401**	**7402**	**7403**	**7404**	**7405**	**7406**	**7407**	**7408**	**7409**	**7410**	**74107**	**74109**	**7411**
74112	**74113**	**7412**	**74123**	**74125**	**74126**	**7413**	**74132**	**74133**	**74136**	**74137**	**74138**	**74139**	**7414**
74140	**74145**	**74147**	**74148**	**7415**	**74151**	**74153**	**74157**	**74158**	**7416**	**74160**	**74161**	**74162**	**74163**
74164	**74165**	**74166**	**7417**	**74173**	**74174**	**74175**	**7418**	**74182**	**74190**	**74191**	**74192**	**74193**	**74194**
74195	**7420**	**7421**	**7422**	**74237**	**74242**	**74243**	**74244**	**74245**	**7425**	**74251**	**74253**	**74257**	**74258**
74259	**7426**	**74260**	**74266**	**7427**	**74273**	**7428**	**74280**	**74283**	**74292**	**74293**	**74294**	**74298**	**7430**
7432	**74365**	**74366**	**74367**	**74368**	**7437**	**74373**	**74375**	**7438**	**74386**	**74390**	**74393**	**7440**	**7442**
7446	**7447**	**7450**	**7451**	**7455**	**7458**	**74589**	**74595**	**74597**	**7460**	**7461**	**7462**	**7465**	**74682**
7472	**7474**	**7475**	**7485**	**7486**	**7490**								

References and other sources

7400 Series	**Texas Instruments Digital Logic Pocket Data Book (REV.B)**
CMOS 4000 Series	**RCA CMOS Databook 1983**
Where I found the original IC tester	**Instructables page**
GitHub page for the original IC tester	**IC tester GitHub**