Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

ELECTRONIC DEBUGGING

Tips and procedures in debugging…

John Lusher II, P.E., Ph.D.

TOPICS

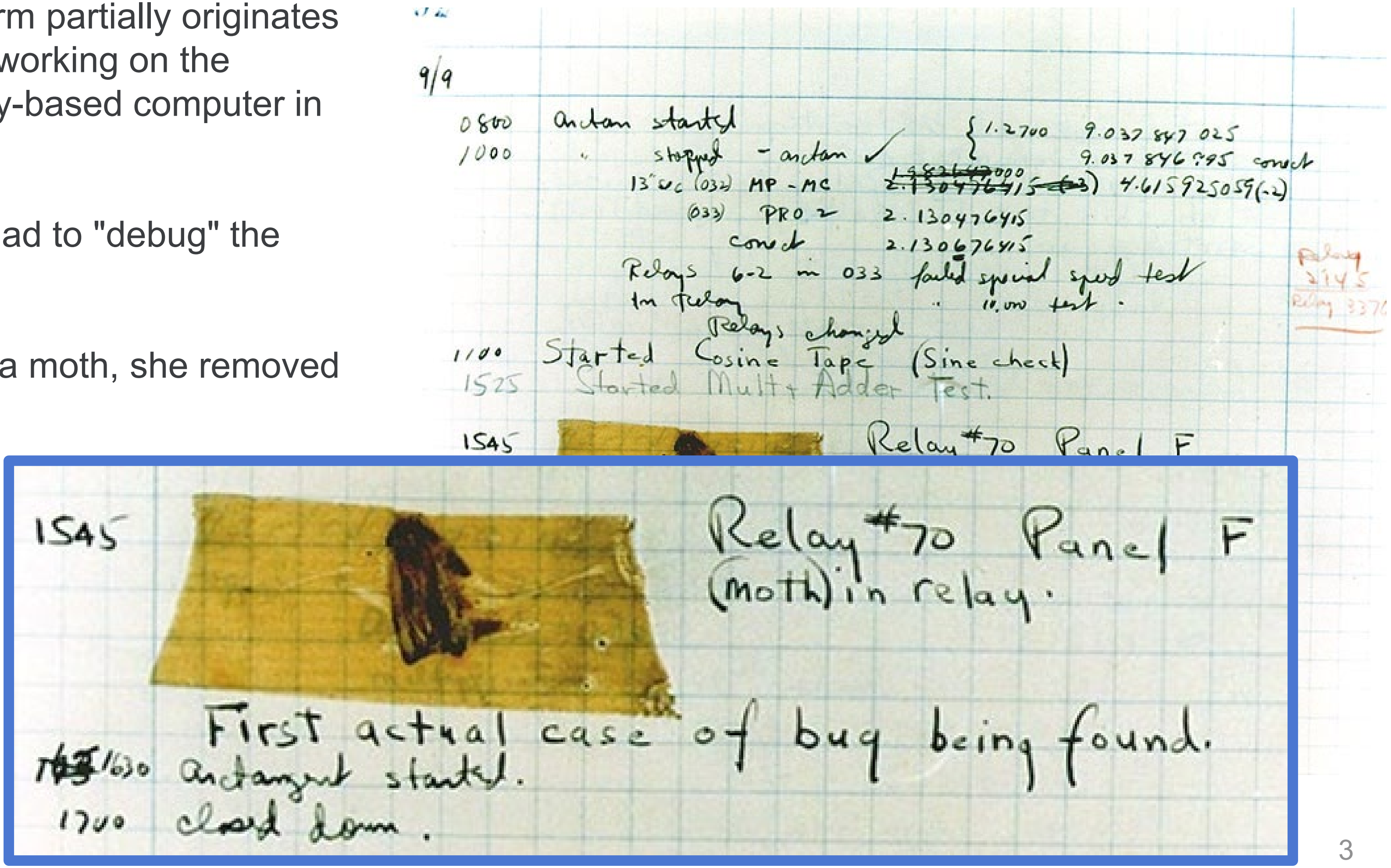
- Introduction
- Planning to debug
- Test and bench equipment
- Oscilloscopes
- The initial steps
- Debugging circuits
- Debugging an embedded system
- Signal noise
- Data acquisition
- Common assembly mistakes
- Common pitfalls

INTRODUCTION

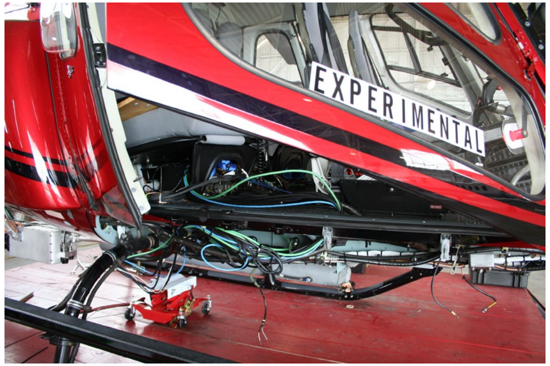
"Debugging" as a term partially originates from Grace Hooper working on the Harvard Mark II relay-based computer in the 1940s.

Where she literally had to "debug" the circuits.

As shown here with a moth, she removed from the computer.



Introduction – Why this seminar?



Introduction – Why this seminar?

- One of the most time-consuming tasks as an engineer is design verification and problem resolution, i.e., debugging.
- Engineers waste many hours debugging a circuit, and regrettably, proper debug techniques are never taught in most colleges.
- This session aims to provide some techniques, tips, and guidance in approaching your next debug session.

Planning to Debug

- Start with the end goal in mind
  - What do you want the circuit or system to do?
  - You need a test plan!
- Do you have a specification and requirements document?
  - If not, you need one, even it is a simple set of notes.
- Do you have an ICD (Interface Control Document)?
  - This is all the I/O to your system in detail.

Planning to Debug

- Creating a test plan
  - List all inputs and outputs. This will include communication protocols used. Verify you have tools or services to interface each I/O interface.
  - List your power needs. Verify that your bench test setup can provide sufficient power in the ranges needed to test.
  - Have a section for the results of each section of the test plan. Even just a simple pass/fail box.
  - Plan to test all the power rails generated onboard. Verify that there are no internal shorts.

Planning to Debug

- Creating a test plan
  - Determine the least complex system to verify to most complex and place testing in this order.

  - Have a short description of each test. This can be short and simple, such as:

    115 kB UART transmit and receive characters to the host computer.

    -or-

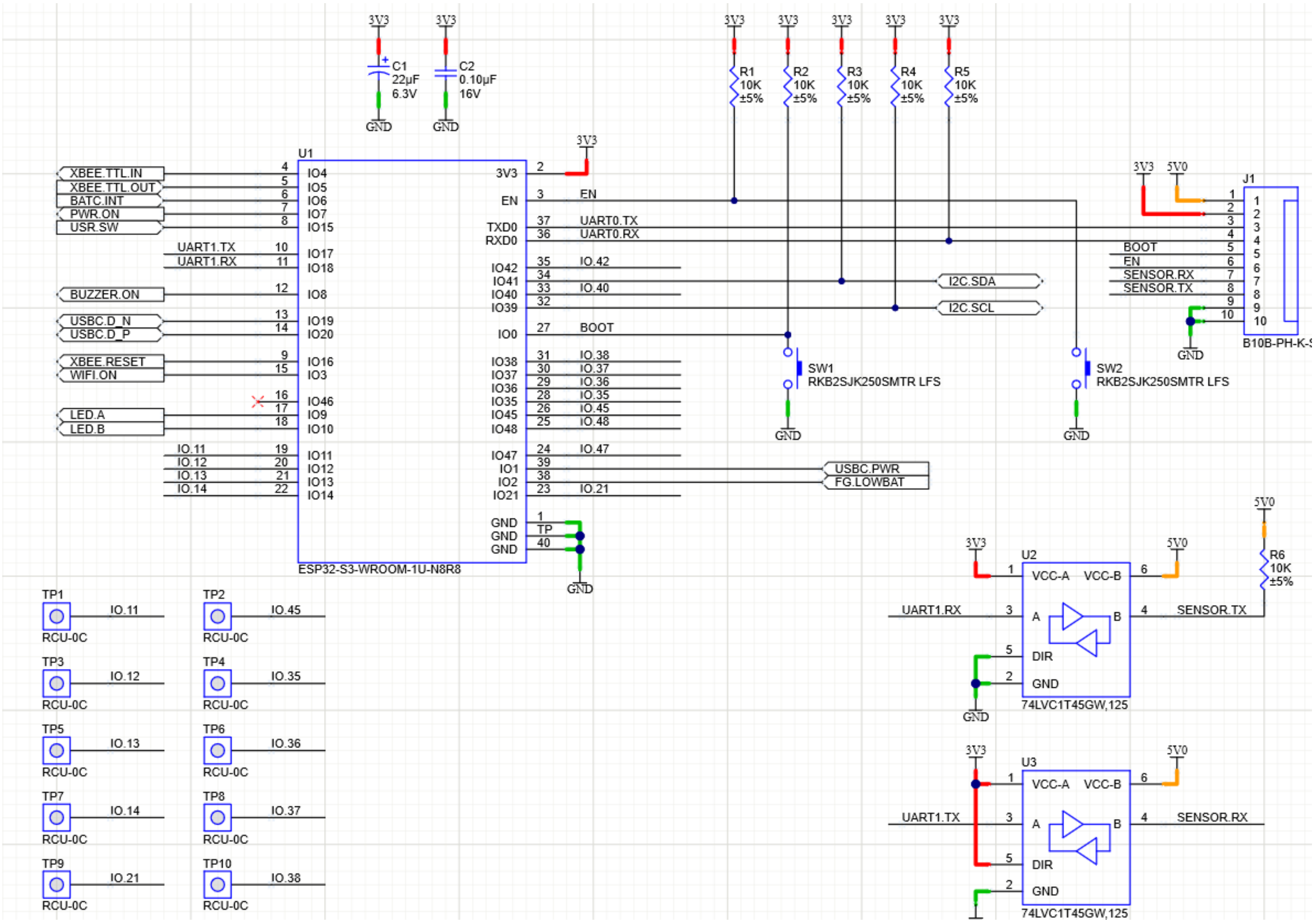
    Ethernet TCP/IP socket communications successful.

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Planning to Debug

- Design to Debug
  - Add test ports or test points to key signals and power rails
    - If you have room, add a test connector that can be used for diagnostics but in mass production can be DNI.
  - Make your design very modular
    - If possible, you can make separate boards of complex circuits to be tested separately.
  - Have a block diagram of your circuit.
  - Make schematics readable
    - You are the one reading them to debug. Use hierarchical designs, ports, and netlabels to reduce the flurry of wires.

PLANNING TO DEBUG



Dwight Look College of
ENGINEERING
TEXAS A&M UNIVERSITY

Planning to Debug

- Datasheets
  - Read each datasheet thoroughly, especially the interfaces
    - Most mistakes happen in not checking the interfaces of an IC!
  - Keep datasheets nearby while debugging.
    - Print out relevant pages, such as pinouts and pin descriptions, and have a diagnostic folder on your bench while working.
  - Refer to application notes and example uses from the manufacturer when possible.

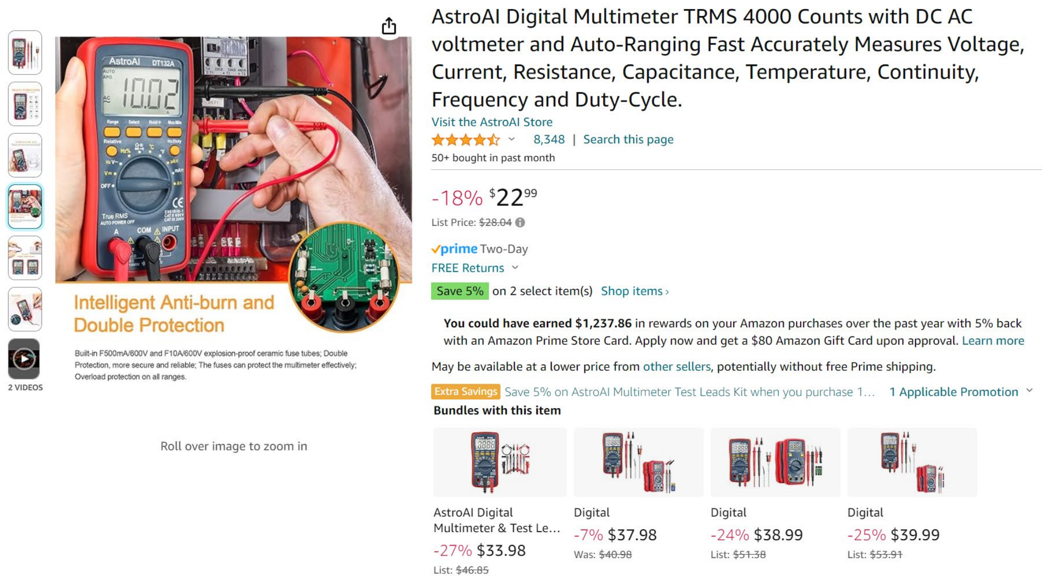
Planning to Debug

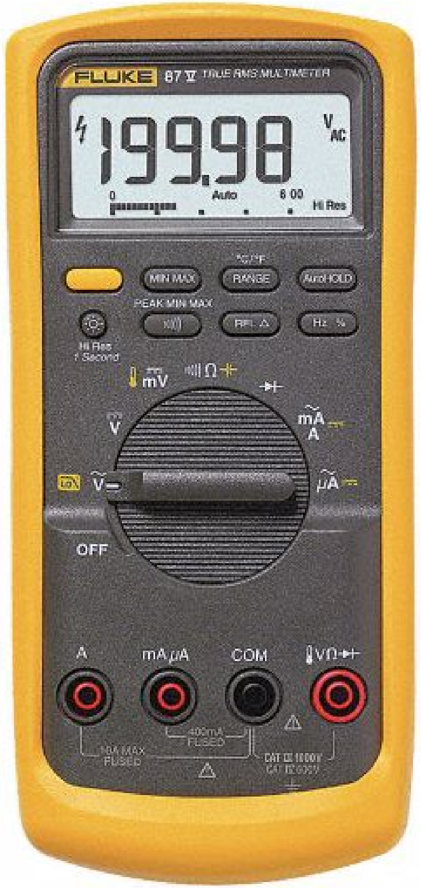
- Evaluation Boards
  - For complex circuits and ICs, purchase an evaluation board.
    - Useful to compare your circuit to a known working example, even if the use case is not the same.
  - Have the evaluation board schematic nearby during debugging as you may find differences between your design and the working example.
  - When your prototype has a circuit that isn't working, sometimes it is a possible "hack" in the evaluation board as a temporary fix to test other circuits while this circuit is redesigned.

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

TEST AND BENCH EQUIPMENT

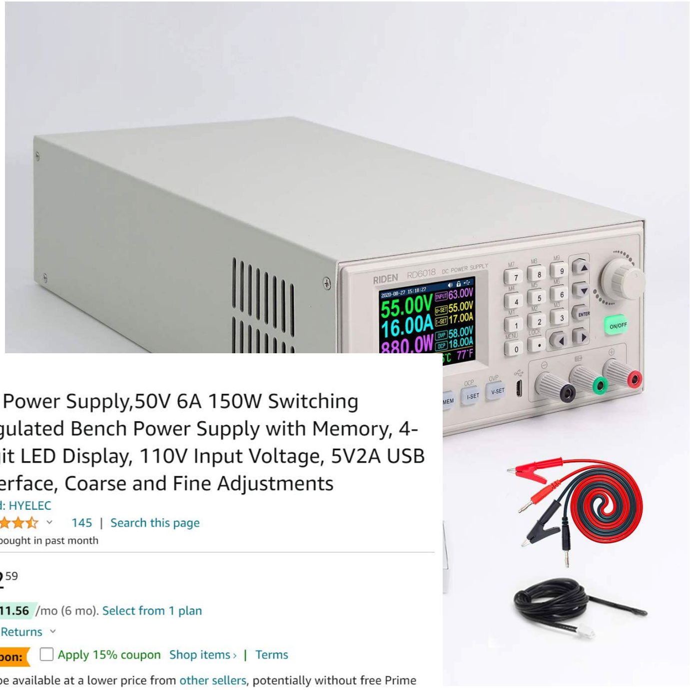
- DMM (Digital Multi-Meter)
  - A must-have!
  - Your instant simple diagnostic tool to most simple problems.
  - It doesn't have to be some high-end Fluke or Keithley.

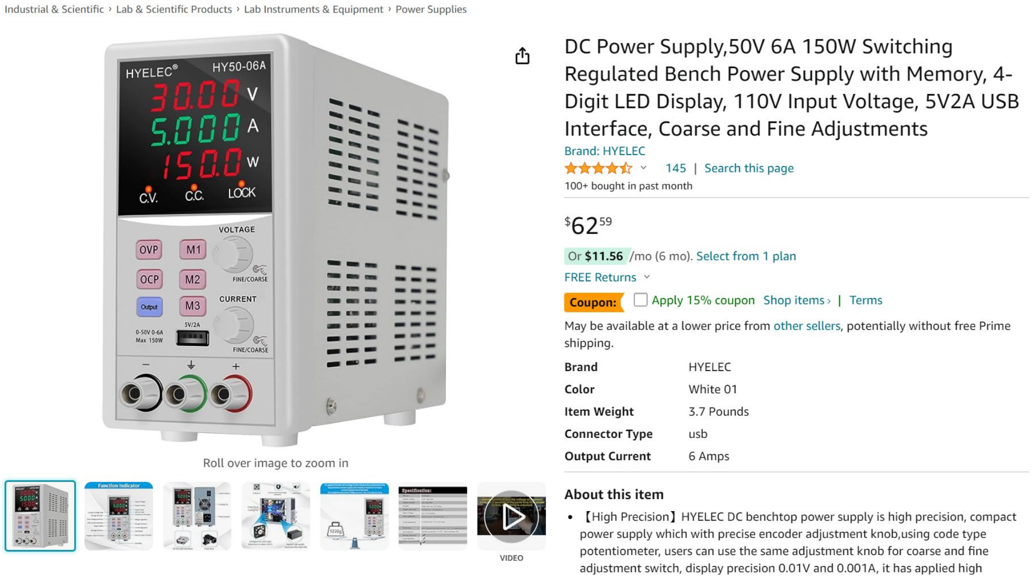




TEST AND BENCH EQUIPMENT

- Power Supplies
  - Multi-Output, Variable voltage, and current level
  - Computer-controlled
    <u>note:</u> this is useful for setting up automatic test fixtures





TEST AND BENCH EQUIPMENT

- Function Generators
  - Useful for the testing of analog and simple digital circuits.

Roll over image to zoom in

Koolertron Upgraded 60MHz DDS Signal Generator Counter,High Precision Dual-Channel Arbitrary Waveform Function Generator Frequency Meter 200MSa/s (60MHz)

Visit the Koolertron Store

906 | Search this page

$159.99

Or $16.08 /mo (12 mo). Select from 1 plan

prime Two-Day

FREE Returns

Size: **60MHz**

15MHz	30MHz	60MHz	60MHz-a	80MHz
$109.99	$138.99	$159.99	$188.99	$205.99
prime	prime	prime	prime	prime

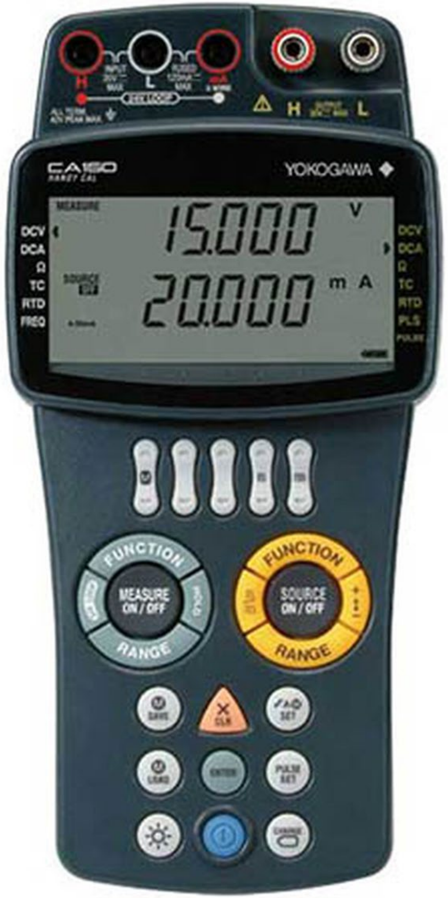
- Arbitrary Waveform Generator adopts large scale FPGA integrated circuit and high speed MCU microprocessor. The internal circuit adopts active crystal oscillator as benchmark. So the signal stability is greatly strengthened.
- Using Dual-channel DDS signal and TTL electric level output to generate precise, stable, low distortion output signal. includes Sine wave, Square wave, Triangle wave, Sawtooth wave, Pulse wave, white noise, user-defined waveform etc. each channel can be independently set the parameters.
- With linear sweep(Max. up to 999.9s) and logarithmic frequency sweep functions.Has a frequency measurement, period measurement, positive and negative pulse width measurement and counting function.
- Storage feature: You can store 99 groups instrument state parameters set by the user, can be called up to Reproduce. Frequency output of Sine wave can be up to 60MHz. 200MSa/s sampling rate. It has 60 positions for saving user-defined waveform. Waveform Length of each one is 2048 and vertical resolution is 14 bits
- This Signal Generator is the ideal instrument for electronic engineering, laboratories, production lines, teaching and scientific research.

Report an issue with this product or seller

TEST AND BENCH EQUIPMENT

- Instrumentation / Sensor Simulation
  - 4-20 mA Current Loops
  - 0-10 V Sensors
  - Strain Gauges
  - RTD / Thermocouples
  - Resistive
  - Signal Pulse

note: some can be computer controlled



Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Test and bench equipment

- Communications
  - RS-232/422/485 to USB cable
  - CAN to USB transceivers
  - I2C / SPI ”sniffers”

TEST AND BENCH EQUIPMENT

Sponsored

Results

Check each product page for other buying options.

Best Seller

Sponsored

YIHUA 926 III 60W Digital Display Soldering Iron Station Kit w 2 Helping Hands, 6 Extra Iron Tips, Lead-Free Solder, Solder Sucker, S/...

5,004

3K+ bought in past month

Limited time deal

$39.99 List: ~~$59.99~~

prime Two-Day

FREE delivery **Thu, Apr 18**

Add to cart

Sponsored

YIHUA 939D+ Digital Soldering Station, 75W Equivalent with Precision Heat Control (392°F to 896°F) and Built-in Transformer. ES...

3,205

1K+ bought in past month

$53.99 List Price: ~~$71.90~~

Exclusive Prime price

prime Two-Day

FREE delivery **Thu, Apr 18**

Add to cart

Sponsored

TOAUTO DS90 Soldering Station-℉ & ℃ Dual Digital Display Soldering Iron Station Kit,90W Soldering Iron,302℉- 842℉ Temperature, Ant...

2,772

200+ bought in past month

$41.99 List Price: ~~$75.99~~

Exclusive Prime price

prime Two-Day

FREE delivery **Thu, Apr 18**

Add to cart

Overall Pick

Weller Digital Soldering Station - WE1010NA

4,274

1K+ bought in past month

$115.00 List: ~~$152.99~~

Or $38.34/month for 3 months (no fees or interest)

prime Two-Day

FREE delivery **Thu, Apr 18**

Add to cart

More Buying Choices

$101.60 (24 used & new offers)

Best Seller

YIHUA 926 III 60W Digital Display Soldering Iron Station Kit w 2 Helping Hands, 6 Extra Iron Tips, Lead-Free Solder, Solder Sucker, S/...

5,004

3K+ bought in past month

Limited time deal

$39.99 List: ~~$59.99~~

prime Two-Day

FREE delivery **Thu, Apr 18**

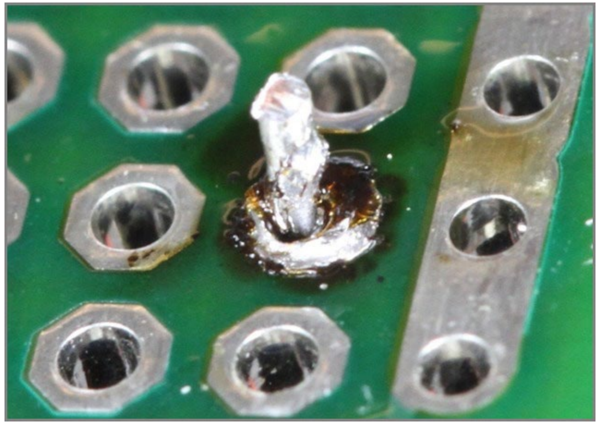
Add to cart

More Buying Choices

$39.59 (3 used & new offers)

Common Assembly Mistakes

- Are you using a breadboard?
  - Don't…This is the cause of more frustration than it is helpful.
- Is your board clean?
  - Flux remaining on a board can be conductive and sometimes corrosive. At a minimum, use a flux removal spray, or better, clean your board fully.
- Examine your board for cold solder joints.
- Examine your board for solder shorts.



Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

Oscilloscopes

- You likely were not taught how to use an oscilloscope to its full potential in college!

- **<u>This is BY FAR your #1 tool in debugging electronic circuits!</u>**

- Investing in a good scope is worth the money as this will save you and your company many hours of engineering time.

- The key to this is to:

<u>LEARN YOUR SCOPE!</u>

OSCILLOSCOPES

- The #1 mistake in using a modern oscilloscope is using…
- You need to know what signal you are trying to analyze
  - Know the following:
    - Is it AC coupled, or DC coupled?
      This determines the input mode.
    - What is the expected voltage range (min to max)?
      This determines the channel's voltage scale.
    - What is the frequency range or period?
      This determines what the time base should be set to.
    - Is it a digital signal, a simple sinusoid, or a complex waveform?
      This determines what you will set the trigger modes to and what to look for.

OSCILLOSCOPES

- Modern oscilloscopes have many features to help you debug a circuit, such as:
  - Multiple analog or digital channels (4 analog and 16 digital is common)
  - Single Acquisition and Run/Stop modes
  - Bus protocol decoding, such as UART, SPI, I2C, Ethernet, 1553, Audio, CAN, etc…
  - Math functions
  - Spectrum analyzers
  - Glitch detection
  - Cursors (useful for measuring time/voltages of signals, and comparing)
  - Wave inspection (signal zoom)
  - Waveform storage (export to CSV file), and screenshots

OSCILLOSCOPES

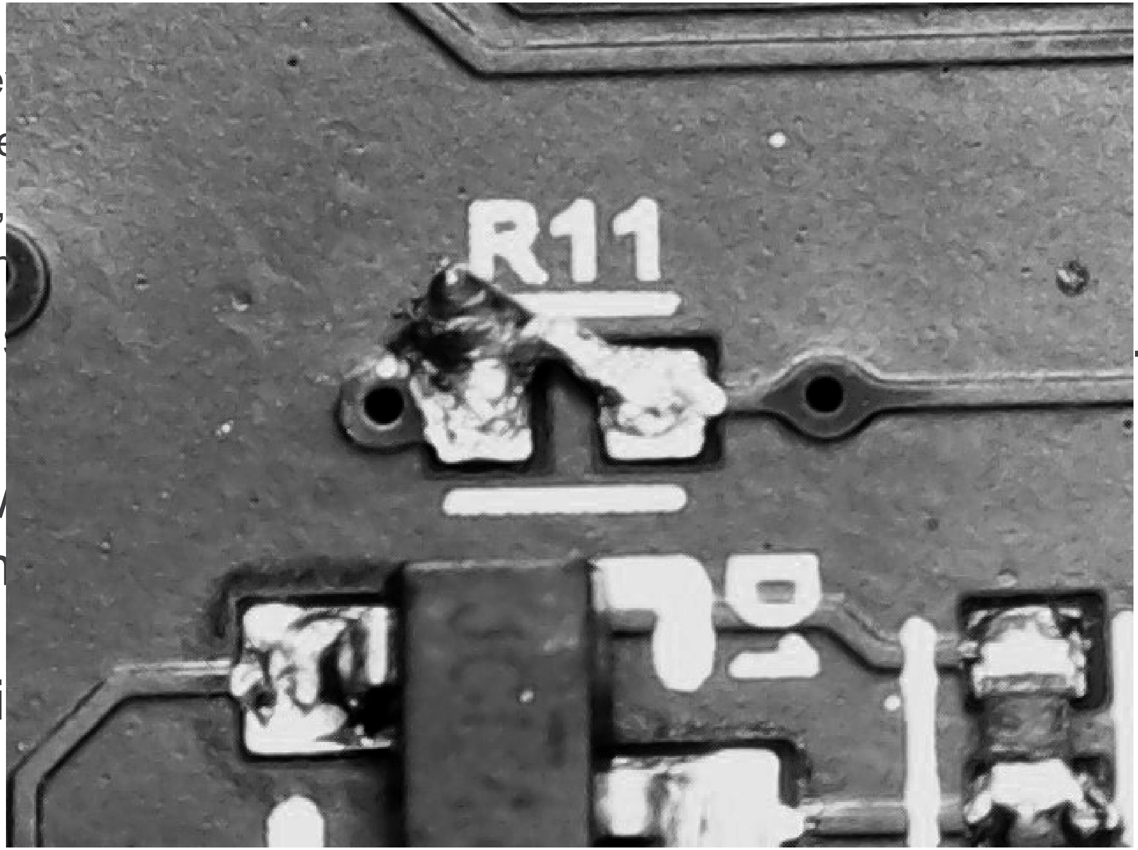
- Various Trigger Modes:
  - Automatic – just rolls and if there is a trigger is syncs on it.
  - Normal – Rising edge, Falling edge, or both, of signal at a set voltage.
  - Sequence – One trigger mode, then another
  - Bus – Look for bus input, like a serial string, or I2C ACK
  - Pulse width – Triggers on a pulse width range
  - Video – Trigger on video sync
  - Rise / Fall times

Dwight Look College of ENGINEERING TEXAS A&M UNIVERSITY

THE INITIAL STEPS

- First, examine the assembled board for any defects and assembly issues.
  - Are parts where the
  - Are there any shorte
  - Are parts damaged,
  - Just because you "h
- Check for shorts u .
- Connect input pow power-up with a lin

  See if you hit the li



The Initial Steps

- Then do the "magic smoke test."
  - Did you release any of the "magic smoke" from the board?
    - BTW, we all do it. Don't fret if you do. It will happen.

- No smoke? Are you sure?

- Then, it's time to move on!

The Initial Steps

- Examine for any parts with thermal issues (i.e., hot parts)
  - This could be due to parts being wired incorrectly!
  - Thermal cameras are great for this. Else, be careful not to burn a finger.

- Check all rail voltages
  - Do you have the expected voltage?
  - Is the noise level acceptable?
  - Any unexpected spikes?

- Check voltages for communications devices, such as:
  - RS-232, LVDS, mSATA, PCIe, etc…

- Now, we are ready to connect signals and verify I/O or attempt to do an embedded system debug.

Debugging Analog Circuits

- Check power rails as all analog circuits, especially op-amps.
  - Do you have filter capacitors at each op-amp?
  - Is the op-amp hot?
    - If so, check if power pins on op-amp are correct.
    - Is there too much current going to the load?

- If you inject a signal into the analog circuit, but do not get the expected output, check the following:
  - Check the configuration of the op-amp (feedback mode)
  - For a high-speed signal, verify that the op-amp's gain-bandwidth product is sufficiently high enough. Also, check the slew rate of the op-amp.

- Simulate design in SPICE to ensure it isn't a design issue.

DEBUGGING CIRCUITS

- If an IC or transistor is failing, check the following:
  - Is it connected as per the datasheet? Check the footprint!
  - Is the power connected correctly?
    - Do you have power bypass capacitors?
  - Do you have any pins floating that should have an input signal?
  - Are the outputs connected to an appropriate load?
  - Is the IC installed correction (direction)?
  - Is the IC/transistor damaged? Such as a crack or “pit” on the case.
  - Are you using the IC/transistor in a typical mode of operation? If so, check your circuit to such a use case. If not, verify your design.

Debugging an Embedded System

- Work with the evaluation board for the MCU you are using first!
- Get the tool-chain fully working (IDE, compiler, linker, and debugger).
- For your system, start with “hello-world” or a star thrower (i.e., ***).
  - This is by far the best test because it proves so much of your system is working, such as power, CPU, clock oscillator, programming connections, tool-chain, basic I/O, and simple communications.

Debugging an Embedded System

```c
//-------------------------------------------------------------------------------------------------------------------------------------
// Definitions for communications
//-------------------------------------------------------------------------------------------------------------------------------------



//-------------------------------------------------------------------------------------------------------------------------------------
//    Routine: write_io_pca9535
//     Inputs: I2C PCA9535A address, Port #, and I/O State
//    Outputs: none
//    Purpose: Set output on I/O expander
//-------------------------------------------------------------------------------------------------------------------------------------
void write_io_pca9535 (unsigned char i2c_address, unsigned char port,unsigned char port_data)
{
                                                                    //---------------------------------------------------------------
    unsigned char i2cData[3], index;                                // Data for I2C communications, Byte index counter
                                                                    //---------------------------------------------------------------

                                                                    //---------------------------------------------------------------
                                                                    // Send Data to EEPROM to read one location
    i2c_address <<= 1;                                              // Shift 1-bit (write is zero)
    i2cData[0] = i2c_address;                                       // Device address and WR Command
    if (port) i2cData[1] = PCA9535A_WRITE_REG_1;                    // Select command based on port #
    else i2cData[1] = PCA9535A_WRITE_REG_0;                         //
    i2cData[2] = port_data;                                         // Data
                                                                    //
    StartI2C1();                                                    // Send the Start Bit
    IdleI2C1();                                                     // Wait to complete
                                                                    //
    for (index = 0; index < 3; index++)                             // Send the address to write to the I2C device
    {                                                               //
        MasterWriteI2C1(i2cData[index]);                            // Write next byte over I2C
        IdleI2C1();                                                 // Wait to complete
        ClearWDT();                                                 // Service the WDT
        if( I2C1STATbits.ACKSTAT ) break;                           // ACKSTAT is 0 when slave acknowledge,
                                                                    // if 1 then slave has not acknowledge the data.
    }                                                               //
                                                                    //
    StopI2C1();                                                     // Send the Stop condition
    IdleI2C1();                                                     // Wait to complete
                                                                    //---------------------------------------------------------------
}

```

DEBUGGING AN EMBEDDED SYSTEM

- Don't be afraid to learn a new processor.

  Sometimes trying to make an older MCU do tasks that are to be performed by a modern MCU makes code more complicated and more likely to fail.

- Use a debugger (i.e., an ICD)!
  - Enable breakpoints (in your C code directly)
  - Watch variables
  - Inspect and alter memory locations

Debugging an Embedded System

- Decide if you use a bare-metal system or have an embedded OS, such as open embedded Linux.
- In design, make sure you include more I/O and memory resources that you are expecting.
- Initially disable the WDT in testing. However, once completed making sure the WDT is enabled.

Debugging an Embedded System

- Test each module separately
  - Especially your peripheral driver code, such as ADCs, DACs, timers, etc.
  - Verify the ICD for each module.
  - Avoid global variables where you can. The module should be independent.

- Use a logic analyzer/oscilloscope in conjunction with your debugger to verify timing externally. Especially in bare-metal applications.

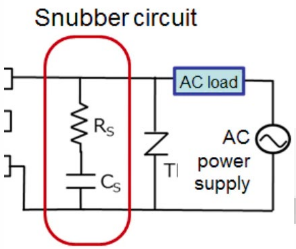
- If you run into issues, check the manufacturer's forums, example projects, any previous projects, etc.

Signal Noise

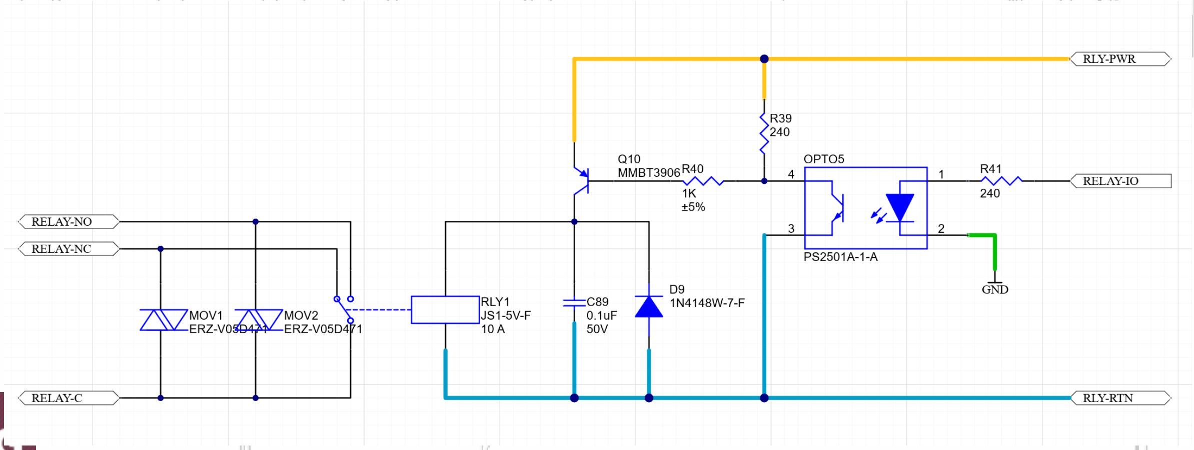
- Do you have power filters for every IC?
- Did you have any ground loops, i.e., multiple paths to ground a signal?
- Consider ferrite beads to filter out undesired frequencies in power rails or signals routes.
- Add filters to specific input/output signals of the board.
- Add signal buffers and filters to your board.

Signal Noise

- Do you have any electro-mechanical relays on your PCB, if so:
  - Consider a snubber circuit on if you are switching high voltages.



  - Have a diode and capacitor circuit to collapse the magnetic field
  - Use optoisolators if needed to separate power and analog systems.



Data Acquisition

- What is the highest frequency of the signal you are acquiring?
- What is your ADC sample rate?
  - This should be a minimum of twice the highest signal you are acquiring, preferably about 10X.
- Make sure you have an anti-aliasing filter for every signal you are acquiring (i.e., a low-pass filter).
- What is the minimum resolution you want to be accurate to? Make sure to consider the bits that are guaranteed to be accurate (MSB) from your ADC manufacturer.

Data Acquisition

- Is the bus your ADC is connected to supply the necessary bandwidth for the sample rate required?
- Can you MCU maintain the sample rate by the bus method? Are you using DMAs, or a built-in peripheral?
- How is your sample rate determined? ADC clock, MCU clock?
- Use an accurate voltage reference. A poor reference, such as using a power rail, will lead to a noisy or low accuracy data acquisition.

DATA ACQUISITION

- If using multiple channels (via a multiplexer), you must consider the channel switch over time in mux. For slow signals, this is not an issue. For high-speed, this becomes an issue.
- Do you have sufficient memory for storing all of the data you are acquiring?
- Is there sufficient DSP/MCU processor cycles to process collected data for the desired signal process?

Common Pitfalls

- Rushing the debug process
  - Debugging takes time, skipping steps will slow it down more.
- Making assumptions
  - Don’t just assume that since this “worked” for another project that it will just “work” for your new project. Sometimes there are subtle differences that cause major problems.
  - Trusting “online” part libraires or example code. Must of that is not truly verified.
  - Datasheets can sometime be wrong!
- The #1 pitfall for most engineers…

Not asking for help!