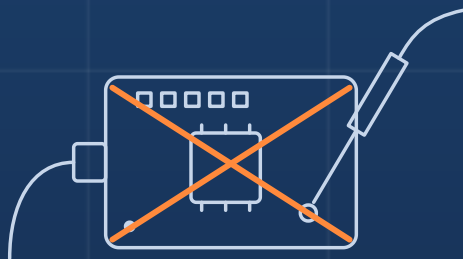


FROM PROTOTYPE TO PRODUCTION

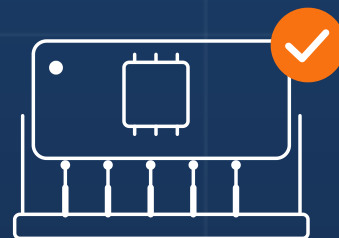
PROGRAMMING AND TESTING AT SCALE

How to program and test every board you build, from your first prototype to high-volume production, and what to design in from day one.



BENCH PROTOTYPE
FLASHED BY HAND

SCALE →



PRODUCTION RUN
JIG FLASHED AND TESTED

PREPARED BY
JOHN TEEL

DOC NO.
PRD-04

REV
A

SOURCE
[PREDICTABLEDESIGNS.COM](https://predictabledesigns.com)

Programming and Testing at Scale

How to program and test every board you build, from your first prototype to high-volume production.

Programming and testing a prototype on your bench is easy. Doing it for a hundred or ten thousand units is a different problem, and if you have not planned for it, it holds up production, blows your budget, or lets hidden faults slip through to your customers. Testing in particular is one of the most critical steps in bringing a product to market, and one of the most overlooked.

This cheat sheet follows the same four stages as my two videos on the topic, from the prototype on your bench to small batch builds, high-volume production, and the units already in the field. For each stage you get the programming methods, the testing methods, and the design choices your board needs to support them. Most of those choices belong on your very first schematic, because adding them after the board is in production costs you thousands.

The Cheat Sheet at a Glance

#	STAGE	HOW YOU PROGRAM	HOW YOU TEST	DESIGN IT IN
01	Prototype	Bootloader over USB, or an ST-Link or J-Link	Multimeter, scope, and logic analyzer	Programming and test access on rev 1
02	Small batch	Pogo pin jig or pre-programmed chips	Bed-of-nails jig running a test script	Labeled, evenly spaced test pads
03	High volume	Programming service or in-circuit on the line	AOI, ICT, FCT, or flying probe	Firmware test mode and ESD protection
04	In the field	USB DFU or over-the-air updates	Burn-in, sampling, and return analysis	A backup way to reflash

Every method above is explained in full on its stage page, along with the trade-offs that decide which one fits your product.

01

Prototyping: Program and Test by Hand

A HANDFUL OF UNITS

HOW YOU PROGRAM IT

- Most early prototypes run on a development board like an Arduino, STM32 Nucleo, or ESP32 DevKit. These boards have a USB interface and a preloaded bootloader, so you can flash new firmware straight from your computer without any special tools.
- A bootloader is a small piece of code that runs as soon as the microcontroller powers on. It decides whether to wait for new firmware or jump straight into your application, and many MCUs include a factory bootloader that listens on USB, UART, or SPI.
- Once you move to a custom board with a bare MCU, you have to decide whether to rely on that built-in bootloader or use an external programmer.

PROGRAMMING OPTIONS

THE METHOD	WHEN IT FITS
Direct USB through the built-in bootloader	This is the simplest option when the chip supports it. Some STM32 models support the official USB DFU protocol, while some ESP32 models use an Espressif bootloader that is not DFU compliant.
External programmer over SWD or JTAG	An ST-Link or J-Link is more robust for development, especially for debugging or for chips without a convenient bootloader. Expect to pay a few hundred dollars, although cheaper knock-offs are usually available.

HOW YOU TEST IT

- At this stage you test everything by hand with a multimeter, an oscilloscope, and maybe a logic analyzer. The only goal is to validate the core functionality, and for small volumes that is fine.
- Just like you need access to program the board, you need access to measure and test it. That means exposing critical signals, power rails, and communication lines with test pads or headers.
- Mechanical testing belongs here too. Check that the enclosure fits and the connectors align, because catching those issues now saves you big headaches later.

Build in access from the very first revision

Your board needs access to the programming interface from day one. That could be a standard header or just test pads, but it has to be there on the first revision, along with test pads on every signal you will need to measure.

02

Small Batch: Program with a Pogo Pin Jig

DOZENS TO HUNDREDS

HOW YOU PROGRAM IT

- Once you are building units for beta testers, crowdfunding backers, or a pilot run, flashing each board by hand becomes a bottleneck.
- Plugging and unplugging every board wastes time, and soldering a header onto every board just for programming adds cost to each unit.

PROGRAMMING OPTIONS

THE METHOD	WHEN IT FITS
Pogo pin programming jig	A simple fixture with pogo pins that contact the programming pads on the PCB. You drop in the board, close the lid, and a script flashes the firmware.
Pre-programmed chips	A chip socket programmer or a small-scale programming service loads the MCU before assembly. This only makes sense when you are confident in the firmware version, because a change means reprogramming or scrapping those chips.

A small thing that saves massive headaches

Always include a readable firmware version number in your software, accessible over serial or USB. Once units are going out in batches, it is how you figure out which units have which firmware loaded.

02

Small Batch: Test with a Bed of Nails

DOZENS TO HUNDREDS

HOW YOU TEST IT

- Manual testing does not scale to dozens or hundreds of units, so this is where bed-of-nails testing comes in. A custom fixture presses spring-loaded pogo pins against the test pads on your board.
- With the board in place, the jig powers it up and runs automated tests on voltages, I2C communication, sensor outputs, LEDs, buttons, and more.
- The key is automation. A microcontroller or PC interface runs a test script, logs the results, and tells the operator whether to accept or reject the board.

DESIGN IT IN

THE CHECK	WHY IT MATTERS
Add test pads for all key signals	The fixture can only test the signals its pogo pins can reach.
Label them clearly in your design files	Whoever builds the fixture needs to know exactly which pad carries which signal.
Space them consistently	Consistent spacing is what lets the pogo pins line up with every pad.
Keep tall components away from test points	Tall parts next to a pad get in the way of the pins pressing down on it.

Program and test in one shot

The test jig and the programming jig both work through pogo pins, so you can combine them. The fixture flashes the firmware and then tests the board without it ever leaving the jig.

03

High Volume: Program on the Production Line

THOUSANDS AND UP

HOW YOU PROGRAM IT

- At high volume you have two main paths, which are pre-programmed chips or in-circuit programming during production.
- Pre-programmed chips are generally the easiest option. A programming service such as Arrow, Avnet, a service bureau, or the chip maker loads your final binary before the chips ship to your contract manufacturer.

PROGRAMMING OPTIONS

THE METHOD	THE STRENGTH	THE TRADE-OFF
Pre-programmed chips	It is fast and scalable, and the factory just places the chip and goes. The service can also label the chips and track firmware versions.	Your firmware must be final, which means tested, versioned, and ideally protected. You also have to coordinate carefully with both the contract manufacturer and the programming service.
In-circuit programming on the line	A factory jig with pogo pins powers up each assembled board and flashes it automatically. You can flash the latest firmware version and run board-level tests in the same step.	It is slower than pre-programming, and your board needs dedicated test points that line up with the jig.

DESIGN IT IN

THE CHECK	WHY IT MATTERS
Add a test mode and test hooks	The test station can run self-checks, read internal data, and trigger hardware behavior on command.
Report results over serial or LED blink codes	The fixture can read the result without your product needing a display.
Add ESD protection and filtering on test lines	Programming and test lines are exposed to the fixture during manufacturing.

How to choose between them

If your firmware is locked and you want the simplest assembly line, use pre-programmed chips. If your firmware is still evolving, or you want to combine flashing with functional testing, program in-circuit on the line.

03

High Volume: Test on the Production Line

THOUSANDS AND UP

HOW YOU TEST IT

- When you are building thousands or even millions of units, testing becomes part of the production line itself, with automated test stations at different stages.
- Most factories use ICT to catch build problems early and FCT to make sure everything actually works, but depending on your budget and volume you might choose just one. Factories often combine all of these techniques based on cost, throughput, and how complex your product is.

TESTING OPTIONS

THE TEST	WHAT IT DOES	STRENGTHS AND LIMITS
Automated Optical Inspection (AOI)	Cameras and image recognition check solder joints, part orientation, polarity, and component placement.	It is fast and contactless, and it catches visual defects before boards move on to more expensive test steps.
In-Circuit Test (ICT)	It checks for shorts, opens, and component values before the board is fully assembled.	It is fast and reliable, but the custom bed-of-nails fixture can be expensive and slow to develop, especially for complex or low-volume boards.
Functional Test (FCT)	It powers up the board and simulates real-world use to confirm that everything works as expected.	It is flexible and tests the actual firmware and peripherals, but it can miss subtle hardware issues like a missing pull-up resistor or the wrong capacitor value.
Flying probe	Two to four moving probes contact test points and vias one at a time to measure continuity, resistance, voltage, and diode drops.	It needs no fixture, but it is slower than ICT. It fits low to mid-volume runs, high-mix production, and design validation before you build a full fixture.

Let the fixture watch the LEDs

A simple photosensor placed near an LED can confirm whether it lights up when it is supposed to. That gives you an automated pass or fail check without needing a person to watch the board.

04

In the Field: Updates, Reliability, and Returns

AFTER LAUNCH

HOW YOU UPDATE IT

- After your product ships, the question becomes how to update the firmware in the field, whether to fix bugs, add features, or patch security holes.
- Even if you never plan to update the firmware, you still need a way to recover a locked-up device or apply a critical patch.

UPDATE OPTIONS

THE METHOD	WHAT IT TAKES
USB with DFU mode or a custom bootloader	This works well if your product has a USB port, and end users can update the firmware through an app or tool.
Over-the-air (OTA) updates	This works over Wi-Fi, Bluetooth, or cellular, but it is a full subsystem rather than a free feature. Its bootloader has to manage firmware partitions, validate digital signatures, and recover from failed updates.
A backup way to reflash	Even exposed test pads inside the enclosure are enough to bring a non-functional unit back.

HOW YOU KEEP TESTING

THE TEST	WHAT IT CATCHES
Burn-in testing	It catches early-life failures before customers find them.
Environmental testing	Thermal cycling, vibration, and humidity show how the product holds up in real conditions.
EMI pre-compliance testing	It catches noise issues before you pay for certification.
Sample-based testing	Testing one out of every 10 or 100 units keeps an eye on long-term quality.
Return analysis	When a product fails in the field, you inspect it, log the failure, and figure out what went wrong. That is how you improve over time.

Is OTA worth it for your product?

For many simple products, OTA is overkill. If your product is connected and in the field long-term, it can add massive value, as long as you budget for the bootloader work it requires.

Design It In from the First Schematic

Every method in this cheat sheet depends on choices you make on the board itself. Run your design through these six checks before you order your first production boards.

- 1 Programming access**
 The programming interface is on a standard header or test pads, and it has been there since the first revision.
- 2 Test access**
 Every critical signal, power rail, and communication line has a test pad that is labeled, consistently spaced, and clear of tall components.
- 3 Traceability**
 The firmware reports a readable version number over serial or USB, so you always know which units have which firmware.
- 4 Factory hooks**
 The firmware has a test mode, reports results over serial or LED blink codes, and includes hooks that trigger hardware behavior.
- 5 Protection**
 Programming and test lines that are exposed during manufacturing have ESD protection and filtering.
- 6 Recovery**
 There is always a backup way to reflash the firmware, even if it is only test pads inside the enclosure.

Get a second set of eyes before your production run.

Test point placement, programming access, and production test plans are an everyday conversation inside the Hardware Academy, where you can get feedback on your specific product from me and lots of other experts.

PredictableDesigns.com/academy