The $10M Error

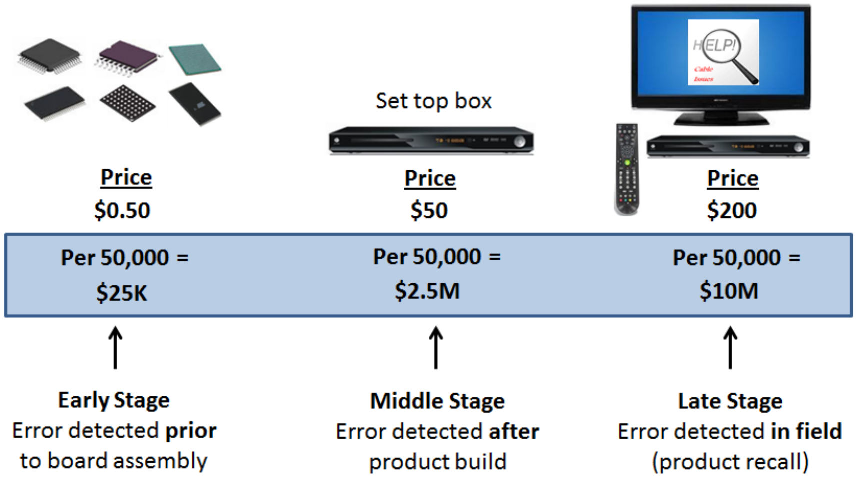
Building an Integrated Approach to Device Programming

A history of device programming, alternative programming methods, and a look at the latest approach: integrated inline programming.

The $10M Error: *Building an Integrated Approach to Device Programming*

A production operator at a high volume consumer electronics manufacturer selected the wrong data file to program into a new model. Before the operator's mistake was discovered, the manufacturer had built thousands of products containing the wrong software and shipped them to customers where they failed during activation. All the faulty products had to be identified and withdrawn from distribution channels. That one mistake, not counting embarrassment and loss of good will, cost the manufacturer over 10 million dollars. Depending on where an error is detected, costs can range from $25K to over $10M.

Costs of error detection at early, middle, and late stages in the production process

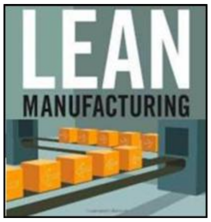


Humans are not perfect. The more complex the task, the more likelihood there is for error, and manufacturing lines are no exception. In today's complex manufacturing environment, getting accurate information into programmable devices requires that people and processes get every step right—from the software engineer who creates the data file and associates it to a build recipe, to the operator who selects that recipe and programs the device. Unfortunately, major forces are working against success.

Historical Roadmap of Device Programming

Over time, the types and severity of programming errors have changed. In the **1970s**, Original Equipment Manufacturers (OEMs) designed and built their own products in-house and sold those products locally. Typically, devices were programmed offline and in-house, conditions which met the OEM's delivery requirements. Any errors in programming were usually caught quickly and affected fewer devices.

In the **1980s**, the move toward outsourced manufacturing began to change the competitive landscape. Electronic Manufacturing Service (EMS) companies set up shop in low cost labor regions. OEMs outsourced their production to EMS companies to remain competitive on cost and delivery. OEMs began to lose control over manufacturing processes. Visibility into programming errors was diminished.



In the **1990s**, companies began adopting the Toyota model of Lean Manufacturing, building product just-in-time and embracing the notion of doing more with less. Designs could now be changed right up to and even during production run time. By the late 1990s companies had turned to programming inline at test, a just-in-time solution well suited to the small file sizes typical at that time.

CONVERGENCE

In the **2000s**, open architecture designs drove new applications for tablet PCs, automobiles, and smartphones, all of which depend on higher density Flash Memory (including NAND and eMMC) to accommodate increased file sizes. As file size grew, so did the time it took to program inline at test, and bottlenecks at test forced production managers and test engineers to look for alternative methods of programming.

Process Control

Factory Integration Software

In the **2010s**, companies continue integrating the factory floor with the business management software—the Enterprise Resource Planning (ERP) software—to improve efficiencies with just-in-time manufacturing. Benefits of integrated programming include remote control of factory equipment, real-time performance feedback, and auditing capabilities used for traceability and quality.

Major Programming Challenges

Human Error

A major challenge is the risk of operators selecting the wrong data file for a product build. These errors occur because no link exists between the software repository and the build recipe. Without a link, operators can program thousands of products using the wrong data, an error potentially costing millions of dollars.

Last Minute Code Changes

A second major challenge occurs when software code changes. If out of date code is programmed into thousands of devices, those devices are unusable. Scrapping is prohibitively expensive since devices can cost several dollars and batch sizes are often in the thousands. Returning pre-programmed devices to be unpackaged, erased, reprogrammed, verified, and repackaged can cost double the original programming price per device. Depending on how quickly the company needs the reprogrammed devices, additional fees may be assessed to expedite and transport reprogrammed parts back to the assembly floor.

Inventory Float

Challenges come in many forms: human error, code changes, inventory float, and IP security.

Devices programmed offline typically sit in factory inventory awaiting placement on PCBs, adding no value and representing sunk cost. Forecast uncertainties can increase inventory float. An example helps illustrate. If current forecast calls for 500K devices for a product build, blank devices are purchased (at $2 each), and programmed (at $.30 each). If only 350K devices are needed, the manufacturer has 150K unusable programmed devices. If they can be reprogrammed, the manufacturer reprograms 150K parts (at double the original programming cost, or $.60 each) for a total programming cost of $135K ($45K for original programming and $90K for reprogramming). If the devices can’t be reprogrammed, 150K devices are scrapped, costing $345K ($300K for devices and $45K for original programming).

Loss of Intellectual Property

Programmed devices in inventory expose the manufacturer to possible intellectual property (IP) theft because every programmed device contains the critical information to make a product operate. When thousands of programmed devices are in factory inventory, the theft of even a single device can be catastrophic. Storage in a secured area is costly and is no guarantee against theft because security systems can be circumvented. Offshore production in regions where IP protection laws are loose further raises the risks of theft.

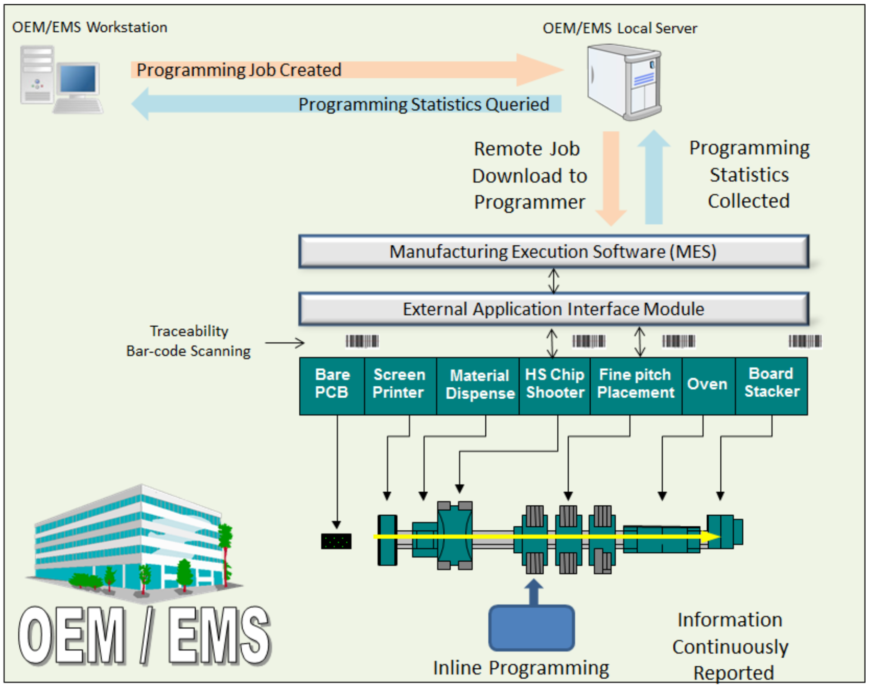
An Integrated Solution

As feature-rich products demand more code, the ability to program as rapidly as possible becomes critical. Programming large files at test is no longer cost effective, or needs to be complemented by an integrated inline programming solution.

An ideal inline solution includes a job management utility to create, encrypt, and transfer programming jobs. The job management utility must have the ability to trace information for each device programmed, including the job name, time and date, error information, and algorithm and socket adapter used. All the programming results for a specific job must be easily exported for Manufacturing Execution Software (MES), Statistical Process Control (SPC) or other analysis or archiving software.

After the product is verified, the job management utility must securely transfer jobs to programming systems on the factory floor or across the globe. When the line manager starts the line, the data file is pulled from its network location to the programming system on the line, thus ensuring data content matches the product build recipe. Additionally, the job management utility should be able to remotely command the programmer to start, stop, pause, and resume a job. If programmer performance declines, the job management utility should alert management so steps can be taken immediately to address the issue.

A factory integrated approach to Just-in-Time programming


OEM/EMS Workstation
OEM/EMS Local Server
Programming Job Created
Programming Statistics Queried
Remote Job Download to Programmer
Programming Statistics Collected
Manufacturing Execution Software (MES)
External Application Interface Module
Traceability Bar-code Scanning
Bare PCB
Screen Printer
Material Dispense
HS Chip Shooter
Fine pitch Placement
Oven
Board Stacker
OEM / EMS
Inline Programming
Information Continuously Reported

What to Look for in a Programming Solution

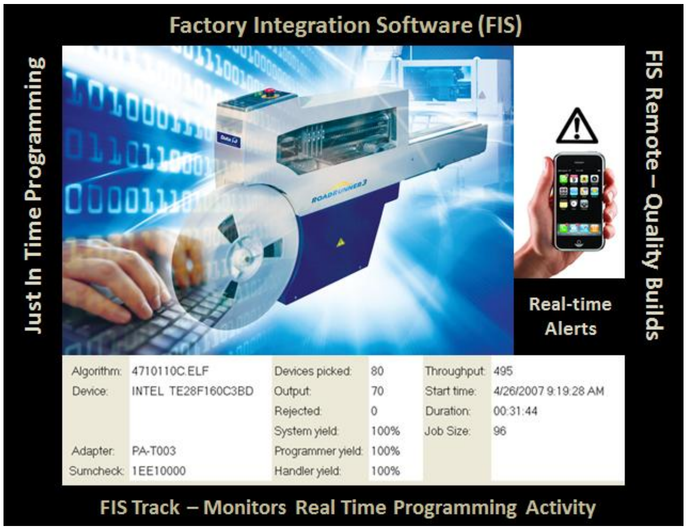
When choosing a programming solution, a number of requirements must be adequately addressed:

Floor space	Occupies little or no floor space.
Device support/ algorithms	Includes a library of algorithms tested to device manufacturer's specifications.
Solution flexibility	Not SMT machine-specific. The solution is flexible enough to work with a range of SMT platforms.
True analytics	Provides up-to-date statistics about performance, throughput, yields, failures, and real-time system performance.
Regulatory compliance	Meets international regulatory compliance requirements.
Distributed manufacturing environment	Once the concept is proven on NPI line, porting to distributed global manufacturing sites is seamless.
Response speed to code changes	Provides the flexibility to respond immediately to last-minute code changes, both prior to and during product build.
Service response	Supported worldwide, with regional service centers that provide answers in customer's time zone.
Breadth of device support	Supports the latest memory (NAND and eMMC) and complex microcontrollers.
Consumable costs	No hidden consumables costs. No additional cost for device support.
Track record of company	Solution provider has a proven track record of financial stability and expertise in the field of programming.
Lean Manufacturing	Supports lean manufacturing principles, doing more with less.
Quality/Process Control	Provides traceability to the component level. Can control system remotely.

DATA I/O'S SOLUTION—ROADRUNNER3 WITH FIS

The industry's most advanced—and only—factory integrated inline programming solution is Data I/O's RoadRunner3 with Factory Integration Software (FIS). This solution is ideally suited for manufacturers with high volume/low mix production environments that program Flash memory and/or Flash microcontroller devices, experience frequent code changes, and require the highest standards of quality control, traceability and IP security. RoadRunner3 addresses the needs of the automotive, wireless, consumer electronics, and industrial controls markets (both OEM and EMS).

RoadRunner3* *with* *Factory Integration Software


Factory Integration Software (FIS)
Just In Time Programming
FIS Remote – Quality Builds
Real-time Alerts
FIS Track – Monitors Real Time Programming Activity

Algorithm:	4710110C.ELF	Devices picked:	80	Throughput:	495
Device:	INTEL TE28F160C3BD	Output:	70	Start time:	4/26/2007 9:19:28 AM
		Rejected:	0	Duration:	00:31:44
		System yield:	100%	Job Size:	96
Adapter:	PA-T003	Programmer yield:	100%		
Sumcheck:	1EE10000	Handler yield:	100%		

RoadRunner3 provides advantages for OEM and EMS manufacturers currently using non-inline programming methods:

- For those **outsourcing**, RoadRunner3 saves costs and eliminates delays associated with long supply chains and frequent code changes that create scrap and rework.

- For those **programming at test**, RoadRunner3 keeps up with constantly growing file sizes, takes no floor space, doesn't add to line processing time, requires no customer algorithm development, and eliminates production bottlenecks.

- For those **programming offline**, RoadRunner3 saves money by reallocating production resources to other areas and reducing costs associated with floor space, offline capital equipment, shipping, and inventory.

RoadRunner3 advantages

Floor space requirement Mounts directly on the SMT machine and takes no additional floor space.

Device support/ algorithms Data I/O is the leader in algorithm development. All algorithms are written to manufacturers' specifications and thoroughly quality tested prior to release.

Solution flexibility Using an interface kit, one RoadRunner3 supports major SMT platforms Fuji, SIPLACE, Panasonic, MYDATA.

True analytics Contains a suite of software elements ("Factory Integration Software") to improve process control from design through manufacture.

Regulatory compliance Meets all international regulatory compliance requirements.

Work in distributed manufacturing environment Can be easily replicated in nearly real time in all environments, from NPI to distributed global manufacturing sites.

Response speed to code changes Provides just in time programming, supporting code changes on the fly.

Service response Data I/O has worldwide support with service centers in the United States, China and Germany to provide live responses to customer questions.

Breadth of device support Supports the latest memories and complex microcontrollers. Data I/O offers the largest suite of NAND bad block management schemes.

Consumable costs No hidden consumables costs. One free algorithm and standard adapter, plus use of all existing RoadRunner algorithms.

Track record of company With over 40 years in business, Data I/O has a proven track record as the premier solution provider for programming.

Lean Manufacturing An inherently lean solution that programs just in time, eliminating waste associated with inventory float and long supply chains, and overhead associated with offline and outsourced programming.

Quality/Process Control Integrated with factory software to eliminate human errors. Only correctly programmed devices are delivered to the SMT pick point.

Conclusion

Data I/O's RoadRunner3 with Factory Integration Software (FIS) is changing the way process engineers and production managers simplify and manage device programming. This solution ensures quality products are built **at the right time** and **at the lowest cost**.

RoadRunner3 with FIS prevents human error, eliminates rework associated with last minute code changes, eliminates inventory float, and prevents IP theft. Configurable to a variety of SMT platforms, RoadRunner3 ensures the image file matches the build recipe for quality builds and provides real-time alerts to optimize machine uptime.

For more information on the benefits of factory integrated just-in-time programming, contact your local Data I/O Sales Representative.

About Data I/O

For 40 years, Data I/O Corporation has been the world leader in the manufacture, distribution and service of innovative solutions for programmable devices. Since 1972, we have championed the use and growth of programmable devices in a wide variety of industries and products. Today, our customers manufacture millions of products each year using Data I/O programming solutions to reliably, securely, and cost-effectively implement their Intellectual Property into programmable devices.