Free online series!

CODESYS® Tech Talk Fall 2026

The Cost of Connectivity: Real-Time EtherCAT and OPC UA on One IPC Controller, with NODKA IPCs and CODESYS

Eddie Chau, Business Development | Cesar Hernandez, Technical Support | NODKA

Agenda

01
The request

02
In practice

03
Cost of connectivity

04
Where to fix it

05
Measuring

06
What to do

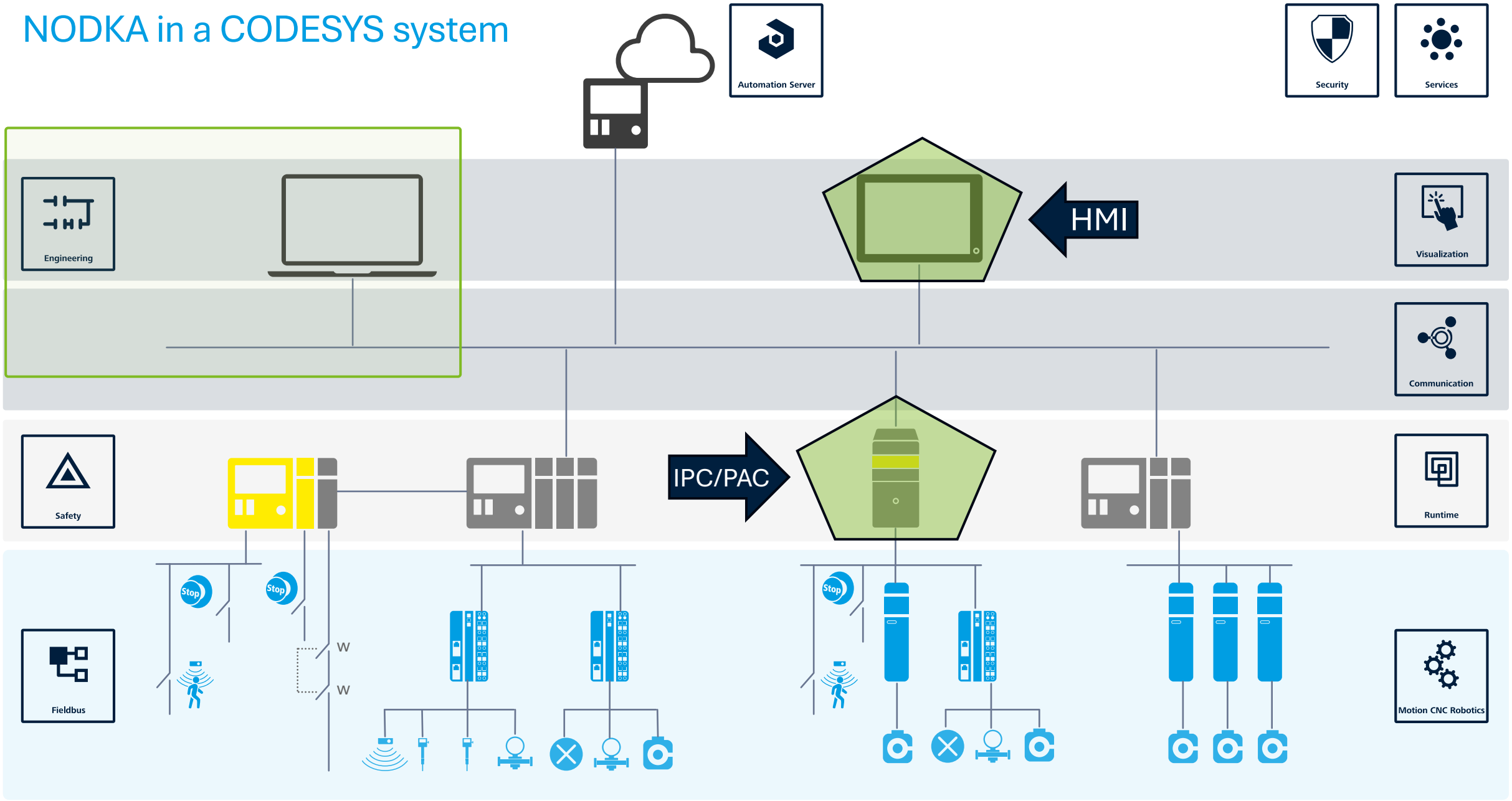


About NODKA

- One-stop IPC, panel PC and HMI design and manufacture
- Specialized platforms for industrial automation, not repurposed general-purpose computers
- High performance hardware for demanding applications
- Customizable at the hardware and firmware level, which is the layer this session is about
- Runtime-neutral: CODESYS provides the runtime, the EtherCAT master and the OPC UA server. We provide the platform underneath.



NODKA in a CODESYS system



The request that keeps arriving

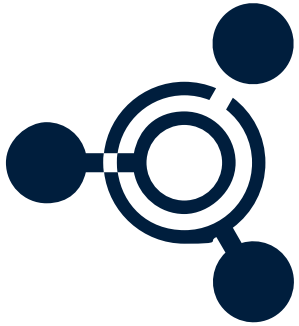
Nothing about the machine changes. Only what it is asked to do.
The cost in this session is measured in microseconds, not in license fees.

The machine today



A fixed cycle it has held for years.
One fieldbus. Proven, and in production.
Nothing else asks it for anything.

What is asked of it now



The same cycle, unchanged.
Plus data northbound, to SCADA, MES or a historian.
Nobody offers to relax the cycle time in exchange.

The application story

One machine, one processor, two jobs that both have to be met

The challenge



A cycle that must not slip.
A demand for data that does not stop arriving.
Both land on one machine.

The IPC



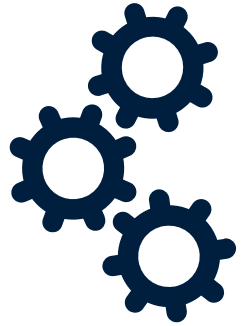
One IPC carrying the runtime, the EtherCAT master and the OPC UA server.
Configured below the runtime, where scheduling cannot reach.

The result



Measured, not asserted.
Maximum jitter and cycle time, under load, over a stated run.
Part 2 fills this in.

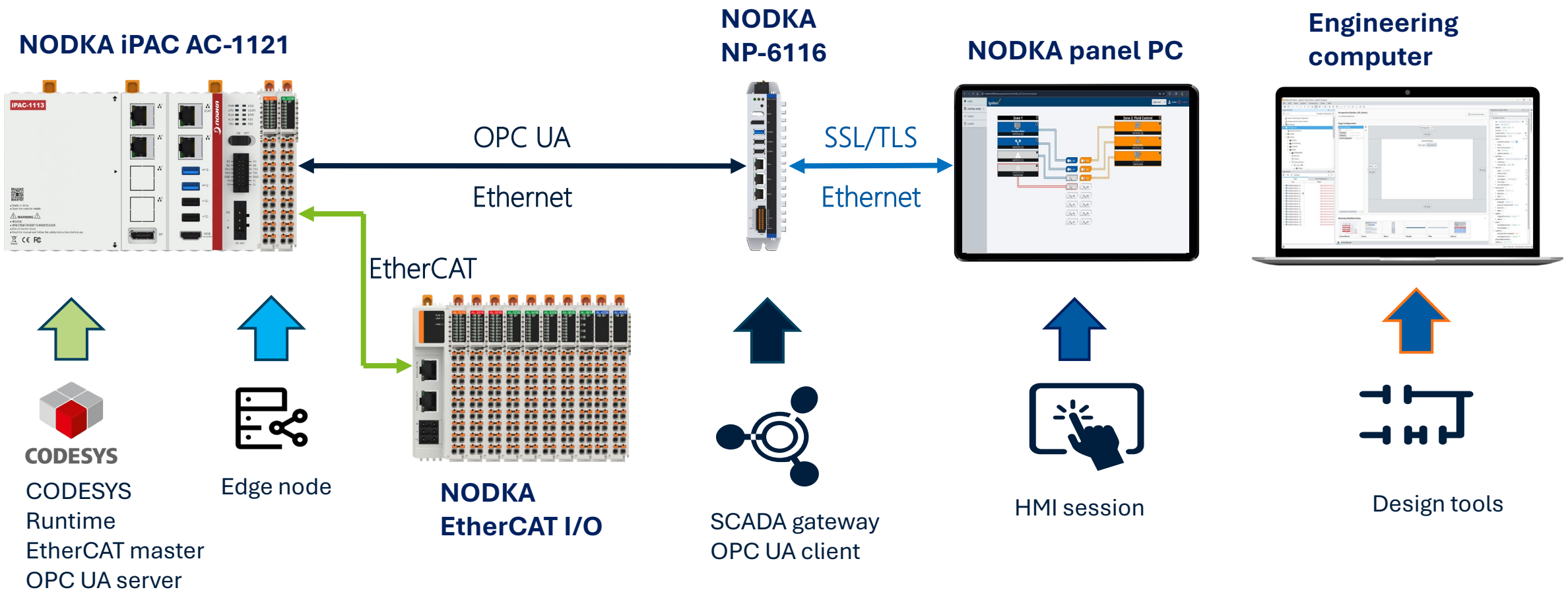
The application



Conveyor and fluid control zones today.
The same shape in motion control, packaging and material handling.

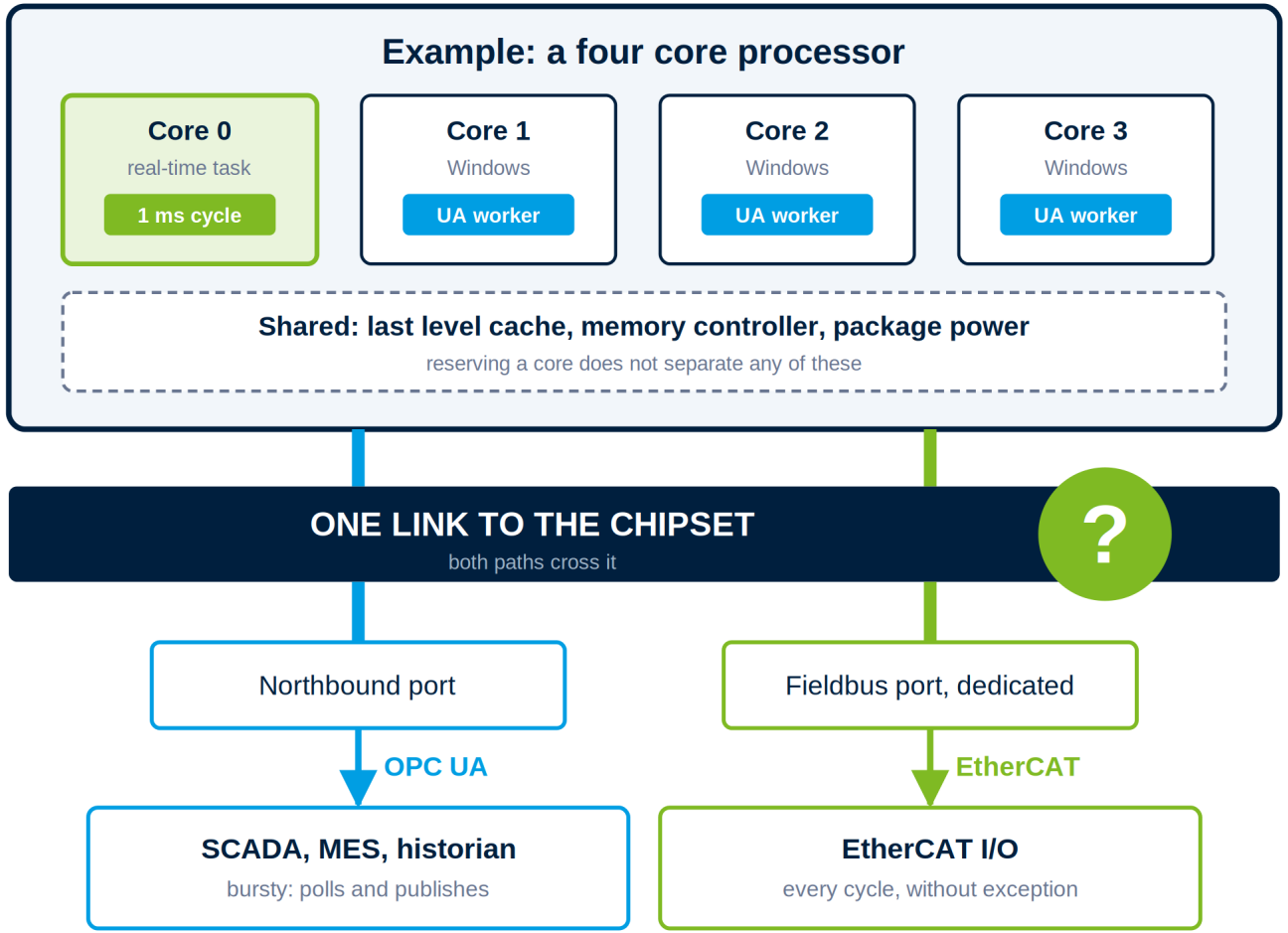
An application case: conveyor and fluid control

A working machine that had to publish to the plant without giving up its cycle



Two jobs, one processor

Where the fieldbus and the northbound connection actually collide



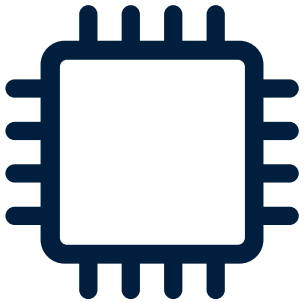
The question

What does the blue path cost the green one, and how much of that can BIOS configuration take back?

Where the cost comes from

What the OPC UA server actually does to the machine it is sharing

It brings its own tasks



The CODESYS OPC UA server generates worker tasks of its own, up to the core count less one. On a four core machine that is up to three, created as the load requires.
They avoid the real-time core. They do not avoid the cache, the memory controller or the link to the chipset.

It arrives in bursts



The fieldbus is cyclic and even. A SCADA client polls and publishes in bursts.
Bursts are what expose transition costs.
Continuous load hides them.

Its defaults set the load



Its scheduler resolution is 1 ms, the same order as the control task.
Monitored items are sampled no faster than 100 ms by default. Lower that floor and the load rises with it.

Where timing is actually decided

Four layers, and the one that no amount of tuning in the layers above can reach

Application logic

Task period, code efficiency, what runs each cycle

Control runtime and operating system

Task priorities, core assignment, RT extension or RT kernel

Firmware and BIOS settings

Power states, link power management, interrupt sources, clock behavior

Processor and chipset hardware

Core count and type, cache, network controller, platform design

The constraining layer

No amount of tuning in the layers above removes a delay the firmware allowed the hardware to introduce.

Delay sources the firmware layer controls, and the scale each one works on

Frequency transition

platform dependent

PCIe link wake up

microsecond scale

Deep C State exit

microseconds to hundreds

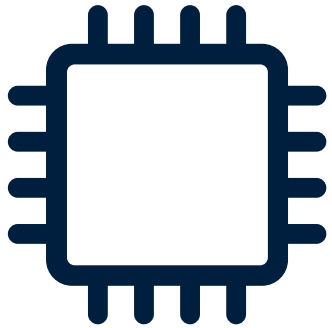
System management interrupt

tens to hundreds of microseconds

What CODESYS asks for

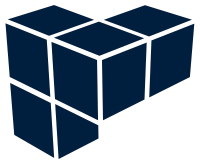
The runtime vendor's own firmware guidance, before anyone measures anything

Core topology



Disable hyper-threading and SMT.
On 12th generation and later, disable the efficiency cores.

Virtualization



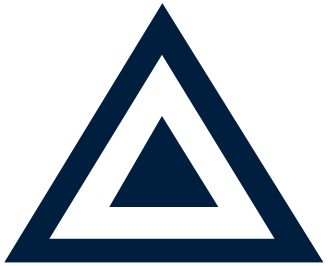
Disable virtualization technology.
VT on Intel, AMD-V on AMD.
The runtime wants to own its cores directly.

Frequency and power



Disable the energy saving options that cause frequency changes, and the deeper sleep states.
Transitions are what cost time, not the low frequency itself.

Interrupt sources



Avoid legacy USB, periodic self-tests, and anything else that produces periodic SMIs.
No software priority preempts them.

What most IPCs already handle

Three classes of machine, and what each one leaves on the table

General-purpose PC



Firmware defaults left alone.
Tuned for average speed, which is a different goal from worst case.

Automation IPC



The well-known items are usually handled already.
Hyper-threading, virtualization, the obvious sleep states.

Real-time tuned IPC



Also frequency and voltage policy, link power management, integrated graphics, memory and System Agent, hybrid cores and legacy interrupts.
This is the layer usually left.

Which settings move the number

Nine groups. The mechanism differs, and the mechanism is what tells you how to treat each one.

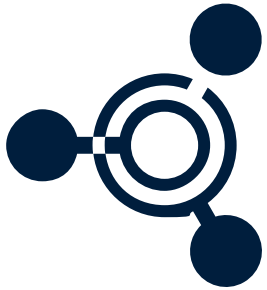
Setting group	Typical names in the menu	Why it costs time
CPU speed changes	Intel SpeedStep, Intel Speed Shift Technology, Turbo Mode	Frequency and voltage transitions add latency and make execution time depend on platform state
CPU sleep states	C States, Package C States, Low Power S0 Idle Capability	Deeper states take longer to exit, and may flush the cache on the way in
Core topology	Hyper-Threading, Simultaneous Multithreading	Two threads on one core means execution time depends on what the sibling thread is doing
Hybrid core types	Performance and efficiency cores	Runtime support varies, and some runtimes need the efficiency cores disabled
Virtualization	Intel VMX Virtualization Technology, VT-d, IOMMU	Requirements vary by runtime, and some need it off so the runtime can own a core directly
Link power management	Native ASPM, DMI Link ASPM Control, PCI Express Clock Gating, SATA Aggressive LPM Support	The first transaction after an idle gap waits for the link to wake
Integrated graphics	RC6 Render Standby, Gfx Low Power Mode, Maximum GT Frequency	Shares package power, cache and memory bandwidth, even with no display attached
Memory and System Agent	SA GV, Maximum Memory Frequency, Page Close Idle Timeout	Voltage and frequency changes stall memory access briefly and invisibly
Interrupt and SMI sources	Legacy USB Support, PS/2 Emulation, TPM Activity, Periodic Firmware Self-Tests	They run below the operating system, so no software priority can preempt them

The color is the mechanism: **Transition cost** **Contention for a shared resource** **Asynchronous interference**

So what does it cost, and what comes back?

Two questions, and the second half of this session answers both on a real controller

What it costs



What does adding OPC UA cost the cycle we are protecting?
Measured on the machine, not estimated.

What comes back



How much of that can BIOS configuration take back?
Same load, same task, one image against the other.

We measured both. Part 2 has the numbers.

Handover to Part 2

The runtime decides what should happen. The IPC decides whether it can.

Part 1

The question, and why the BIOS layer is where it is answered.

Part 2

The measurement, on one controller, with the numbers stated.

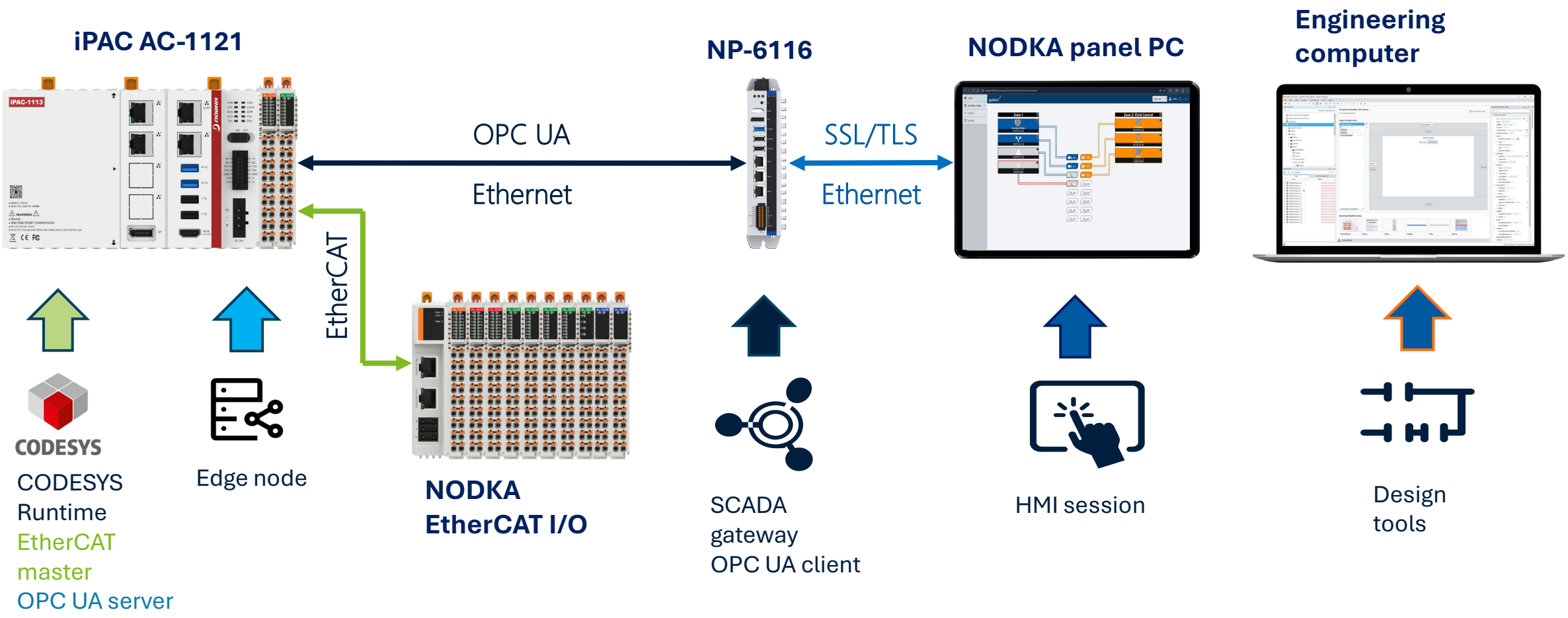
CODESYS® Tech Talk Fall 2026

Part 2: Real-Time Performance Connectivity Costs & BIOS Control

Cesar Hernandez | Automation Systems Specialist, NODKA

Measurement Setup

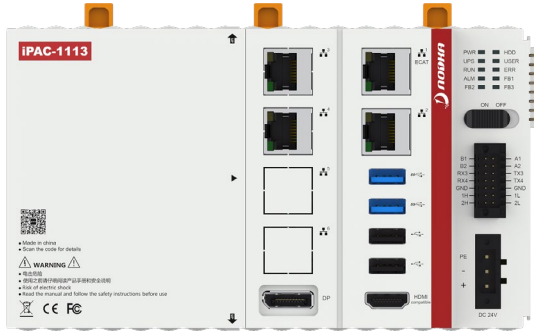
Setup executing EtherCAT master and OPC UA operations



System Configuration

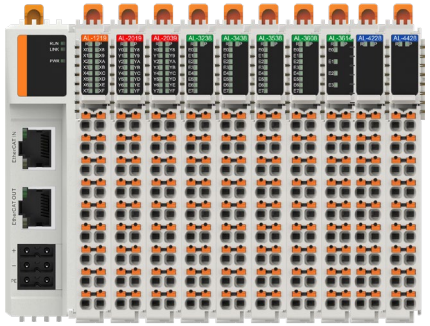
Specific component and runtime configurations used for luation

Controller



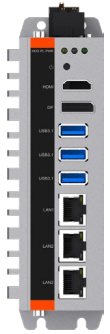
- iPAC AC-1121 Intel Core i3-8100, four cores
- No hyper-threading, no Turbo Boost
- CODESYS Control RTE 3.5.17.10
- One cyclic task, fixed for the session. The period is stated with every result

EtherCAT I/O



- Internal AK1100 coupler, plus a second coupler on a remote segment.
- AL series digital and analog modules.
- This is the process image the cycle has to service.

OPC UA client



- Running on a separate NP-6116, not on the controller.
- Connected on the northbound port, not the fieldbus port.
- Nothing measured here depends on which client.

BIOS images tested. The standard and real-time BIOS images were flashed and tested separately on the same controller.

EtherCAT initialization & ESI revision

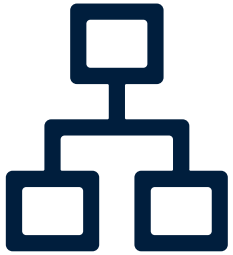
How we brought the EtherCAT network online before starting the performance tests

Scan and identify the devices



- Read Vendor ID
- Read Product Code
- Read Revision

Build the EtherCAT device tree



- Local and remote I/O under one EtherCAT master.
- One process image for the control application.

Verify the ESI revision



- Device revision must match the project definition.
- A mismatch can prevent EtherCAT from reaching Operational State.

> If the device tree looks correct but EtherCAT will not reach OP, check the ESI revision.

EtherCAT Segment Isolation

Keeping local I/O active when a remote EtherCAT segment is disconnected

Default: one shared sync unit



- All devices share one datagram
- One missing device can invalidate the entire datagram
- The failure affects both local and remote I/O

Isolated: one sync unit per segment



- Local and remote segments use separate datagrams
- A remote disconnect invalidates only the remote group
- Local I/O can continue exchanging data

CODESYS Real-Time Measurement Method

How we measured cycle timing and execution behavior during each test

Fixed Cyclic Task

- 1 ms task period
- Same application code for each comparison
- Light and heavy workloads executed inside the task

Timing Measured in CODESYS

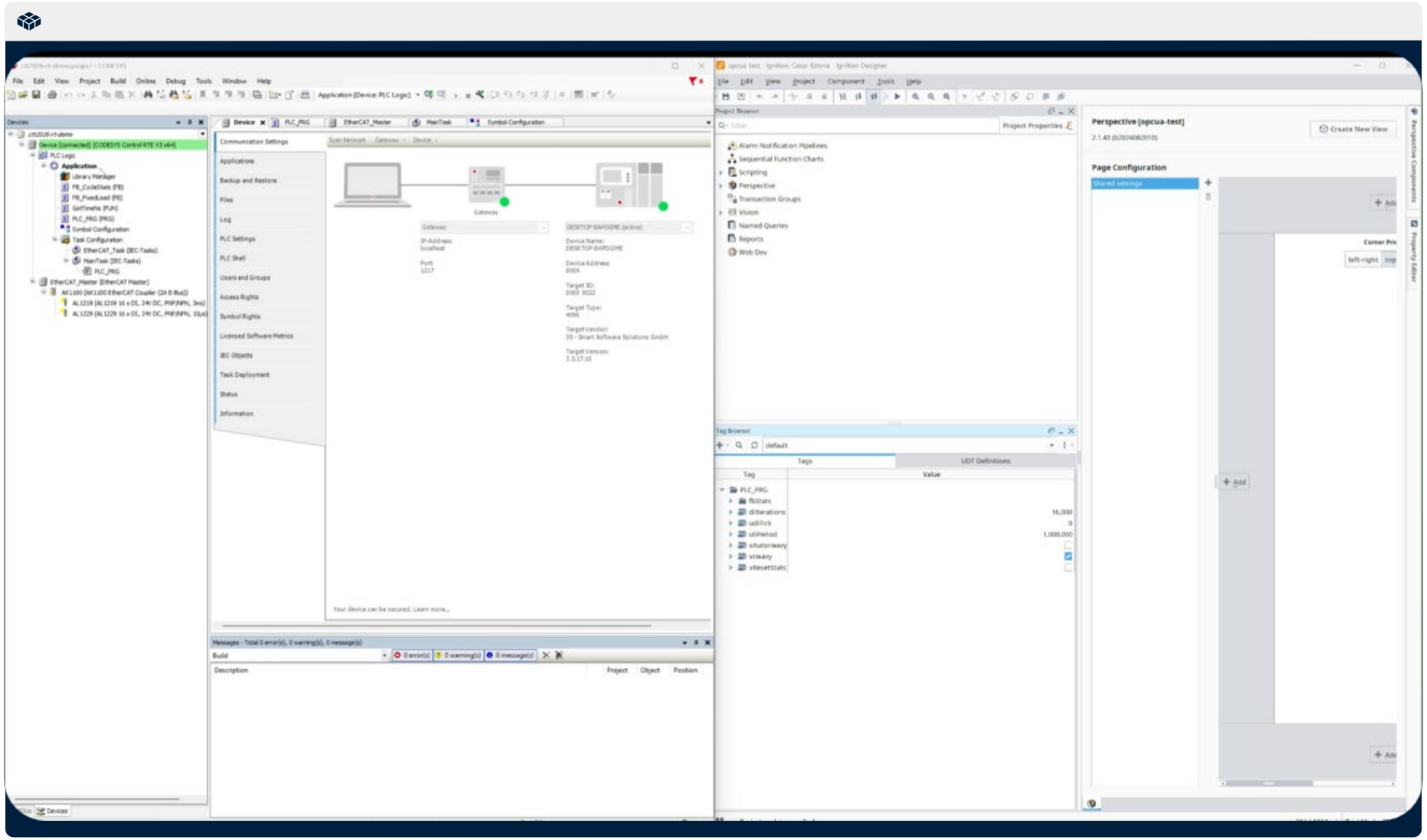
- Cycle deviation / Jitter
- Task Execution time
- Minimum and maximum values tracked for each run

Results Recorded over Time

- Maximum values first
- Threshold counts at 5 µs and 50 µs
- Run duration and total cycle count recorded for every test

> Measured inside the task using the runtime clock. If the minimum stays the same but the maximum moves, the variation is coming from the system around the task.

CODESYS over OPC UA



Designing the Disturbance Load

Creating a repeatable load to expose timing variation

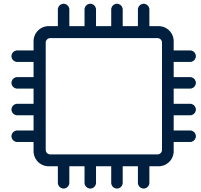
Quiet tests can miss the timing delays caused by firmware transitions.

Short bursts with idle gaps



- **Repeated transitions expose wake-up and ramp costs**
- **Continuous load can hide those effects**

Stress shared resources



- **Memory Bandwidth**
- **Network activity on a non-fieldbus port**
- **Display Activity**

How we generate it: fixed schedule, alternating light and heavy work, display activity, network bursts on a non-fieldbus port, and memory-bandwidth bursts. The same pattern is used for every run.

Reading the Timing Results

What to watch before we compare OPC UA timing

1 Maximums matter most

- One delayed cycle can be enough to create a real-time problem

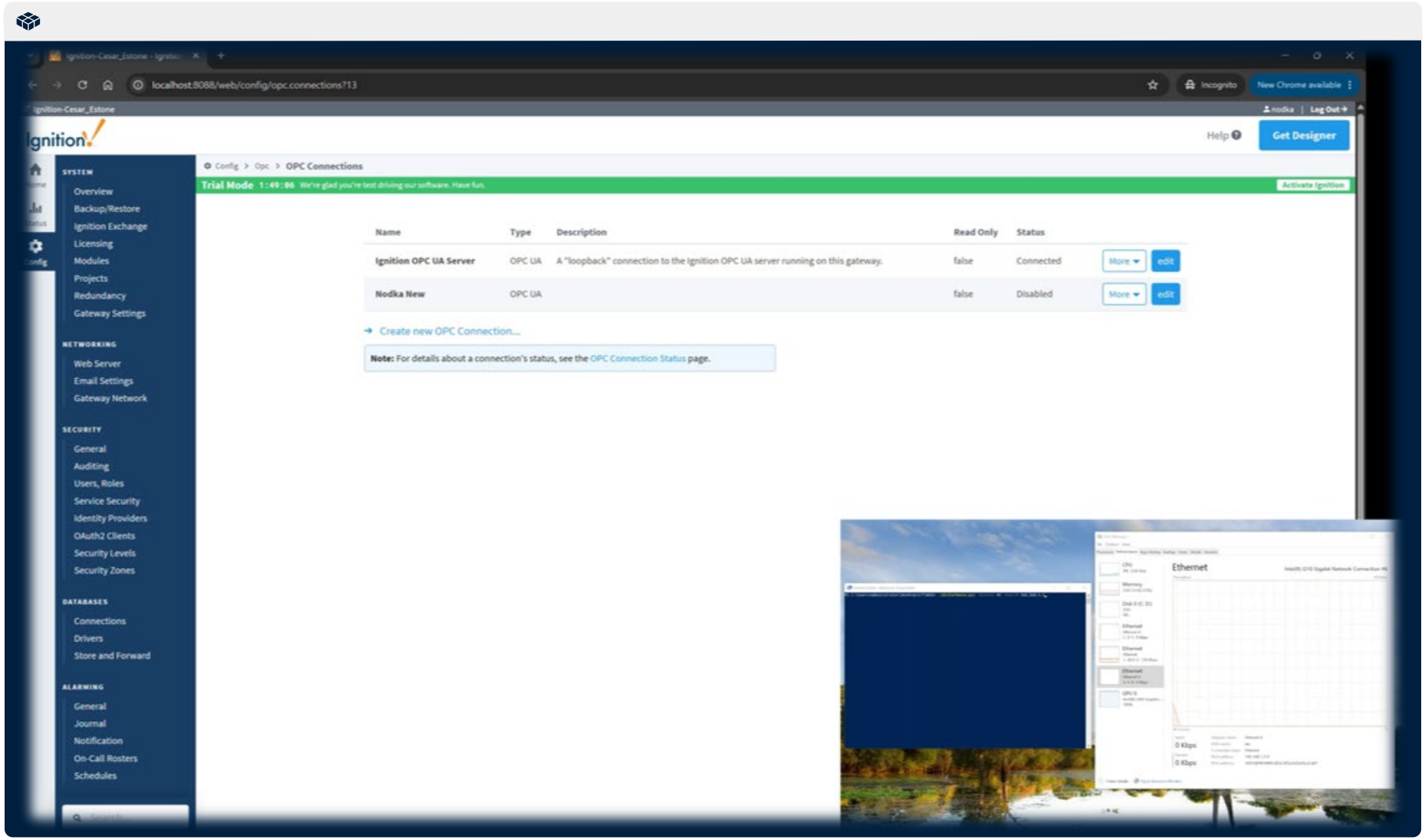
2 Percentiles add context

- They show the overall distribution, but not the worst case

3 Run length matters

- Maximum values only have meaning when the test duration is known
- 30 second runs are not the same as several hours

OPC UA Reconnect Timing



OPC UA Client Impact on Timing

Same real-time setup, client disconnected vs. connected

Client off

Heavy execution	**116 – 124 µs**	1.07 : 1
Cycle deviation	**−25 / +24 µs**	p-p 49 µs
Cycles above 50 µs	**0**	of 1.20 M
Cycles above 5 µs	**46.97 %**	

Client on

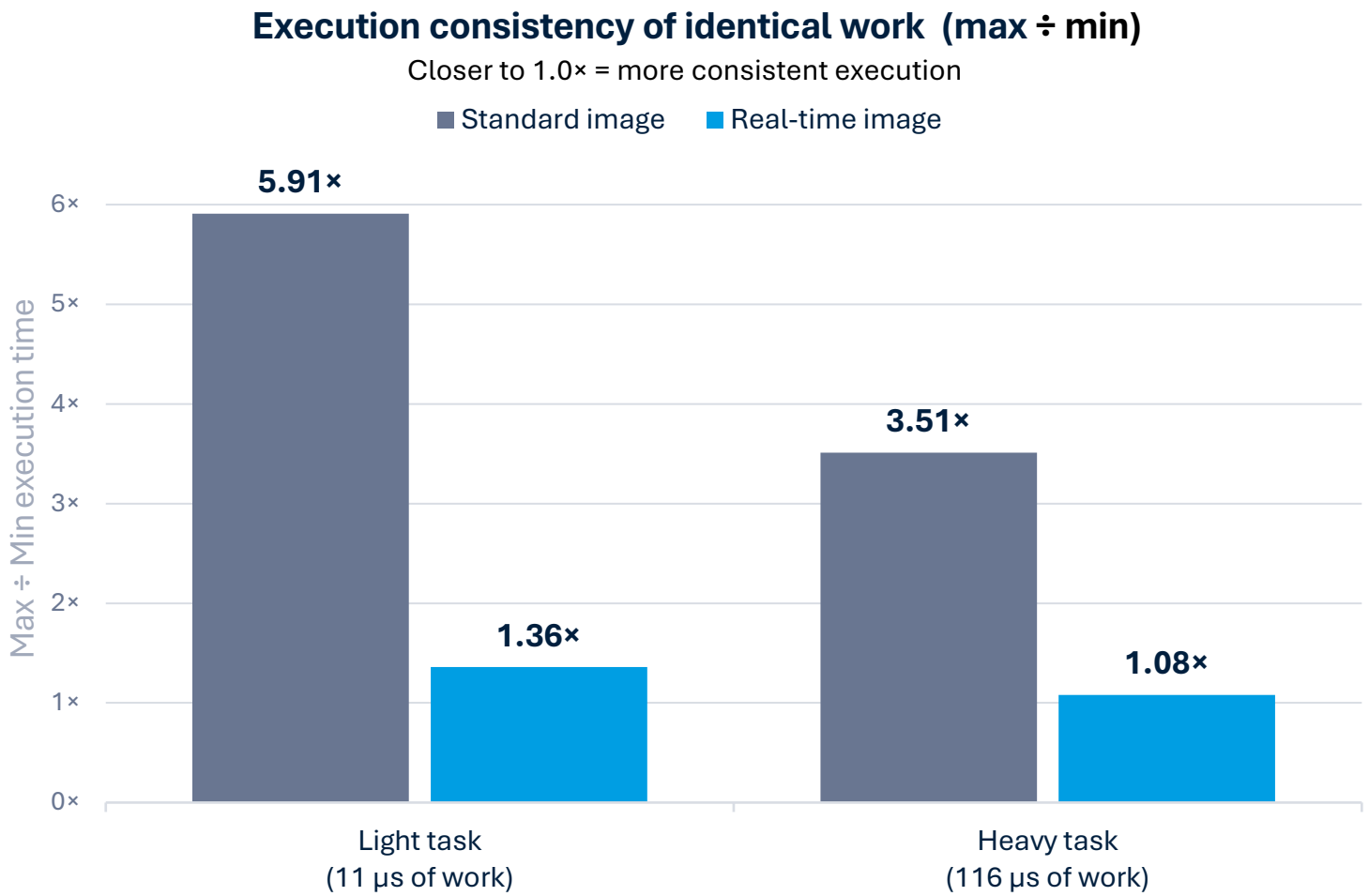
Heavy execution	**116 – 142 µs**	1.22 : 1
Cycle deviation	**−69 / +66 µs**	p-p 135 µs
Cycles above 50 µs	**22**	of 1.20 M
Cycles above 5 µs	**34.97 %**	

The extremes moved. With the client connected, maximum execution increased and cycles above 50 µs appeared.

Performance Comparison: Standard vs. Real-Time

Same controller · same code · same disturbance · BIOS image changed

45 min per image · 1 ms cyclic task · five-channel disturbance · ~2.69 M cycles each · iPAC AC-1121, CODESYS Control RTE



- **The floor stays the same**

Light: 11 µs on both images
Heavy: 116 µs on both images

- **Only the ceiling changes**

Standard BIOS max: **65 µs light / 407 µs heavy**
Real-time BIOS max: **15 µs light / 125 µs heavy**

- **Jitter difference is smaller under load**

Cycles beyond 5 µs:
11.24% Standard vs **9.98% Real-Time**

Standard vs Real-time BIOS under disturbance

Standard vs. real-time BIOS in quiet and disturbed conditions

	Quiet Setup	PowerShell disturbance
Standard firmware	max **31** µs 60 µs peak-to-peak · cycles beyond 5 µs: 18.41 % deadline misses: 0 · 20 min · 1.20 M cycles	max **45** µs 87 µs peak-to-peak · cycles beyond 5 µs: 11.24 % deadline misses: 0 · 45 min · 2.69 M cycles
Real-time firmware	max **23** µs 45 µs peak-to-peak · cycles beyond 5 µs: 4.26 % deadline misses: 0 · 20 min · 1.20 M cycles	max **36** µs 66 µs peak-to-peak · cycles beyond 5 µs: 9.98 % deadline misses: 0 · 45 min · 2.69 M cycles

Same controller, same 1 ms task, same code. No deadline misses. Compare down each column: 20 min quiet, 45 min disturbed.

Conclusion

What did OPC UA cost, and what did the real-time BIOS recover?

1 OPC UA moved the extremes

- Heavy execution max: 124 → 142 µs
- Cycles above 50 µs: 0 → 22
- Bulk distribution did not simply get worse

2 Real-time BIOS tightened execution

- Heavy execution max: 407 → 125 µs
- Heavy spread: 3.51× → 1.08×
- Worst-case jitter lower in both conditions

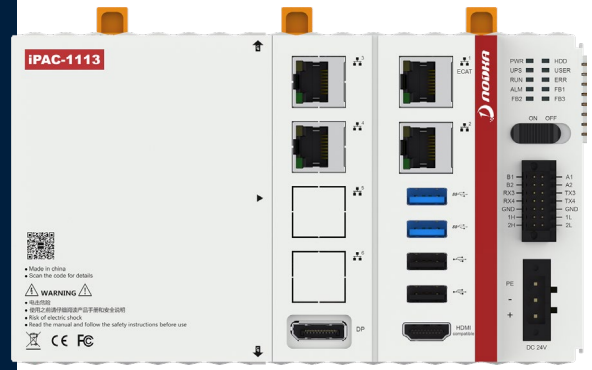
3 Answer to Part 1

- Connectivity can add timing cost
- Real-time BIOS can reduce worst-case variation
- Measure under load, maximum first

Bottom line: One IPC can run EtherCAT and OPC UA together, but real-time behavior depends on how the platform underneath CODESYS is configured.

Key takeaways

One controller, two jobs, one set of shared resources



Connectivity has a measurable cost

- EtherCAT and OPC UA still share processor, memory, and chipset resources

BIOS configuration changes the worst case

- Real-time BIOS settings reduced execution spread and maximum jitter

The biggest improvement is consistency

- The task was not fundamentally faster; the maximum stayed much closer to the minimum and more predictable.

CODESYS

CODESYS®
Tech Talk Fall 2026
October 5-9

CODESYS® is a registered trademark. Subject to misprints, errors, and technical changes. Reproduction (even in parts) and distribution only with prior permission. Please note: Not all CODESYS functions are available in all countries. For further information on these country-specific restrictions, please contact sales@codesys.com.

Industrial PC BIOS settings for real-time

nodka.com/technology/industrial-pc-bios-real-time-settings/

EtherCAT with CODESYS on an IPC

nodka.com/technology/step-by-step-guide-configuring-an-ethercat-controller-with-codesys-on-an-ipc/

Connecting an iPAC to SCADA via OPC UA

nodka.com/news/step-by-step-guide-connecting-a-nodka-ipac-to-ignition-perspective-via-opc-ua/

iPAC IPC controller series, and sales

nodka.com/ipac-ipc-controller-series/ · +1 909-594-7630 · sales@nodka.us



Contact

CODESYS North America

NorthAmerica@codesys.com