FrankenTrace: Low-Cost, Cycle-Level, Widely Applicable Program Execution Tracing for ARM Cortex-M SoC

M. Matraszek, M. Banaszek, W. Ciszewski, K. Iwanicki

UNIVERSITY OF WARSAW

All images were generated with Stable Diffusion v1.5/v2.1 models.

CPS-IoTBench 2023

Imagine Mike

Mike has a centrifuge

Mike has a ~~centrifuge~~ testbed

Mike has a ~~centrifuge~~ ~~testbed~~ drone

Mike has a ~~centrifuge~~ ~~testbed~~ ~~drone~~ white-box CNC device

Mike has a ~~centrifuge~~ ~~testbed~~ ~~drone~~ white-box drone

The firmware behaves weird

The firmware directly drives motors, which produce weird sound.

It runs on *ZombuPilot*

Imagine ZombuPilot

standard software

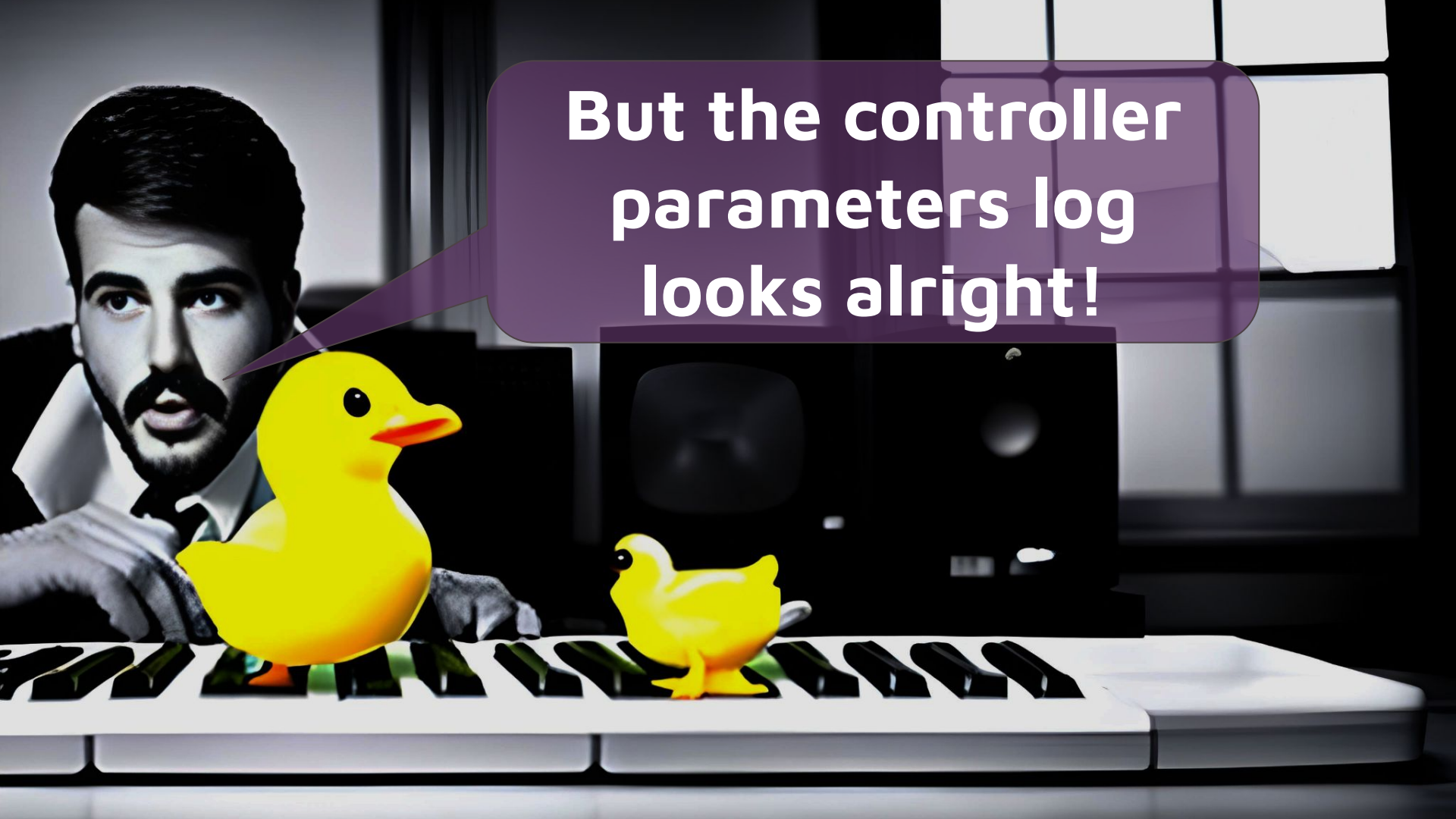
open source

ported to multiple platforms

but everyone uses it with STM32~~H7~~F103


Mike starts
with the
rubber duck
method


That’s a typical problem that needs tracing!

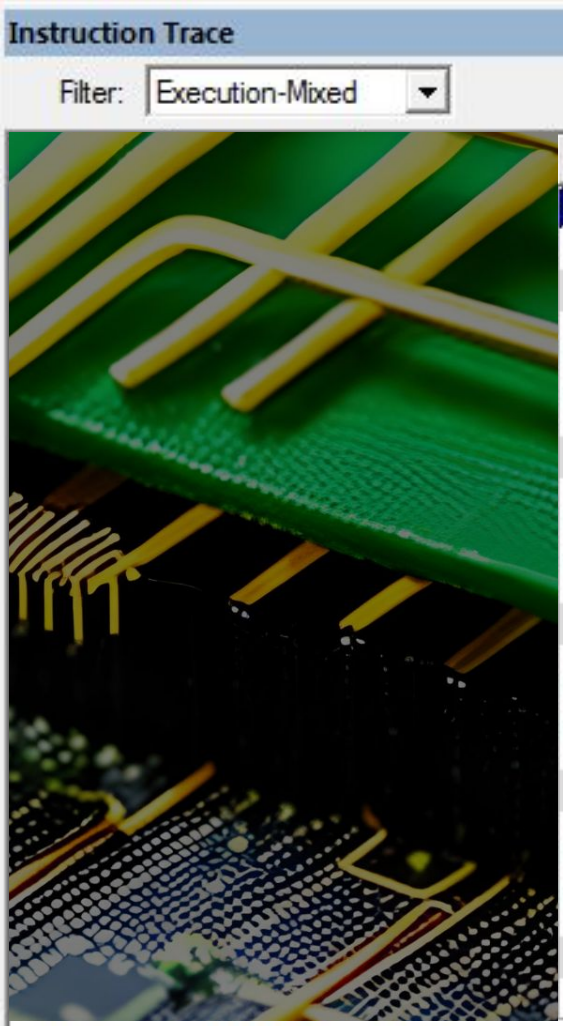

But the controller parameters log looks alright!


No, we mean program-flow tracing!

STM32F103REY
does **not** support
high-speed tracing

Program Instruction tracing

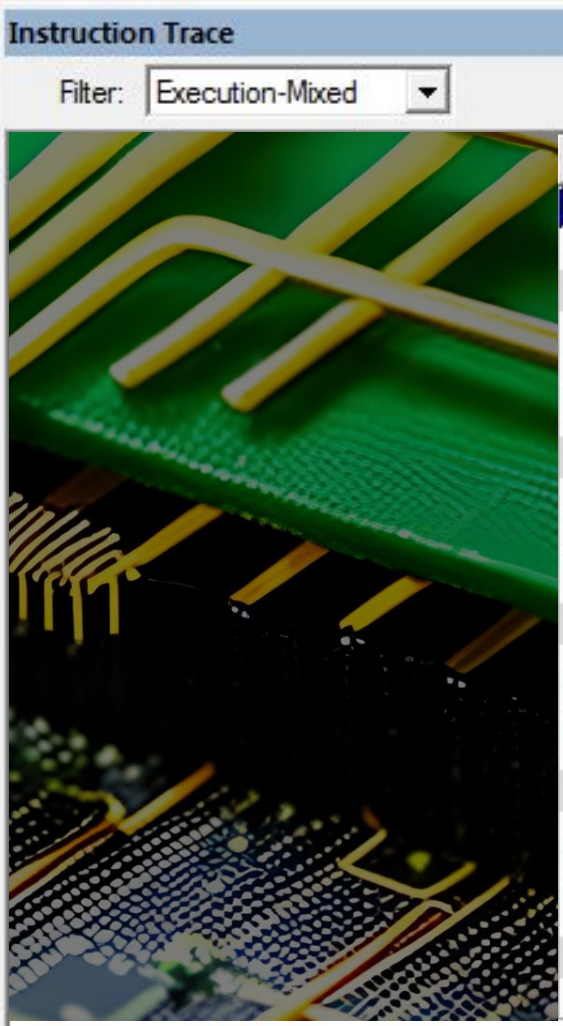
- exact profiling
- real-time
- no buffer limit



source: ARM KEIL µVision User's Guide

Program Instruction tracing

- exact profiling
- real-time
- no buffer limit

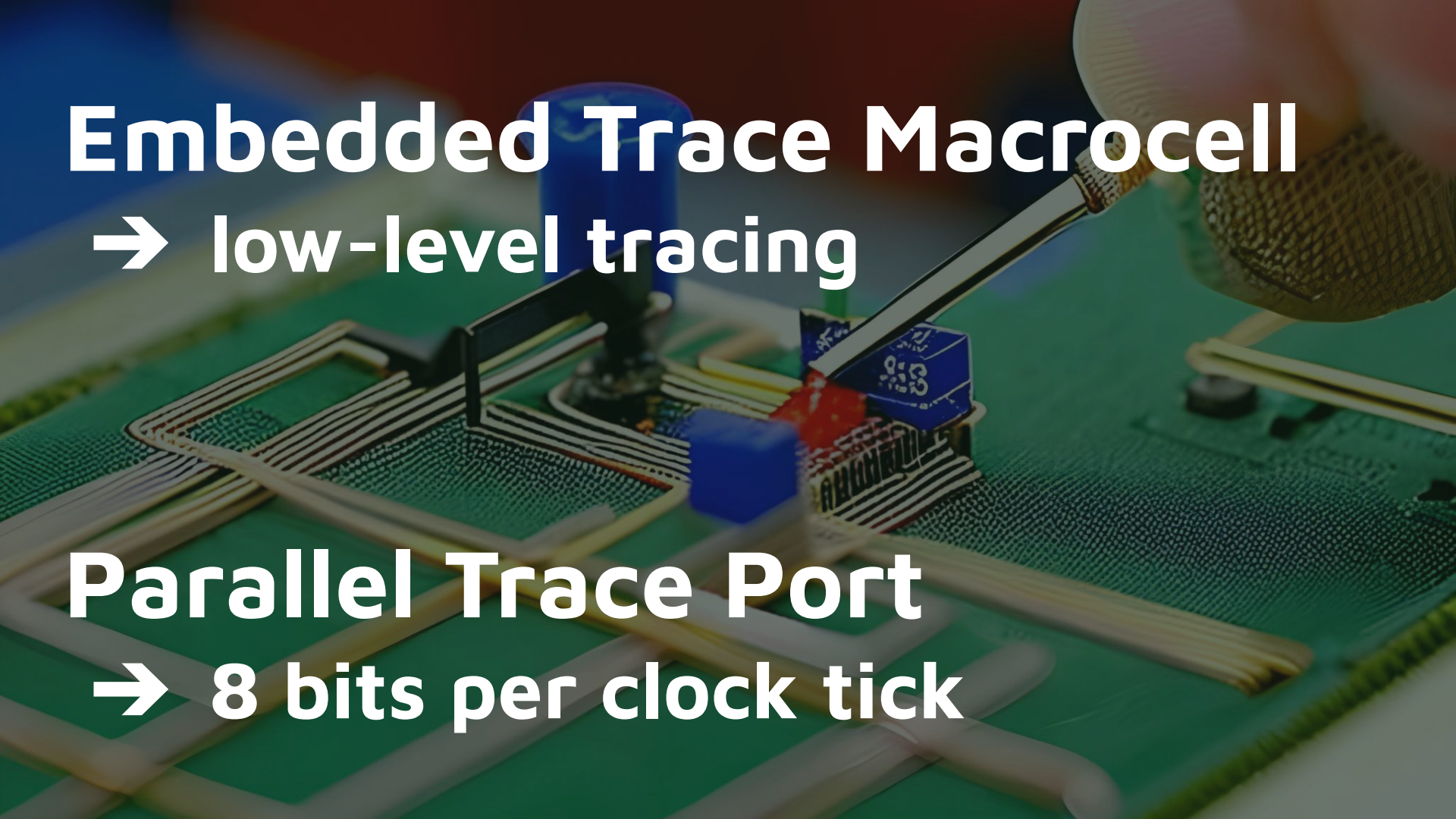


source: ARM KEIL µVision User's Guide

Instruction Trace

Filter: Execution-Mixed

Nr.	Address	Opcode	Instruction
65517	0x00000158	D5FC	BPL 0x00000154
65518	0x00000154	7D0A	LDRB R2,[R1,#0x14]
65519	0x00000154		31: while (!(U1LSR & 0x20));
65520	0x00000156	0692	LSL R2,R2,#26
65521	0x00000158	D5FC	BPL 0x00000154
65522	0x00000154	7D0A	LDRB R2,[R1,#0x14]
65523	0x00000154		31: while (!(U1LSR & 0x20));
65524	0x00000156	0692	LSL R2,R2,#26
65525	0x00000158	D5FC	BPL 0x00000154
65526	0x00000154	7D0A	LDRB R2,[R1,#0x14]
65527	0x00000154		31: while (!(U1LSR & 0x20));
65528	0x00000156	0692	LSL R2,R2,#26
65529	0x00000158	D5FC	BPL 0x00000154
65530	0x00000154	7D0A	LDRB R2,[R1,#0x14]
65531	0x00000154		31: while (!(U1LSR & 0x20));
65532	0x00000156	0692	LSL R2,R2,#26
65533	0x00000158	D5FC	BPL 0x00000154
65534	0x00000154	7D0A	LDRB R2,[R1,#0x14]
65535	0x00000154		31: while (!(U1LSR & 0x20));
65536	0x00000156	0692	LSL R2,R2,#26


Embedded Trace Macrocell
➔ low-level tracing
Parallel Trace Port
➔ 8 bits per clock tick

Capture the trace!
Buy now!

For $1848

Source: Segger J-Trace PRO marketing materials

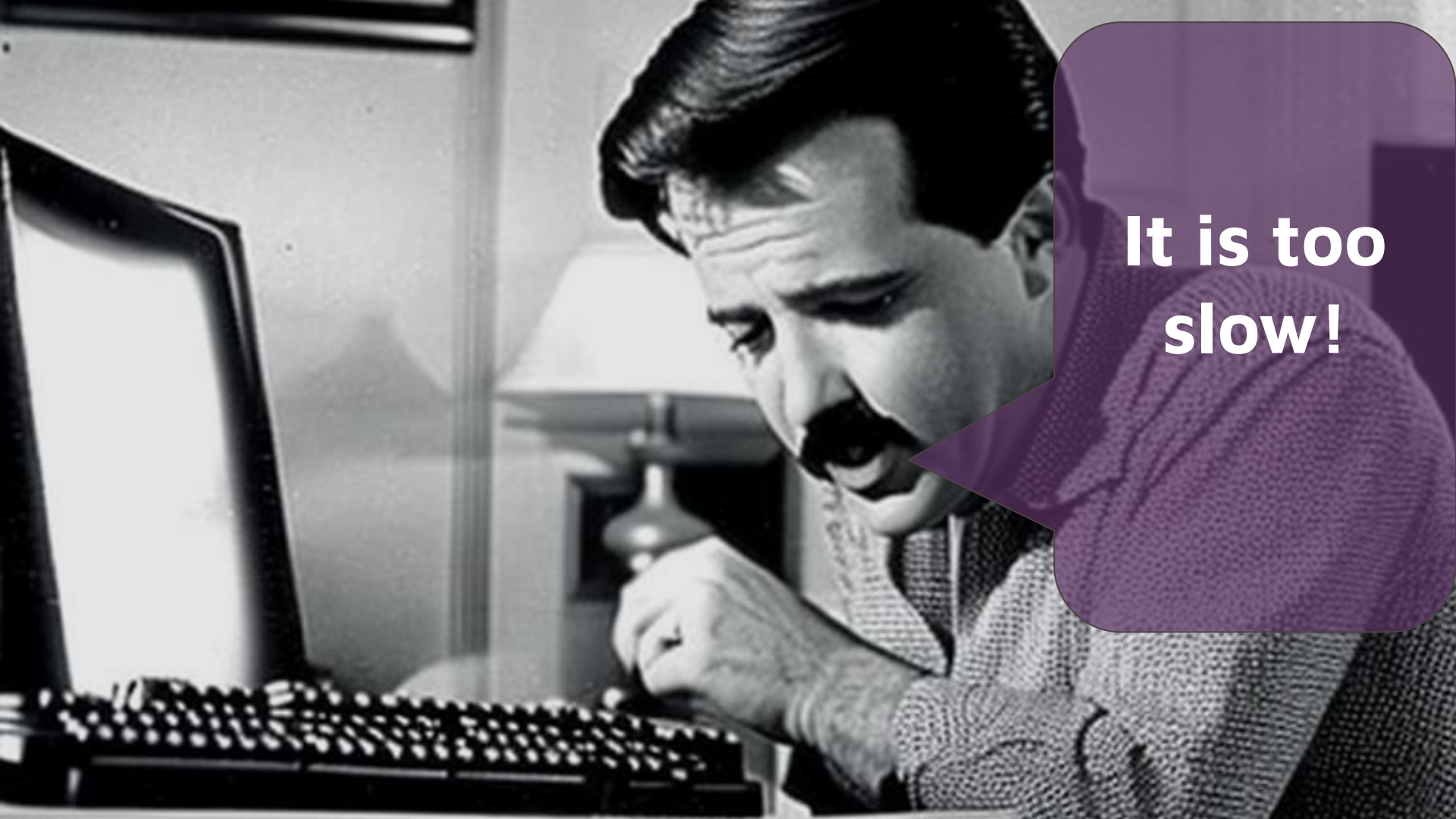

Thank you.
Any questions?


STM32F103REY (64 pin)
does **not** support
high-speed tracing
CC2650 too


Oh, I can just launch a debugger session!

A moment of silence

while we wait for 2 seconds for “step instruction” to execute


It is too slow!


We can do better
Yeah, there are *other* debug components

FrankenTrace

"do more with slow"

FrankenTrace

Low-Cost → $10 USB2UART

Widely Applicable → most Cortex-M (3+)

Cycle-Level → NECROMANCY



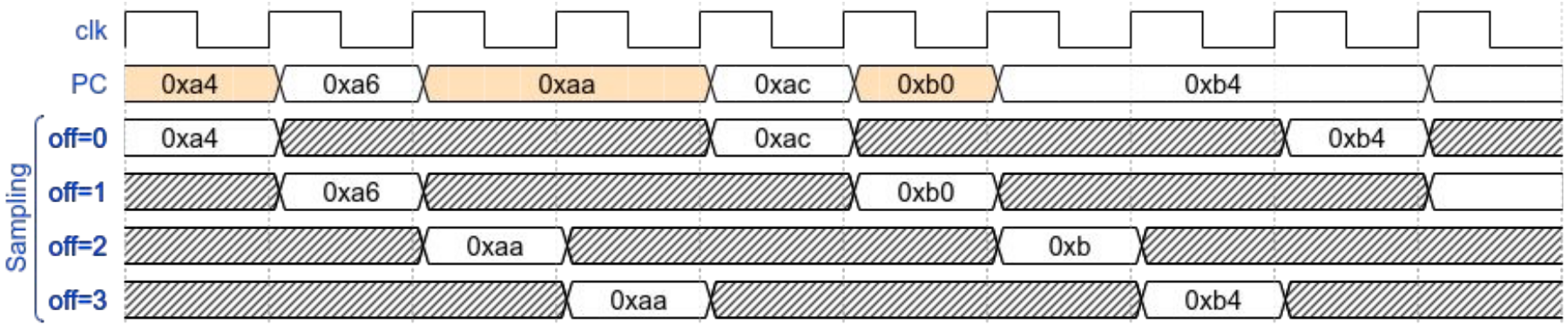




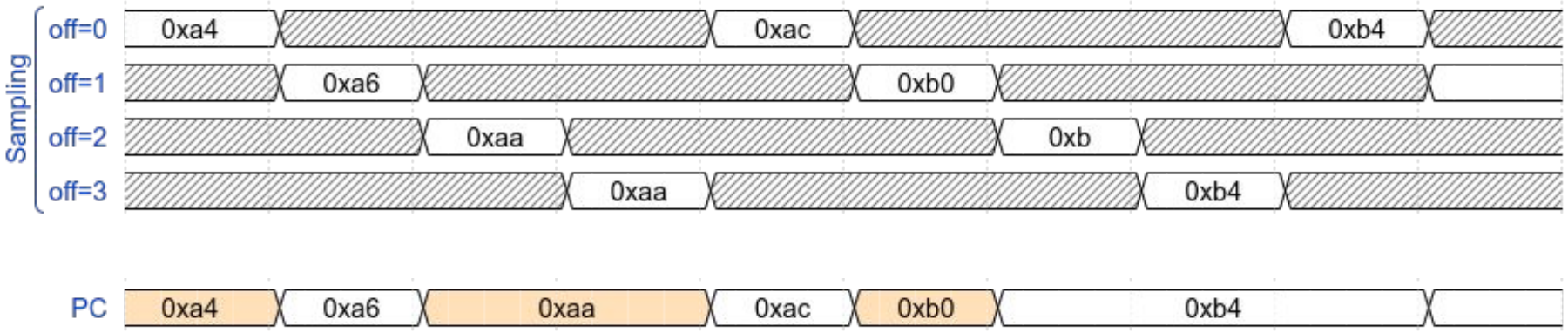
FrankenTrace
PC trace

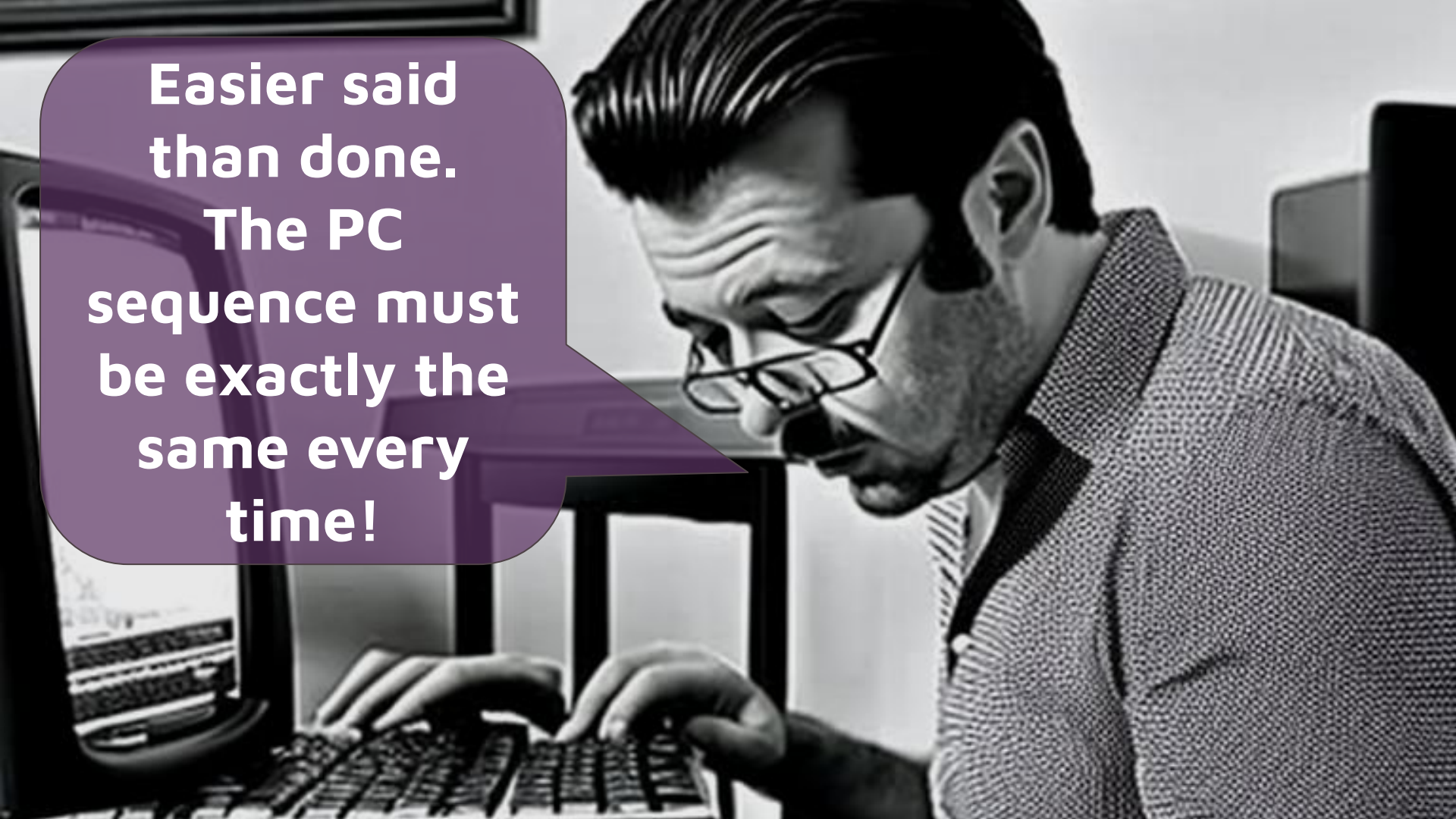
Debug Watchpoint and Trace unit can sample **PC** every **N** $\in$ [32, 8192] **CPU** cycles

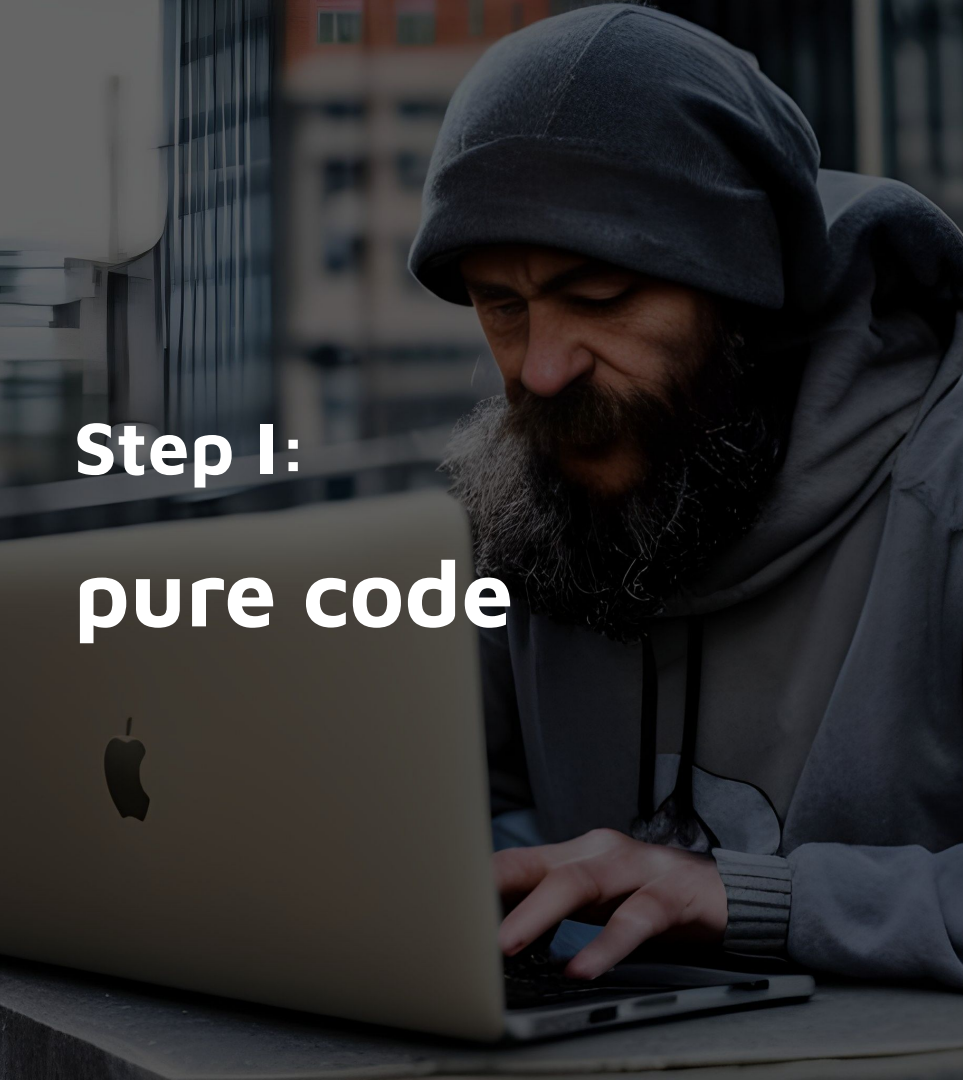
Assume we can sample every N=4 cycles



Assume we can stitch the samples together




Easier said than done. The PC sequence must be exactly the same every time!



Step I:

pure code

Pipeline

Cache

Bus Arbiters

Reset

Step II:

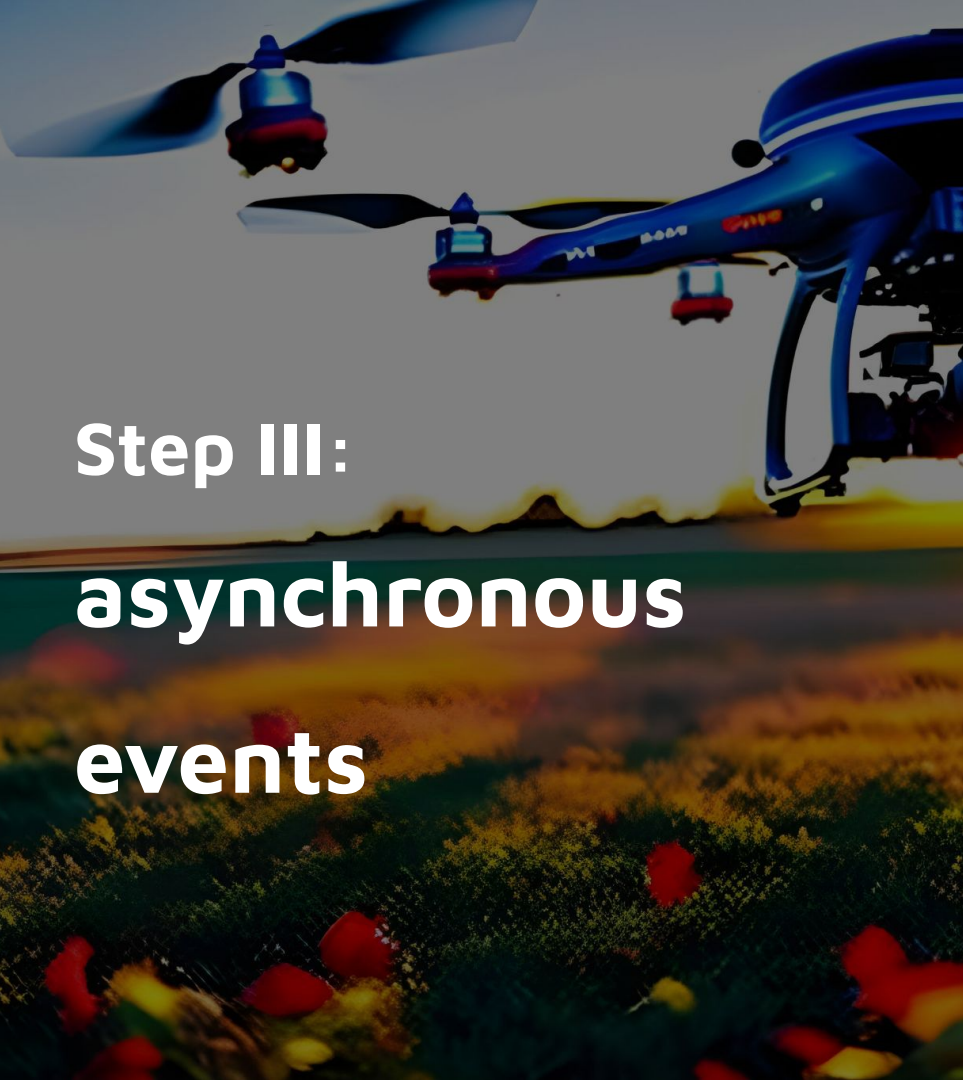
I/O, SoC devices

Async clocks

Derive from single source

Input values

Mock



Step III:

asynchronous events

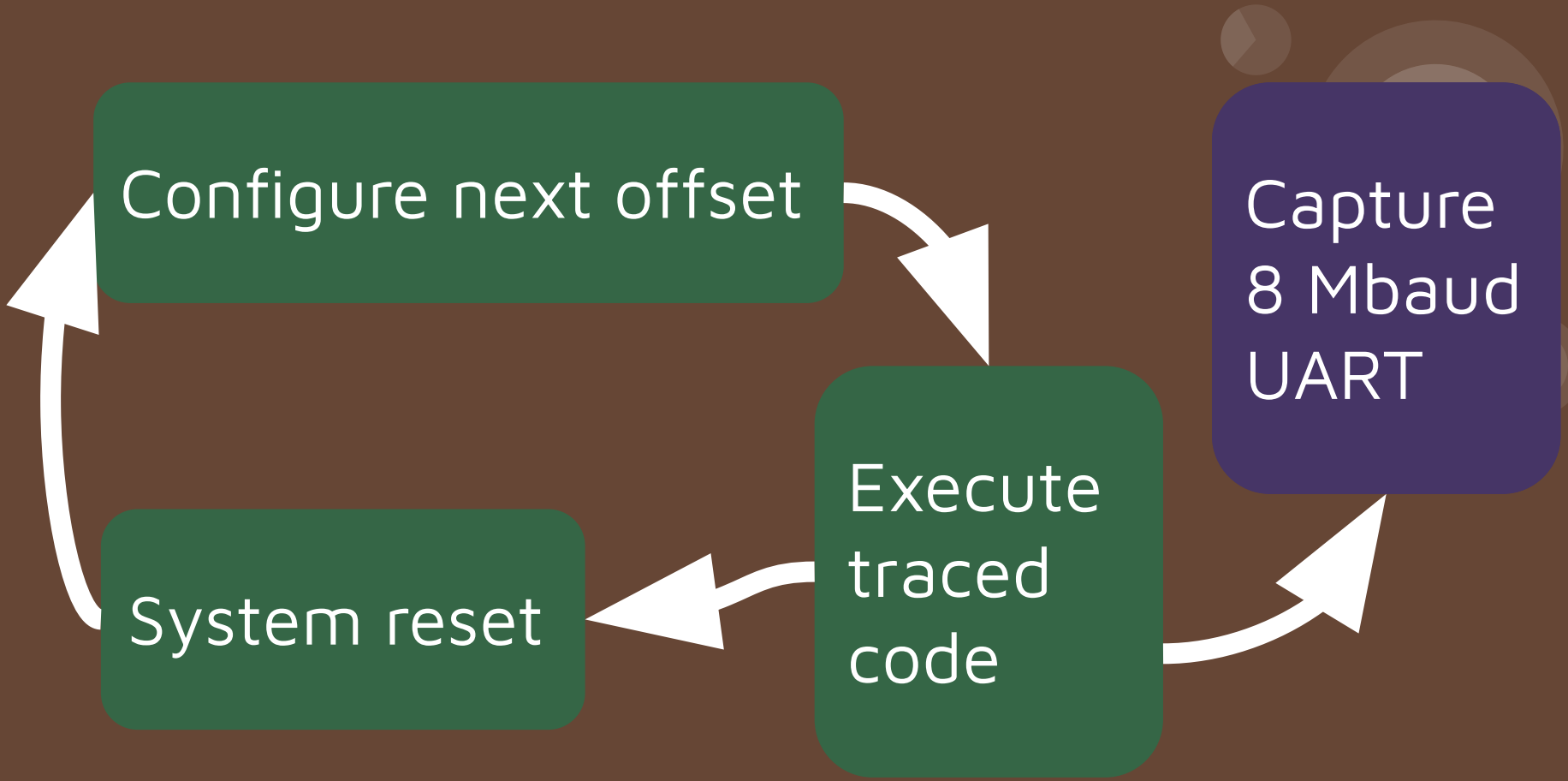
Do you need them?

No → **Mock**

Are they repeatable and predictable?

Yes → **Set new reference point**

No → **Trace only the part without them**


Configure next offset
Capture 8 Mbaud UART
Execute traced code
System reset

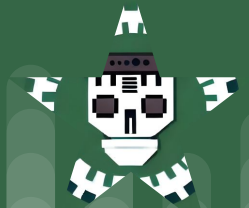

An interrupt here?
Wow! That is incredible. But... What values are written to the devices?

FrankenTrace
LSU trace

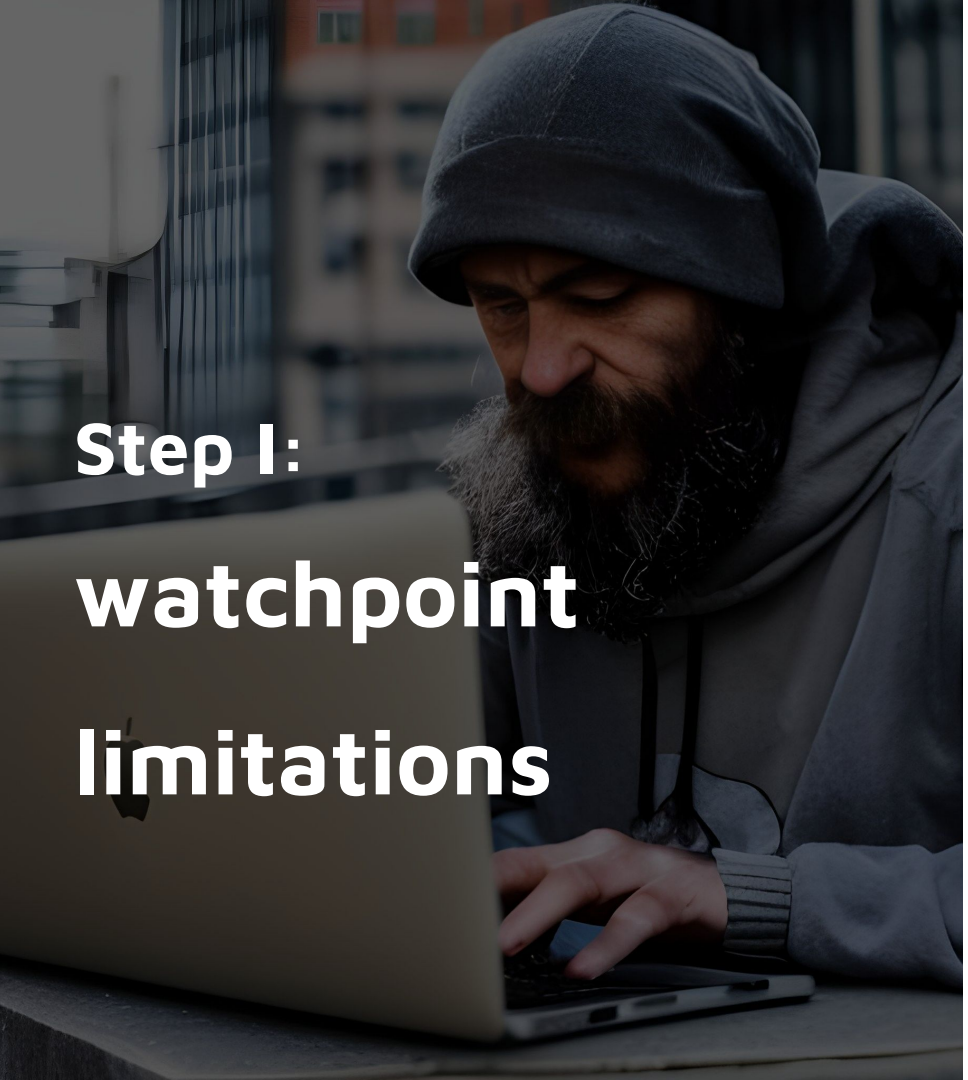








Debug Watchpoint and Trace unit can match memory transfer **address**



Step I:

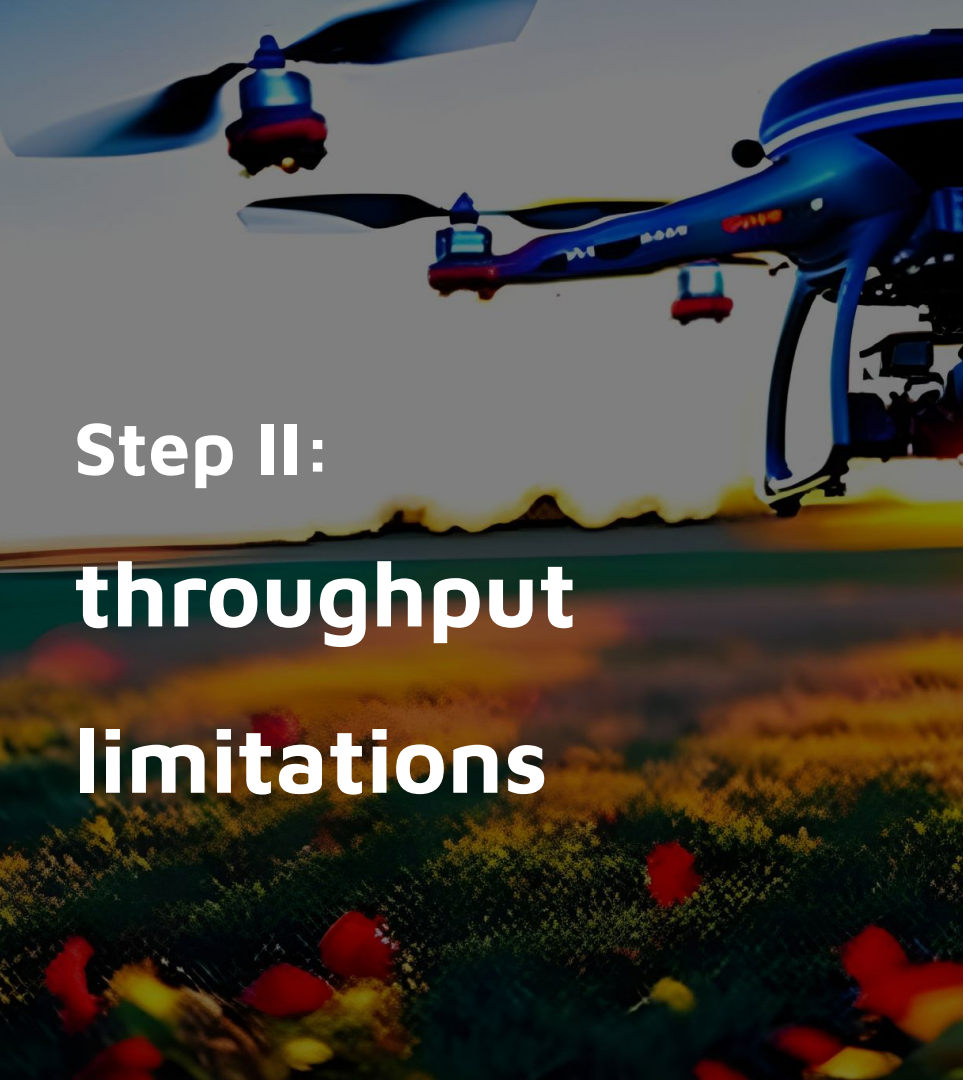
watchpoint limitations

16-bit range

PC/addr/data

4 watchpoints

Reset



Step II:

throughput limitations

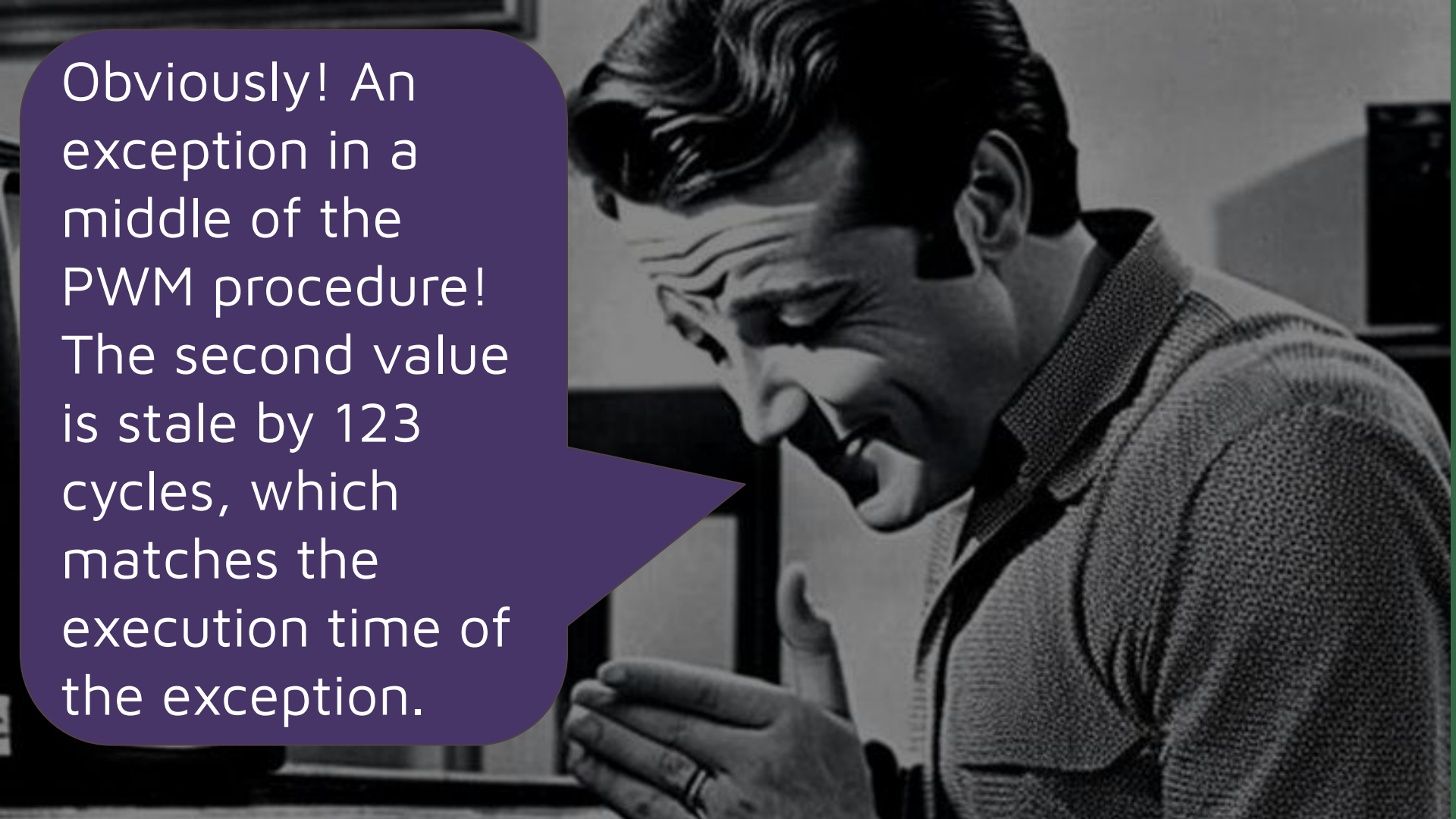
Do you need a noninvasive trace?

- Yes → **Accept inevitable holes**
- No → **Co-trigger some delay**

Trace interrupts and critical sections?

- Yes → **Fixup execution priorities**

value	cycle (with delays)	wait states	dir	size	PC (named)	address (named)
0x00006000	55891	?	Write	4	IOCPortConfigureSet+0x10	IOC::IOCFG20
0x00106000	45168	1	Read	4	GPIODirModeSet+0x2	GPIO::DOE31_0
0x00106000	46603	?	Write	4	GPIODirModeSet+0x10	GPIO::DOE31_0
0x00100000	48040	?	Write	4	ResetISR+0x54a	GPIO::DOUTCLR31_0
0x00006000	57387	?	Write	4	IOCPortConfigureSet+0x10	IOC::IOCFG13
0x00106000	49522	1	Read	4	GPIODirModeSet+0x2	GPIO::DOE31_0
0x00106000	51038	?	Write	4	GPIODirModeSet+0x10	GPIO::DOE31_0
0x00002000	52473	?	Write	4	ResetISR+0x556	GPIO::DOUTCLR31_0
0x00006000	58882	?	Write	4	IOCPortConfigureSet+0x10	IOC::IOCFG14
0x00106000	53949	1	Read	4	GPIODirModeSet+0x2	GPIO::DOE31_0
0x00106000	55384	?	Write	4	GPIODirModeSet+0x10	GPIO::DOE31_0
0x00004000	56821	?	Write	4	ResetISR+0x562	GPIO::DOUTCLR31_0


Obviously! An exception in a middle of the PWM procedure! The second value is stale by 123 cycles, which matches the execution time of the exception.

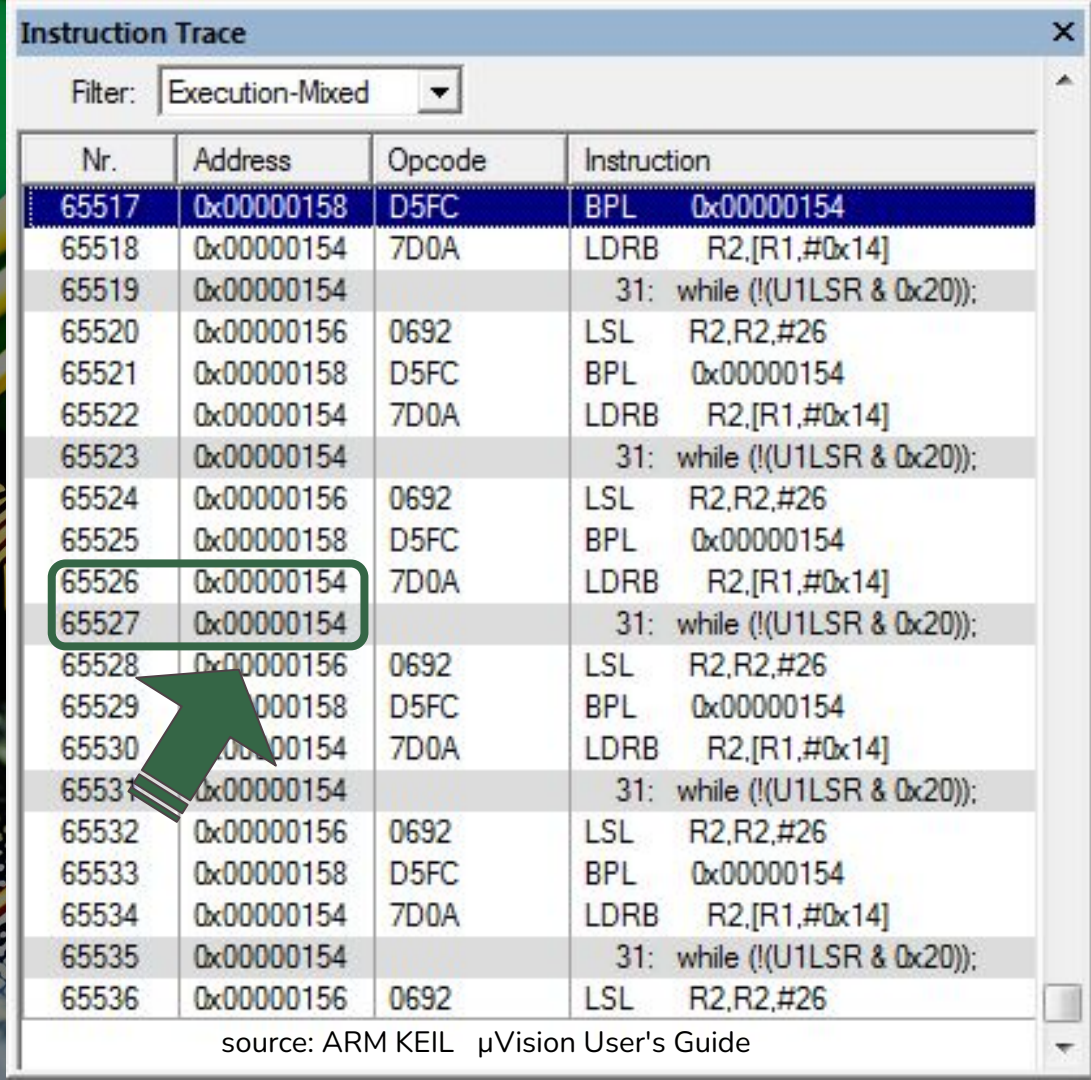
Thank you.
Questions?

Any possible similarity to real people is strictly a coincidence of the model's imagination of "Mike".

Backup slides (secret!)

Program Counter tracing

- exact profiling
- real-time
- no buffer limit

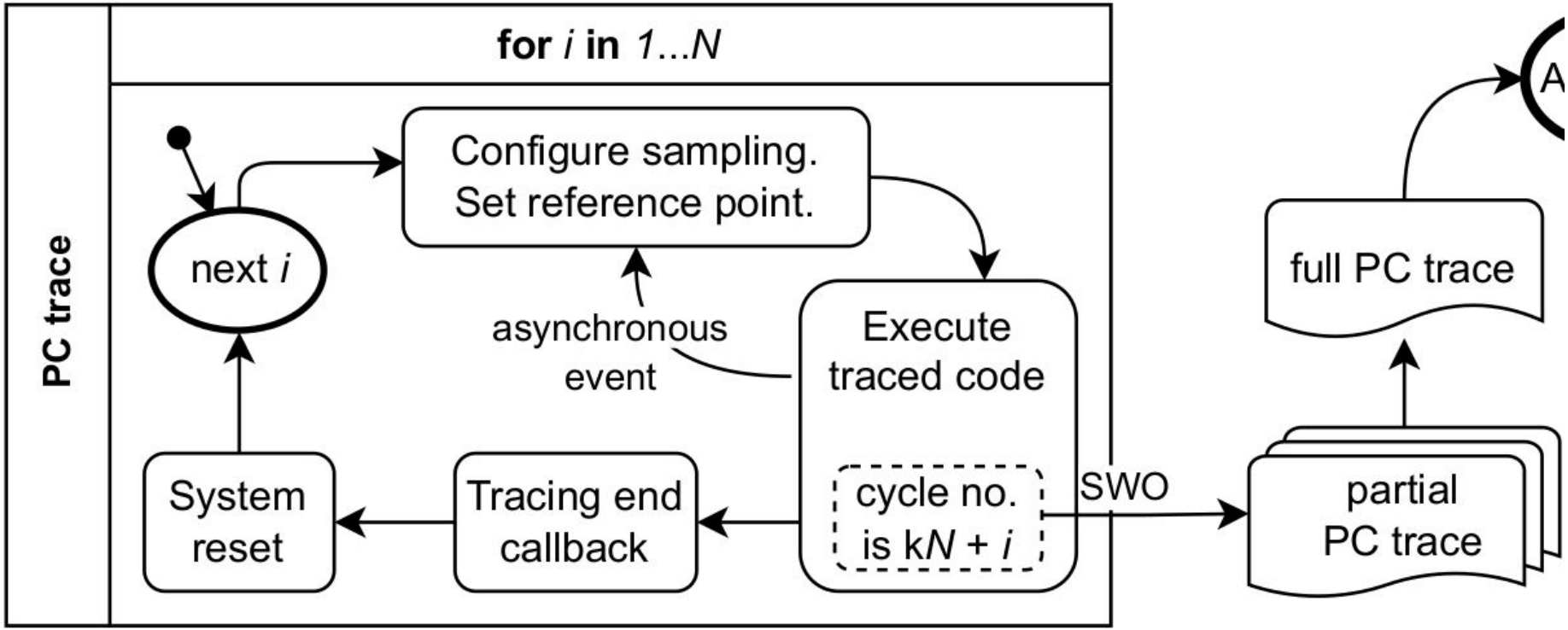


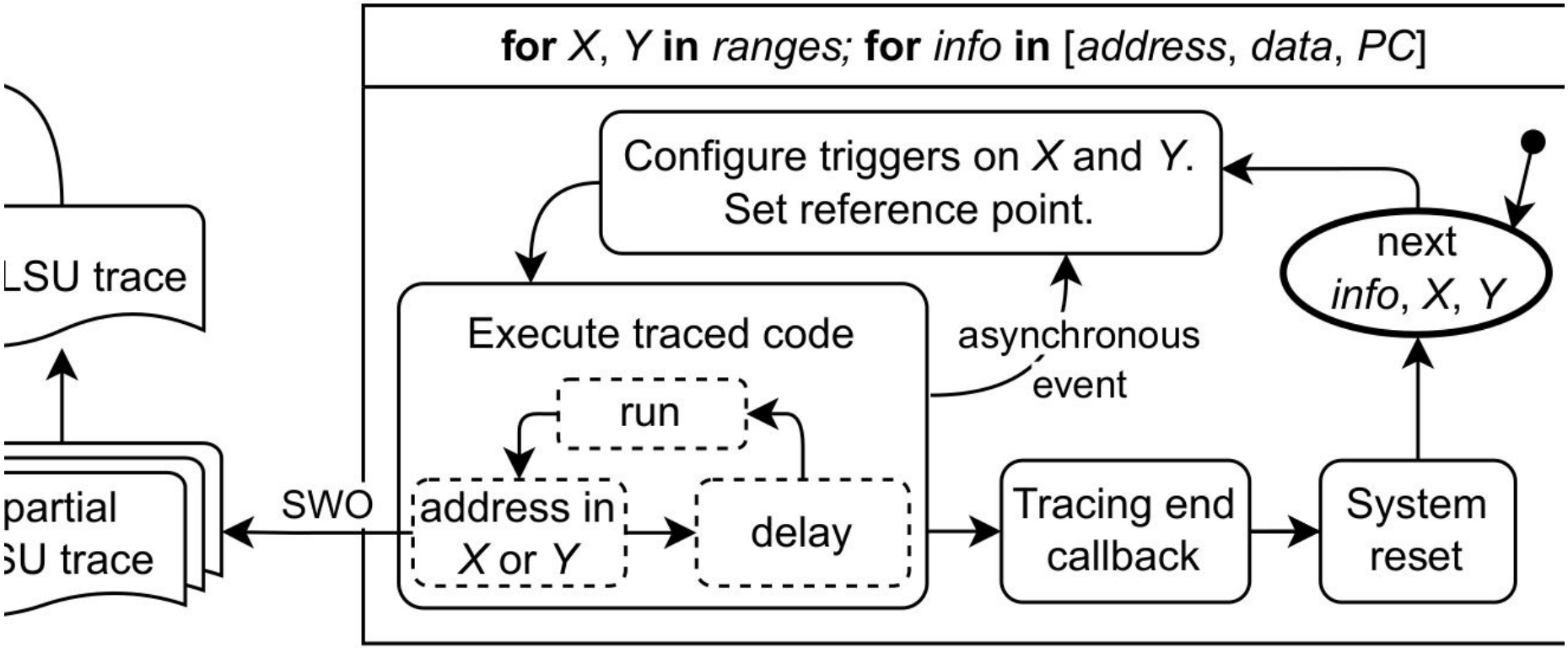
FrankenTrace

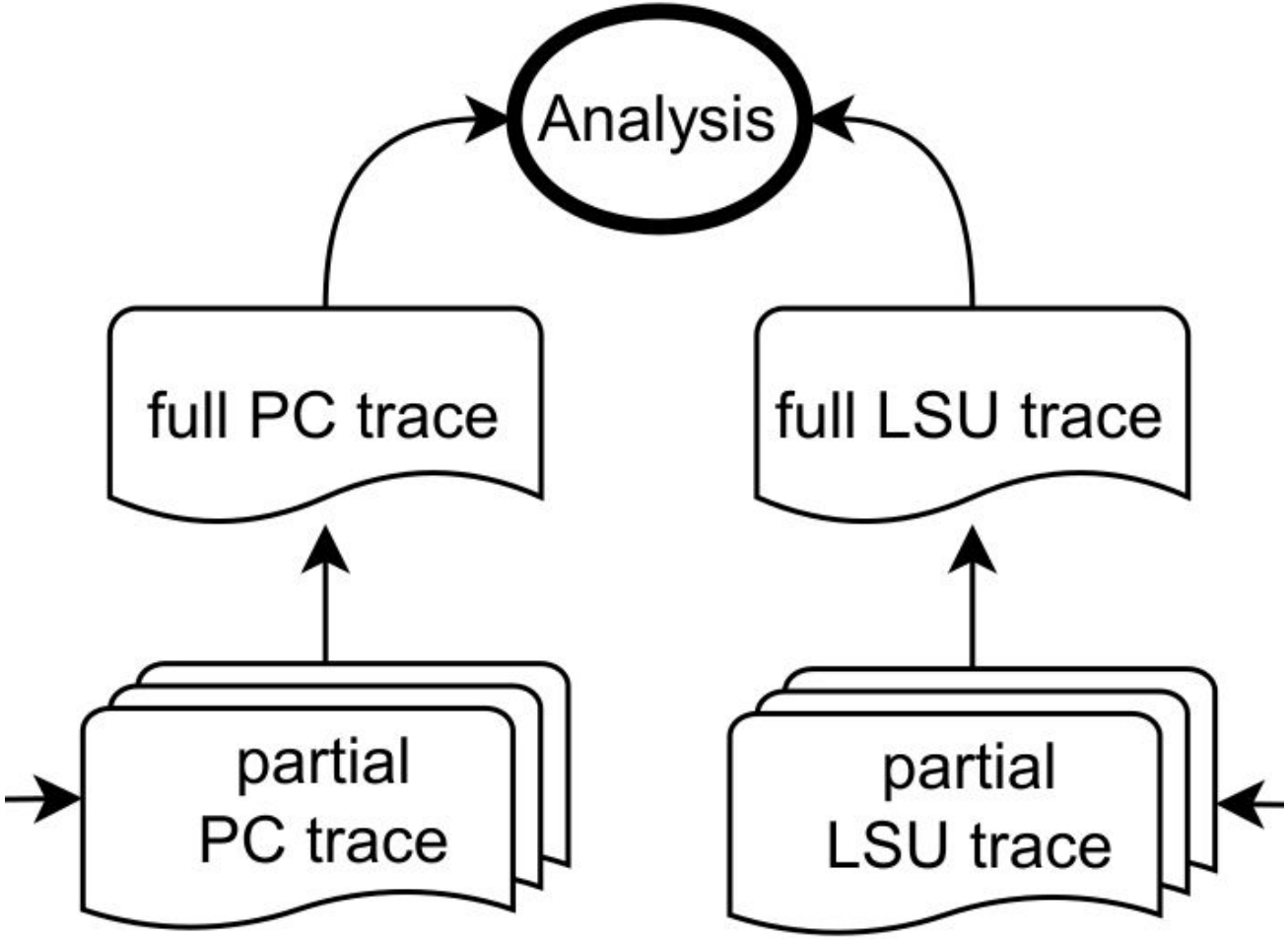
Low-Cost → $10 USB2UART

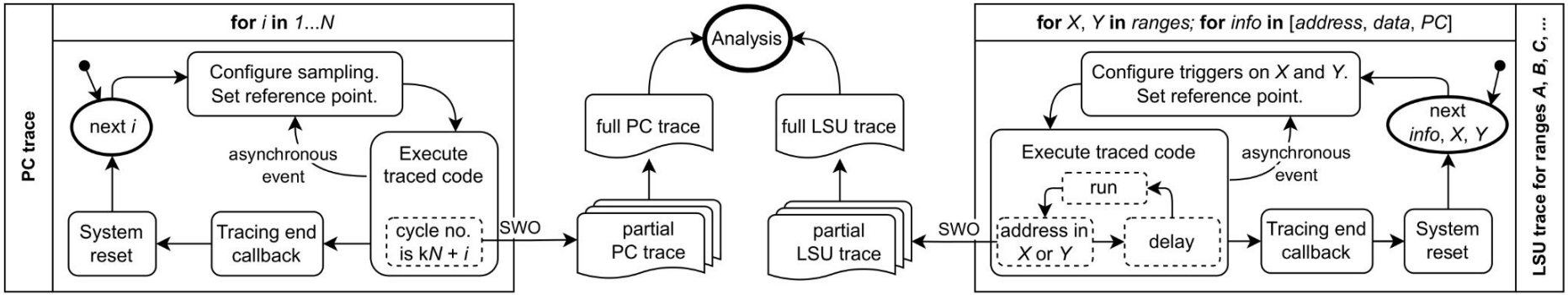
Widely Applicable → most Cortex-M

Cycle-Level → NECROMANCY


PC trace
for i in 1...N
Configure sampling.
Set reference point.
next i
asynchronous
event
Execute
traced code
cycle no.
is kN + i
SWO
System
reset
Tracing end
callback
partial
PC trace
full PC trace
A


for X, Y in ranges; for info in [address, data, PC]
Configure triggers on X and Y.
Set reference point.
next
info, X, Y
Execute traced code
asynchronous
event
run
address in
X or Y
delay
Tracing end
callback
System
reset
SWO
LSU trace
partial
SU trace


Analysis
full PC trace
full LSU trace
partial
PC trace
partial
LSU trace



todo put sigrok teaser here

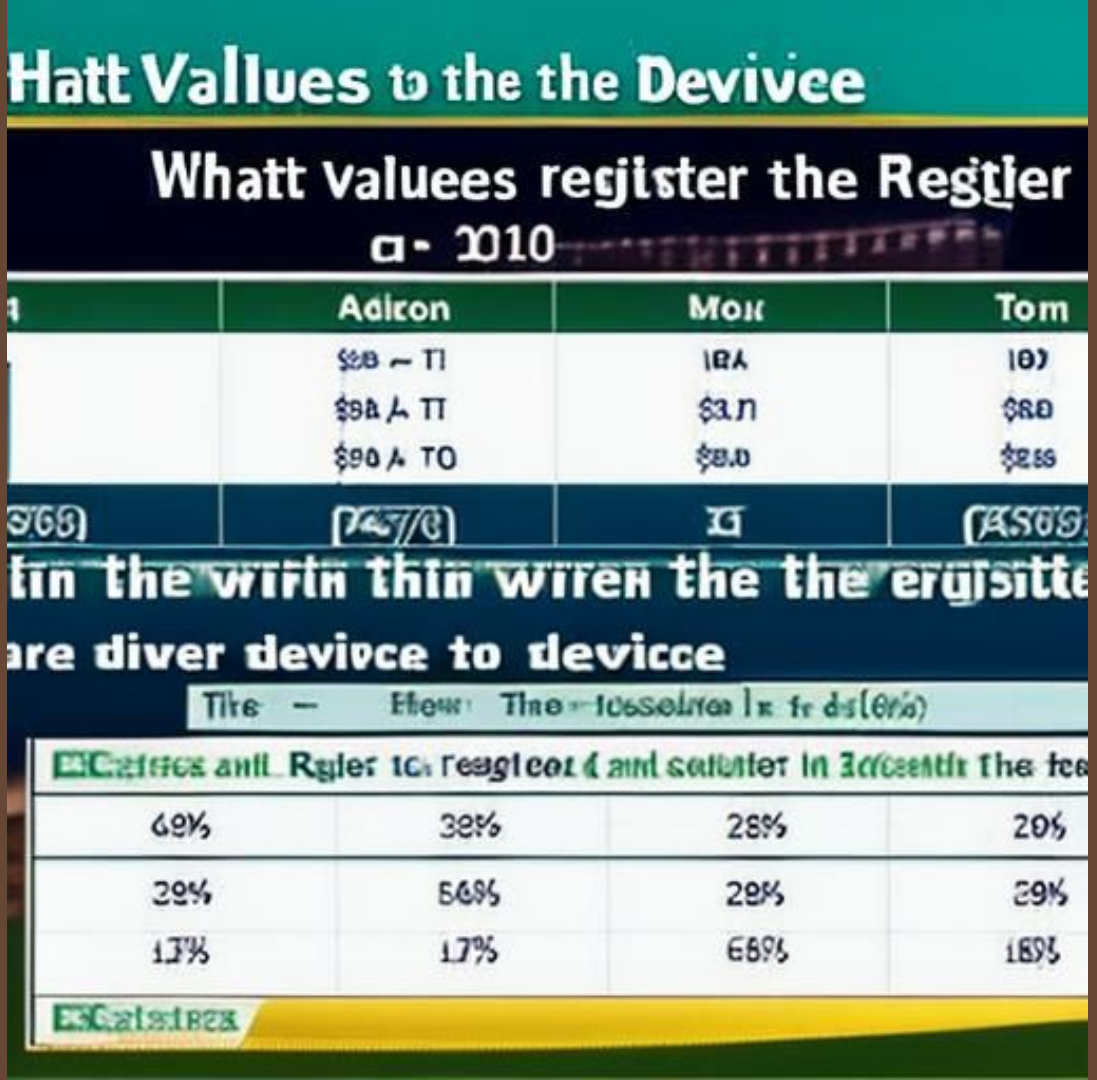
Table 2: DWT and ITM configurations for various output baudrates from a 48 MHz-clocked TI CC2650 that optimally use the available bandwidth. TPIU is configured to use SWO with UART 8N1 encoding and bypasses the packets formatter.

baud rate	N	TPIU prescaler	DWT_CTRL[12:0]
1 Mbaud	5120	47	0x1209
2 Mbaud	3072	11	0x1205
8 Mbaud	512	5	0x100f
24 Mbaud	192	1	0x1005
48 Mbaud	128	0	0x1003

Table 1: Low- and high-speed debug components featured in representative (as of Feb. 2023) Cortex-M3-based SoCs.

SoC series	low-	high-speed	SoC series	low-	high-speed
CC26x0	✓	✗	ADUCM360/3029	✗	✗
SAM3	✓	✗	PSoC 5LP/FM3	✓	some
STM32F100	✓	✗	STM32F103	✓	some
EFM32G/TG	✓	✗	EFM32LG/GG	✓	✓
LPC1300	✓	✗	LPC1800	✓	✓

Texas Instruments CC2650 SoC featured in 1KT and Indriya2 has no ETM and PTP.
STM32F103REY SoC featured in IoT-LAB's M3 and A8-M3 boards has no PTP pins.


Hatt Vallues to the the Devivee
Whatt valuees regjister the Regtler
a- 2010
Adicon
Mor
Tom
diver device to device

This is the end of the presentation.
You may now safely turn your computer off.