

# INTRODUCTION TO iSYSTEM TRACE



# Introduction to iSYSTEM Trace

## Contents

<b>01</b>	Why trace with iSYSTEM tools?	3
<b>02</b>	Trace Techniques	4
<b>03</b>	Trace - Overall concept	5
<b>04</b>	Trace Access Technologies	6
<b>05</b>	iSYSTEM winIDEA Analyzer	7
<b>06</b>	What to record?	8

# 01 Why trace with iSYSTEM tools?

Breakpoints and stepping are fundamental debug methods, however they change software behavior in real-time systems, and cannot observe timings.

On the other hand *hardware* trace provides **non-intrusive** and deepest possible insight into the application **without affecting real-time** behavior. It can be used to debug the most difficult, complex code defects as well as timing issues and offers a complete trace history capture of the program execution.

For recording the program execution and data events followed by high-level analysis, iSYSTEM has developed effective hardware and software tools:

- BlueBox On-Chip Analyzers
- winIDEA™ IDE Analyzer



iC5700 BlueBox with Active Probe



iC5700 BlueBox with "DTM" debug adapter

## 02 Trace techniques

Three main trace techniques are:

**Software Tracing** – Trace messages are inserted by the user software and often buffered in the CPU on-chip memory.

**Hardware Tracing** – Trace messages are generated by dedicated On-Chip Trace (OCT) logic and send off-chip via trace interface.

**Hybrid Tracing** – Combination of Software and Hardware tracing.

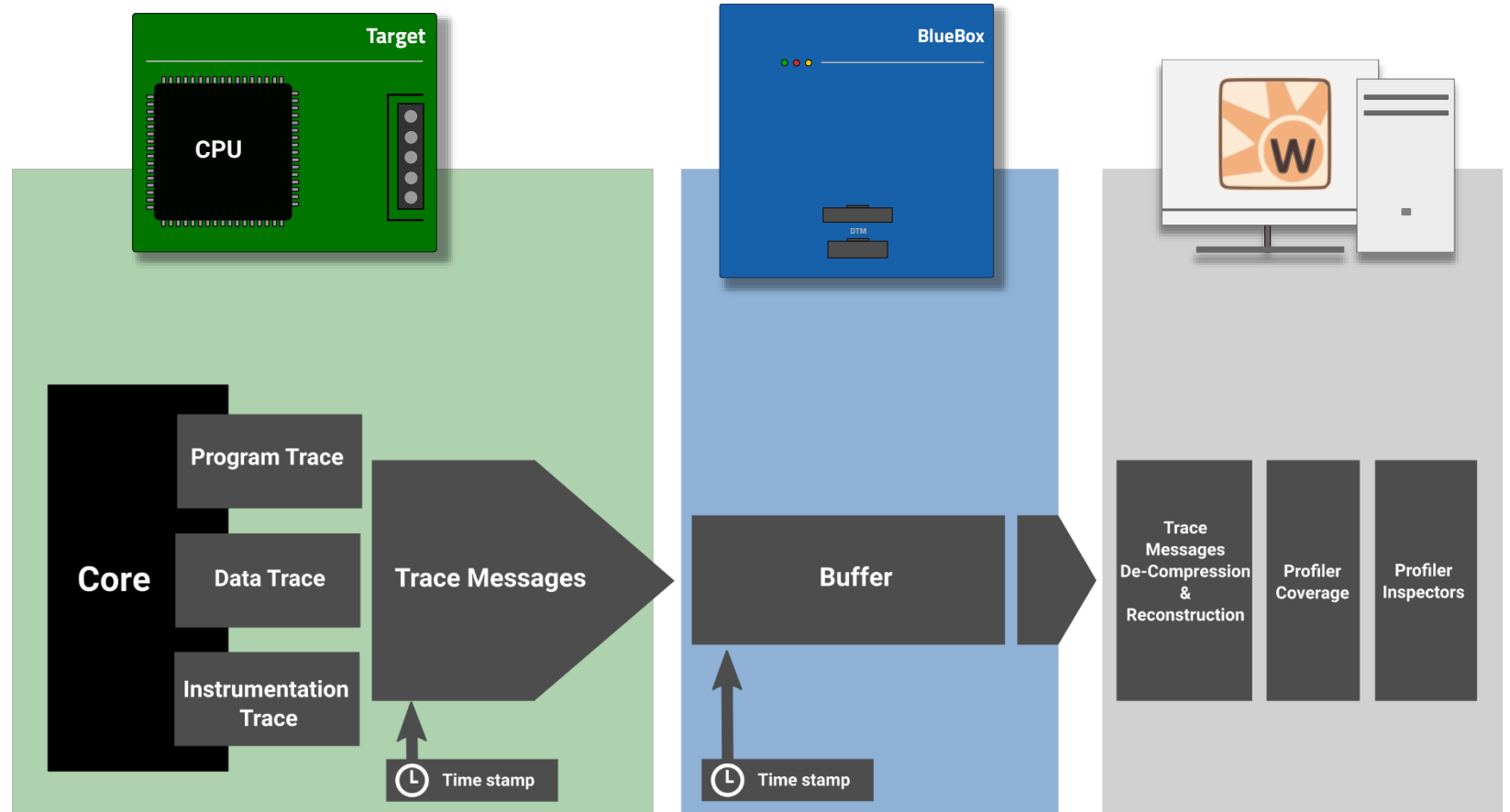
iSYSTEM with its hardware and software tools focuses on the hardware tracing technique.

Requirements	Advantages	Disadvantages
Software Tracing	No special hardware required	Significant computing Small number of trace objects Short trace duration
Hardware Tracing	<b>No computing overhead</b> <b>High number of trace objects</b> <b>Long trace duration</b>	<b>Dedicated hardware required</b>
Hybrid Tracing	Small overhead High number of trace objects Long trace duration	Software changes necessary

# 03 Trace - Overall concept

iSYSTEM Analyzer is a powerful tool that enables debugging with assistance of a hardware trace, which can record various information generated by the CPU:

- **Program Trace** – Recording instruction execution together with time information enables function execution time measurement.
- **Data Trace** – Recording data accesses, together with time information enables RTOS profiling. Combined with Program Trace it gives you an insight how your system is executing.
- **Instrumentation Trace** – Recording of user inserted events (e.g. measuring RTOS timings via hooks)
- **Time** – During recording time stamps are assigned to every trace message. Time stamps can be generated by the CPU or by the BlueBox.

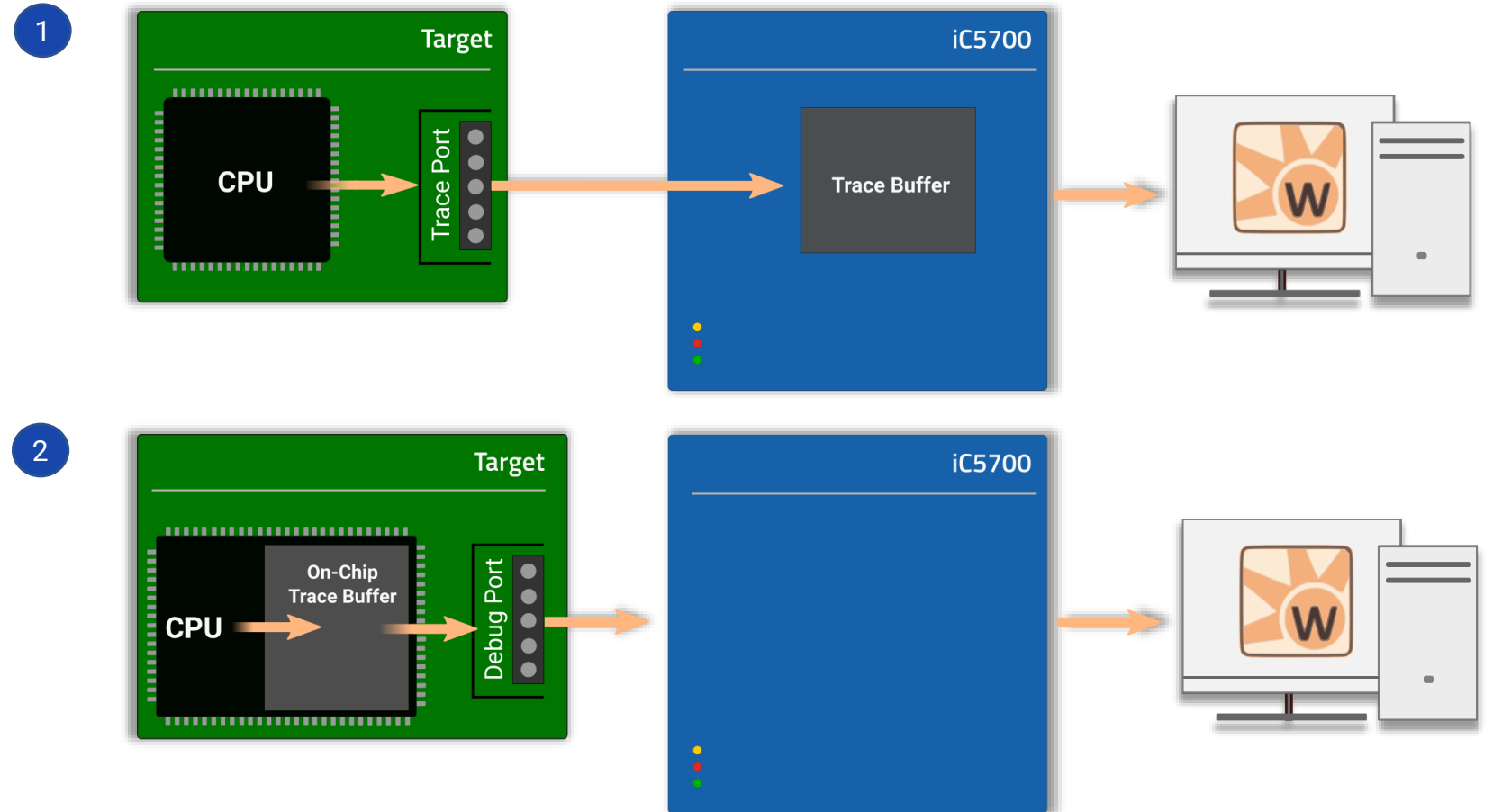


# 04 Trace Access Technologies

On-Chip Trace (OCT) logic observes the CPU activity and records the changes. Typically it implements triggers and qualifiers, which restrict tracing to certain code areas.

1. **Trace Port** – Trace messages are routed via the Trace Port to the BlueBox, where they are stored and then uploaded to the PC.

2. **On-Chip Trace Buffer (OCTB)** – Trace messages are read through standard Debug Port (e.g. JTAG) and uploaded to the PC via a BlueBox.



# 05 iSYSTEM winIDEA Analyzer

Analyzer combines trace, profiling and code coverage measurements in one window.

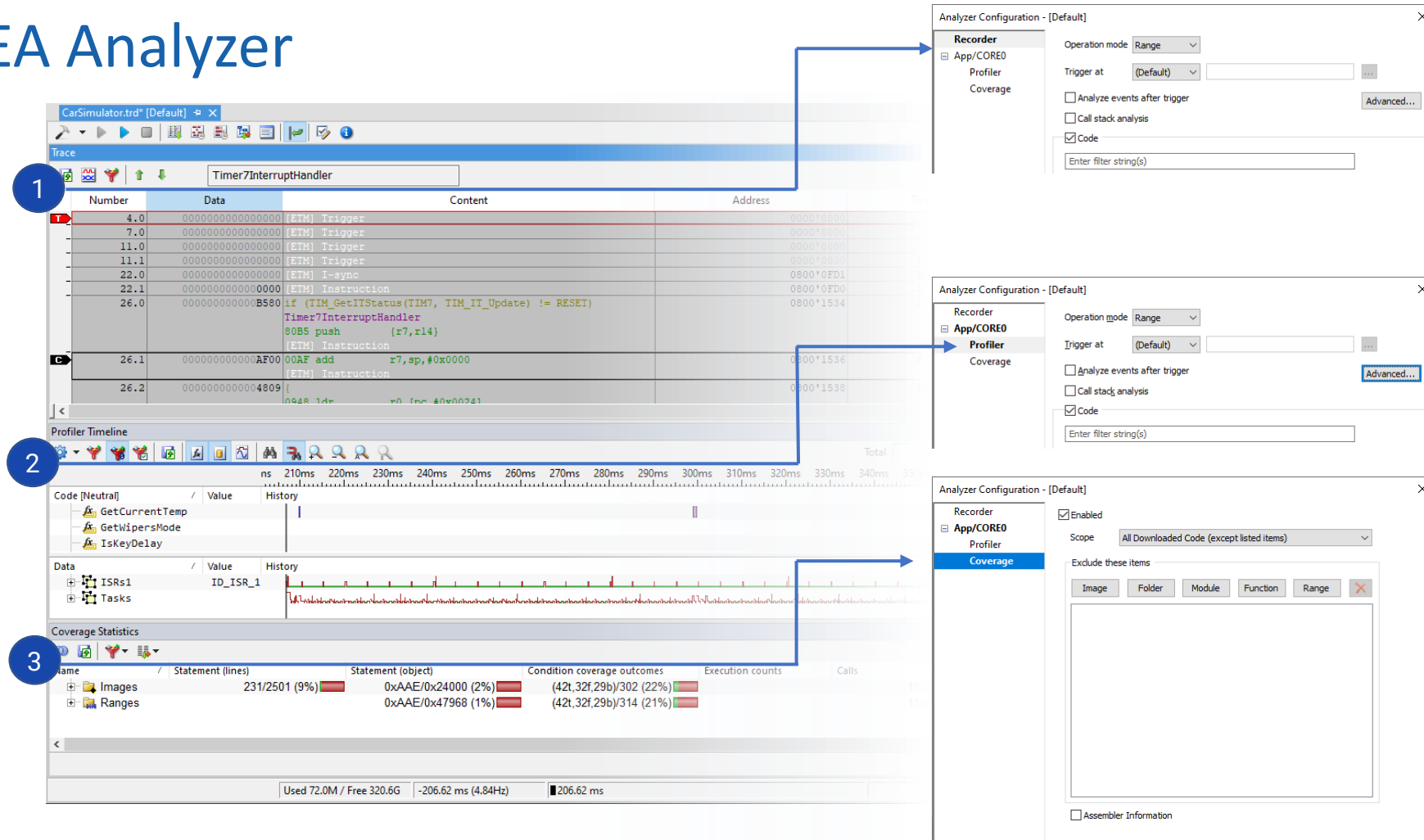
Captured trace data can be exported for post processing by third party tools.

Analyzer window provides three different views:

1. **Trace** is the primary window of the Analyzer and shows the executed code and data accesses with time information.

2. **Profiler** provides timeline and statistic information based on the traced data.

3. **Coverage** displays which parts of the code have been executed and tested.



The screenshot displays the iSYSTEM winIDEA Analyzer interface. The main window is divided into three sections, numbered 1, 2, and 3:

- 1. Trace:** Shows a list of executed instructions with columns for Number, Data, Content, and Address. The content includes instructions like [ETM] Trigger, [ETM] I-sync, and [ETM] Instruction.
- 2. Profiler:** Shows a timeline view with a graph of execution time. Below the graph, there are sections for Code (Neutral), Data, and Coverage Statistics.
- 3. Coverage:** Shows a table of coverage statistics for various code elements.

Three configuration windows are shown on the right side, each corresponding to a different analyzer feature:

- Recorder Configuration:** Shows settings for the Recorder, including Operation mode (Range), Trigger at (Default), and checkboxes for Analyze events after trigger, Call stack analysis, and Code.
- Profiler Configuration:** Shows settings for the Profiler, including Operation mode (Range), Trigger at (Default), and checkboxes for Analyze events after trigger, Call stack analysis, and Code.
- Coverage Configuration:** Shows settings for the Coverage analyzer, including Enabled (checked), Scope (All Downloaded Code (except listed items)), and a list of items to exclude (Image, Folder, Module, Function, Range).

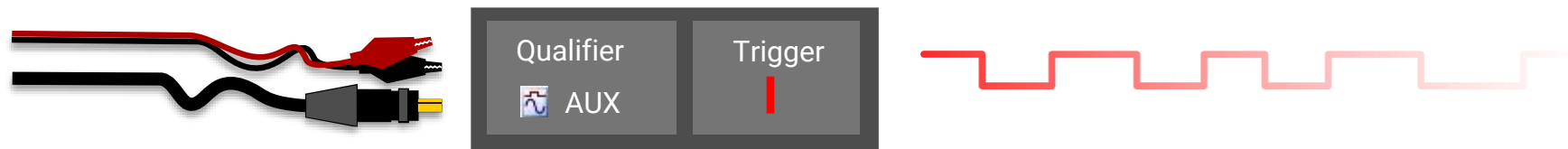
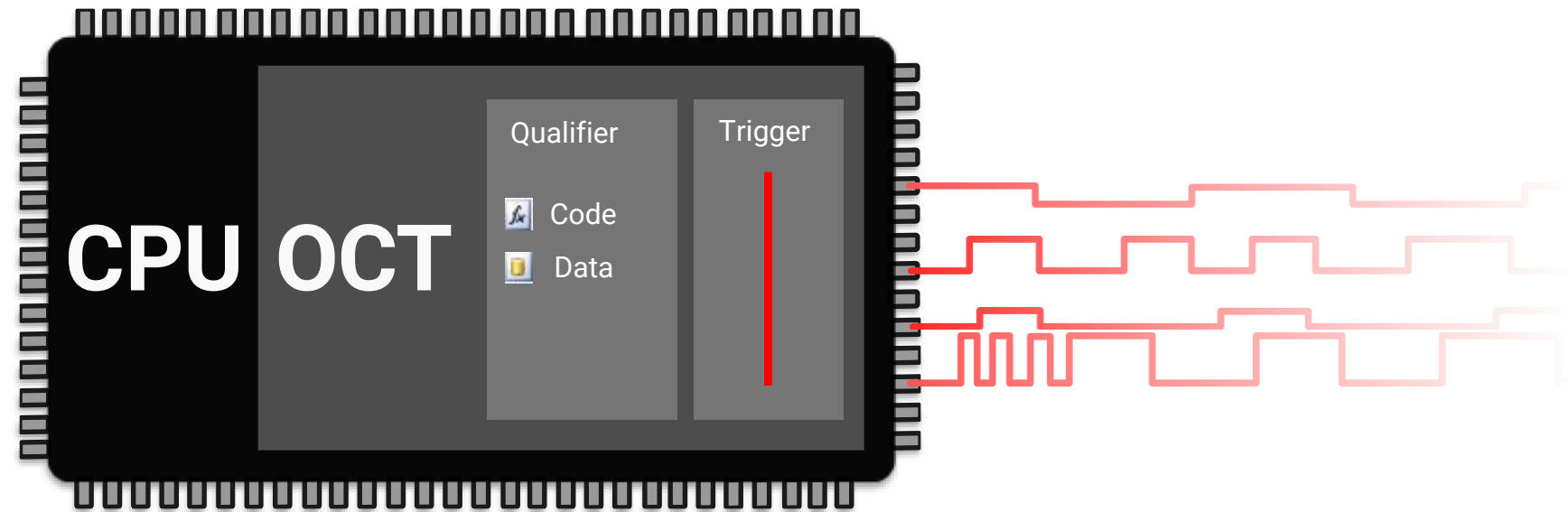
## 06 What to record? 1/2

Today's CPUs run at high frequencies and can generate a lot of trace information. Filtering mechanisms provide effective techniques to reduce the amount of trace data. On-Chip trace logic (OCT) provides:

**Qualifier** filters which information (Code, Data, AUX\*) is traced.

**Trigger** defines starting or end point of a trace recording.

\*AUX Profiling is available in a combination with IOM6 ADIO and IOM6 CAN/LIN.



Refer to iSYSTEM internet site to learn more about our products for AUX and Network profiling – [IOM6 ADIO](#) and [IOM6 CAN/LIN](#).

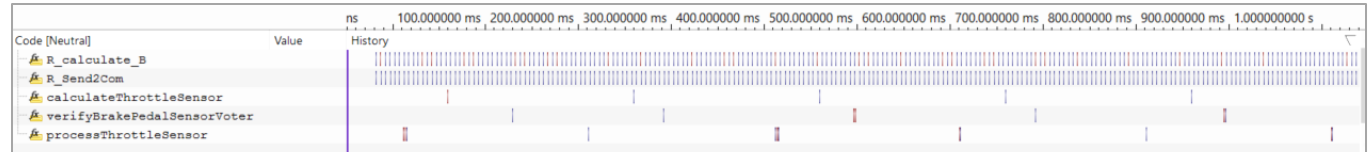


# 06 What to record? 2/2

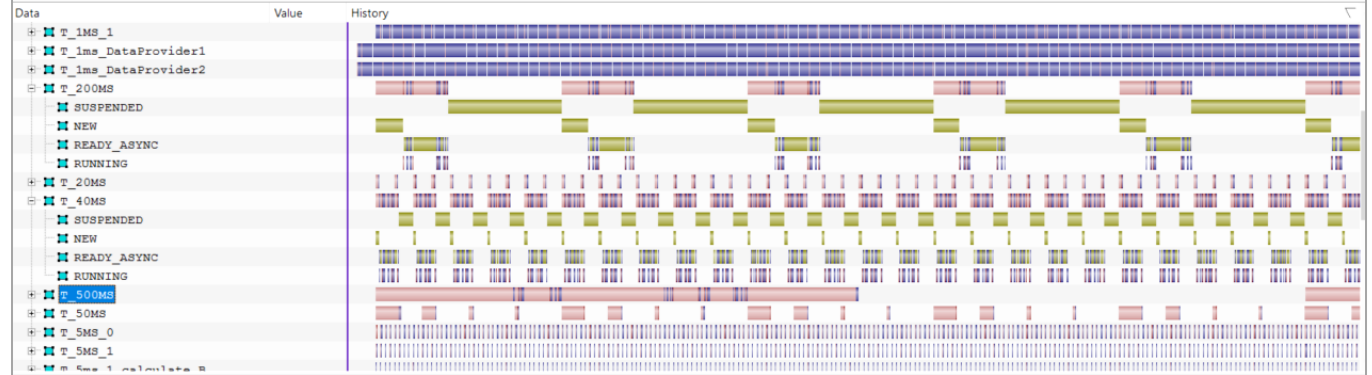
Profiler Timeline displays high-level trace information in the timeline view.

- Functions
- Data
- Instrumented events
- Digital / analog events
- Network protocol messages

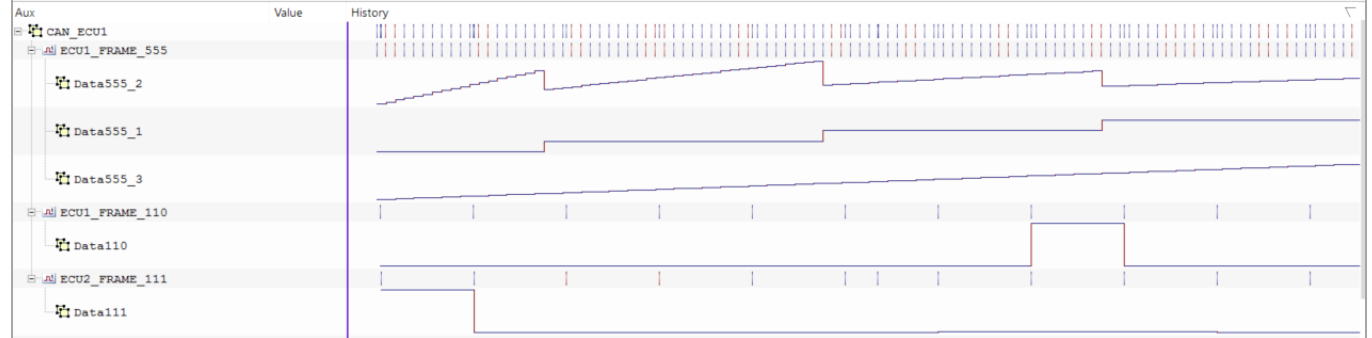
Code



Data



AUX





# Further Reading

For more information refer to our online resources:

- Hardware Solution:
  - On-Chip Analyzers [BlueBox](#)
  - IOM Modules [Analog/Digital and Network Trace](#)
- winIDEA Online Help:
  - iSYSTEM [Analyzer](#) tool
- [Knowledge Base](#)