

Embedded System Design Techniques™

Debugging Real-time Embedded Software – Hands-on

Session 3: Debugging the ARM Cortex-M Microcontroller

July 13th, 2016

Jacob Beningo, CSDP

Course Overview

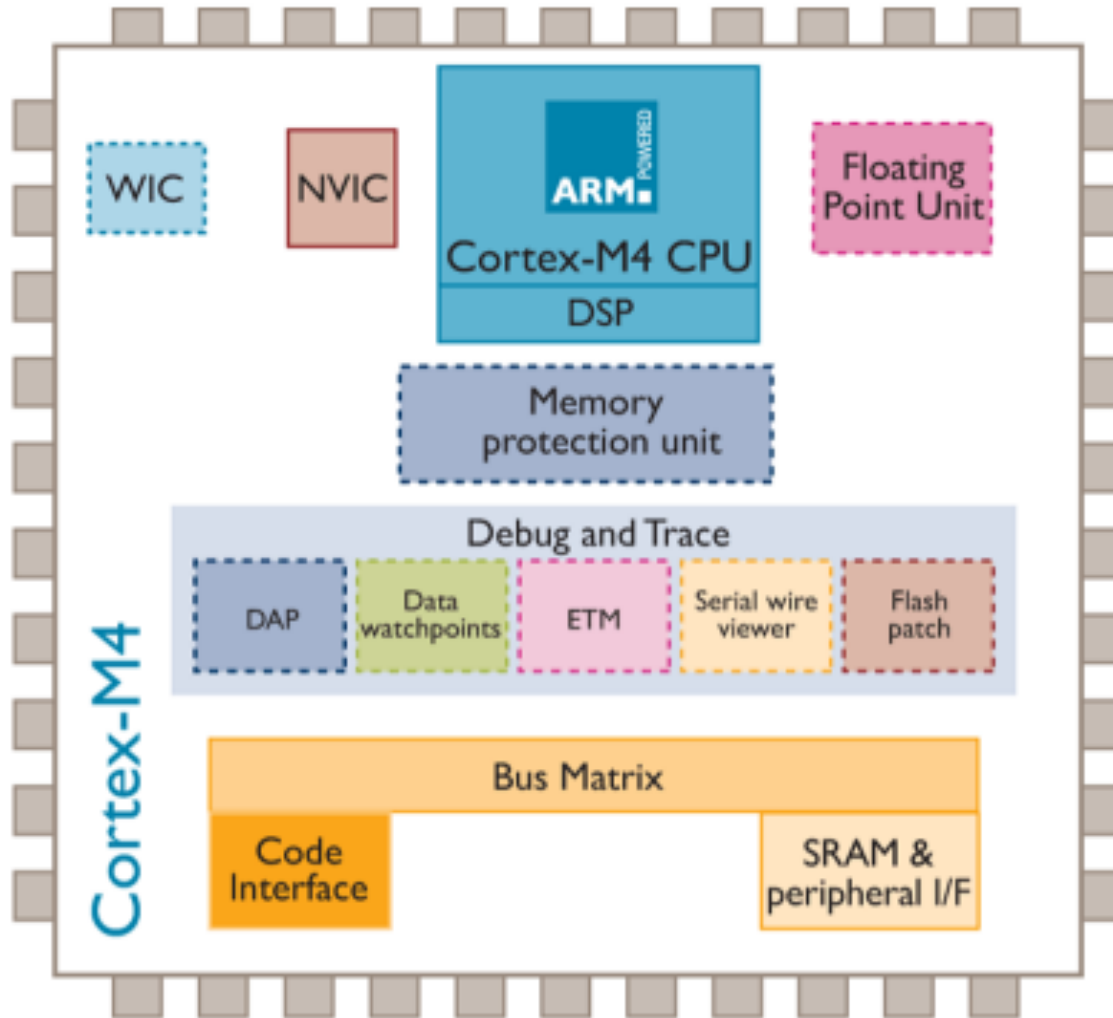
- Introduction to Debugging Real-time Embedded Systems
- Foundational Debugging Techniques
- **Debugging the ARM Cortex-M Microcontroller**
- Utilizing Systems Viewers and Trace tools to Debug Firmware
- Tips and Tricks for Debugging Embedded Systems

Session Overview

- The ARM Core
- Debugging Capabilities
- Debug and Trace Interfaces
- Trace Capabilities
- Statistical Profiling with SWV



The ARM Core



Debugging Capabilities

Invasive Debugging – features that need to stop the processor or change the program execution flow significantly

- Program halting, single stepping, reset, resume
- Breakpoints
- Data watchpoints
- Internal register accesses
- Debug monitor exception
- ROM based debugging using flash patch logic

Non-invasive debugging – features that have no or very little effect on the program flow

- On the fly memory / peripheral access
- Instruction trace
- Data trace
- Software generated trace
- Profiling

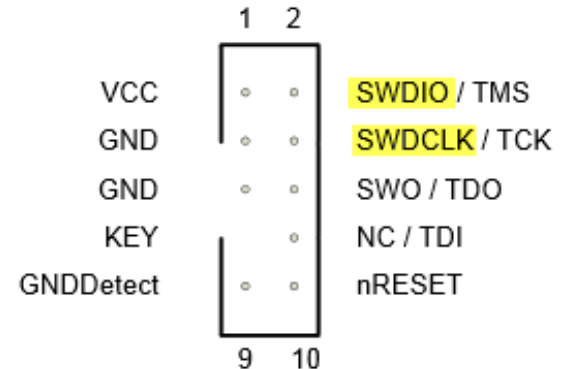
Debug and Trace Interface

Debug Interface

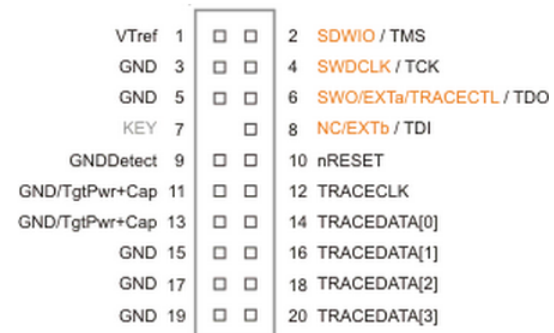
- Joint Test Action Group (JTAG)
- Serial Wire Debugger (SWD) (ARM protocol)

Available types of trace

- Serial Wire Viewer (SWV)
 - Single data pin
 - Limited to 2 Mbps
 - Used with SWD
- Trace Port Interface (TPI)
 - 4 Data pins and a clock
 - Separate connector from SWV



Cortex 20-pin 0.05" JTAG/SWD/ETM Connector



Trace Capabilities

What information can be exported?

- Exception events
 - Information associated with a data watchpoint event (data value, program counter, address values, etc)
- Events from profiling counters
 - Software generated trace data (instrumentation trace: ex printf)
 - Timestamp information – for each trace data, you can enable a timestamp packet to go with it so that the timing of events can be reconstructed by the debug host.

Data Watchpoint and Trace (DWT)

Debug Event Generation

- Generates debug event (like a data watchpoint) that can be used to halt the processor or generate the debug monitor exception
- Used by ETM to start/stop trace

Data trace

- Generate data trace packets when the comparator matches that contain values, data addresses or PC values.

PC Sampling

- Generate periodic PC sampling to the trace stream for profiling
- Allows debugger to occasionally sample the PC
- Can be monitored in two ways
 - Output pc samples via trace periodically
 - Debugger can read the register directly
- Major uses
 - Estimating functions that have been executed
 - Determining execution path
 - Estimate how much time is spent in each function and generate approximate min, avg and max values

Profiling

- Counters are available to generated trace packets and collect information about the system.

Exception trace

- DWT can generate a trace packet at exception enter and exit. When used with ITM can provide detailed timing of how much time was spent in the exception.

Instrumentation Trace Macrocell (ITM)

Multiple functionalities

- Software trace – software can directly write to the trace port and registers
- Trace packet merger for the DWT, stimulus port and timestamp packet
- Can generate timestamp packets that are inserted in the trace stream to help the debugger reconstruct the timing of the system

Can only be used if the ITM module is included on the chip and the trace port is used with a debugger that supports trace.

Main purpose of ITM is for debug message output such as printf

- Contains 32 stimulus ports allowing multiple processes to output to multiple ports
- Using the ITM does not cause much of a delay in program execution unlike traditional debug messages
- Output can be directed to the trace port interface or to the SWV
- Debug message code be left in the final application because if TRCENA is disabled, the ITM will be inactive and not transmit the messages. **(SECURITY RISK!?)**
- Debugger normally sets up the ITM for developers

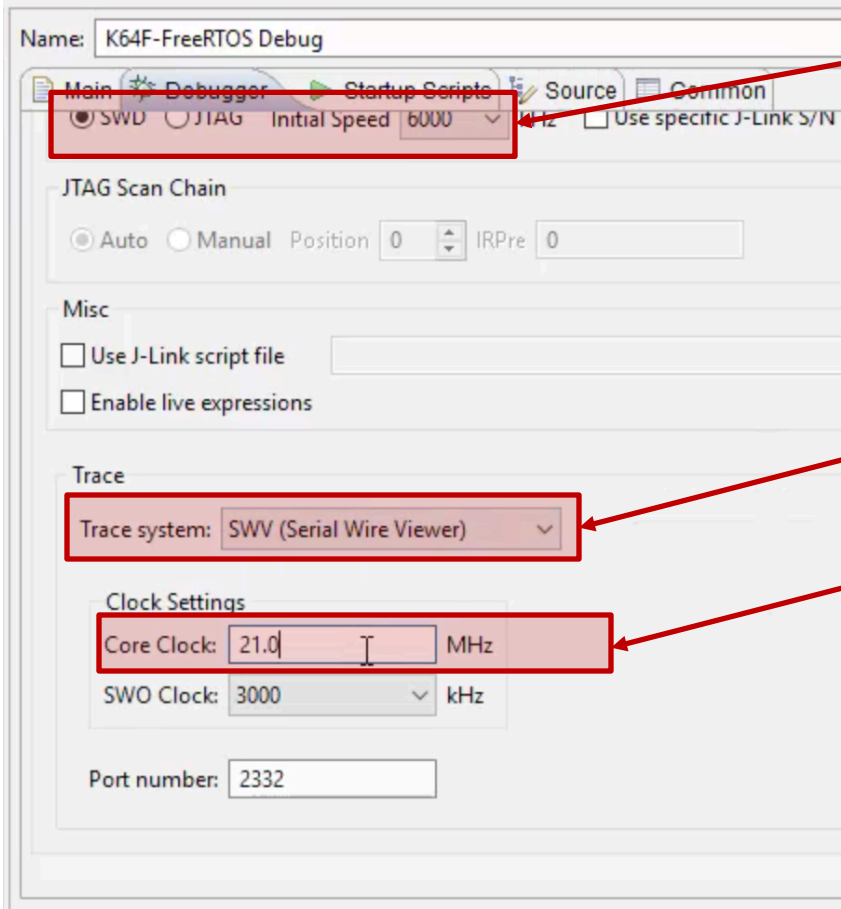
Embedded Trace MacroCell(ETM)

Instruction Trace – needs an optional on-chip hardware component called Embedded Trace MacroCell (ETM).

- Provides entire instruction trace history that can be reconstructed by the host.
- Can verify code coverage of the application during testing.
- Useful for code coverage, complex software bugs, performance analysis.
- Requires special interface
- Requires advanced (more expensive) debug probe

Statistical Profiling with SWV

1



Select SWD Interface

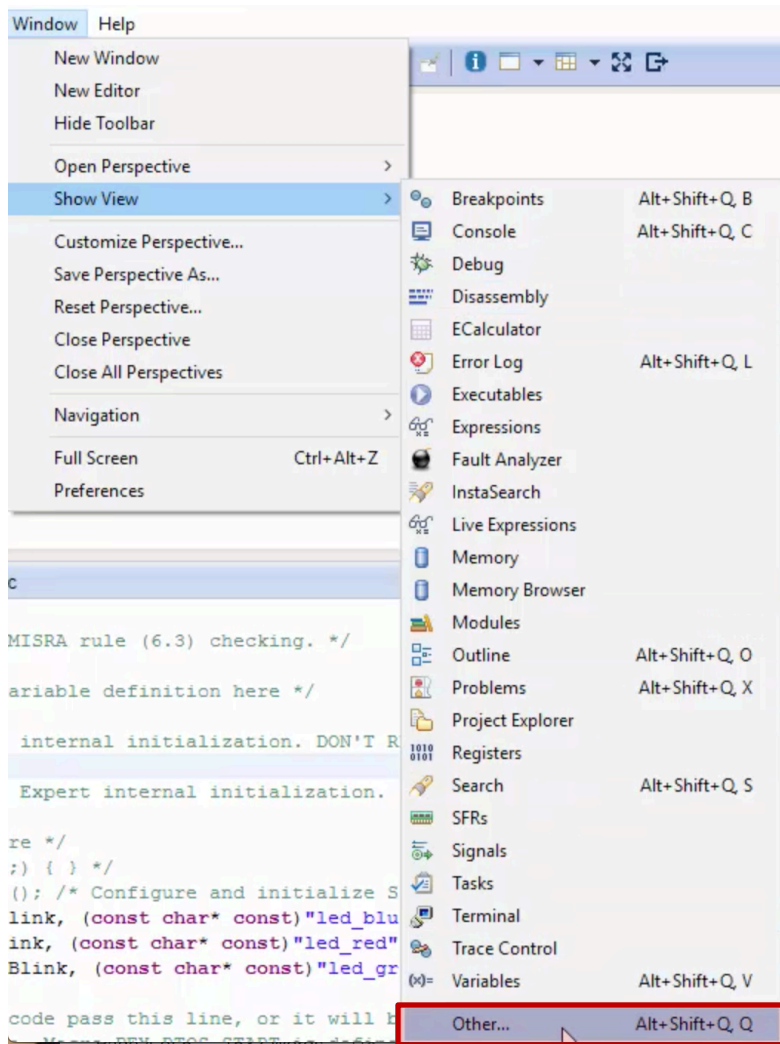
SWV Option from dropdown

Default is 8 MHz! What is your clock speed?

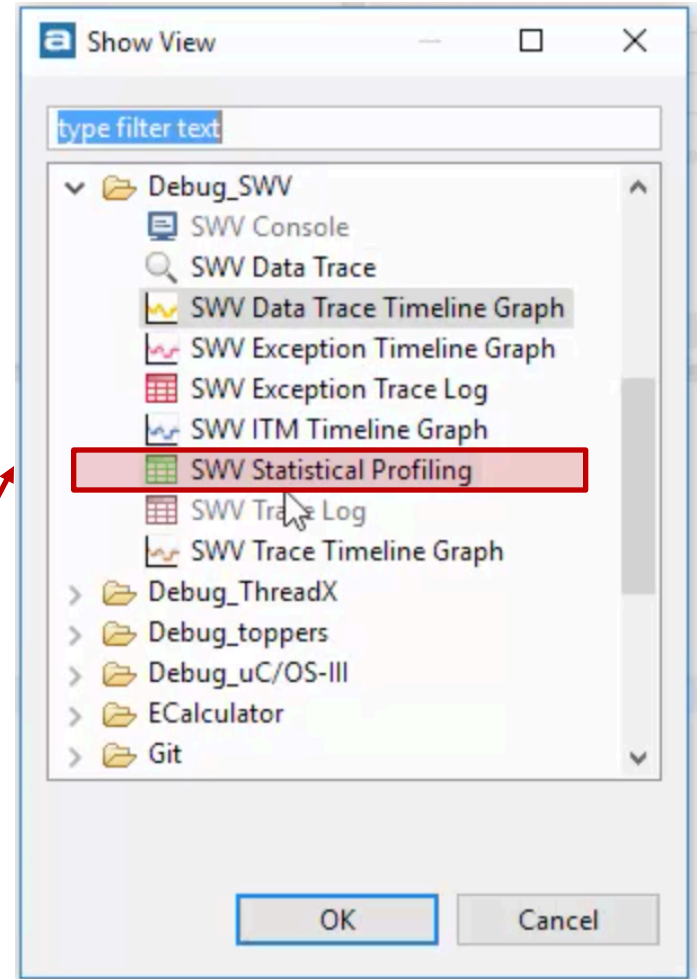
Run the debugger!

Statistical Profiling with SWV

2



3



Statistical Profiling with SWV

Log Console Profile

4 Settings 6 Record

Function	% in use	Samples	Start address	Size

Overflow packets: 0 PC Samples: 0

Statistical Profiling with SWV

5

Enable

Resolution

Clock Settings
Core Clock: 21 MHz
Clock Prescaler: 7
SWO Clock: 3000.0 kHz

Trace Events
 CPI: Cycles per instruction
 EXC: Exception overhead
 SLEEP: Sleep cycles
 LSU: Load store unit cycles
 FOLD: Folded instructions
 EXETRC: Trace Exceptions

PC Sampling
 Enable Resolution: 10240 Cycles/sample

Timestamps
 Enable Prescaler: 64

Data Trace

Comparator 0
 Enable
Var/Addr: 0x0
Access: Read/Write
Size: Word
Generate: Data Value

Comparator 1
 Enable
Var/Addr: 0x0
Access: Read/Write
Size: Word
Generate: Data Value

Comparator 2
 Enable
Var/Addr: 0x0
Access: Read/Write
Size: Word
Generate: Data Value

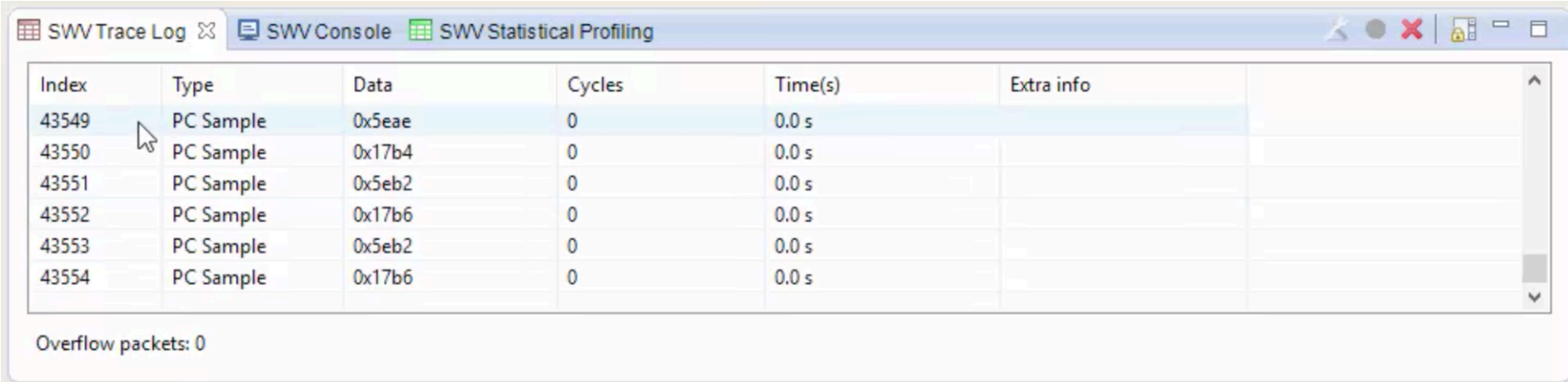
Comparator 3
 Enable
Var/Addr: 0x0
Access: Read/Write
Size: Word
Generate: Data Value

ITM Stimulus Ports
Enable port: 31 24 23 16 15 8 7 0
Privileged only ports: Port 31..24 Port 23..16 Port 15..8 Port 7..0

OK Cancel

Statistical Profiling with SWV

7 Run the code

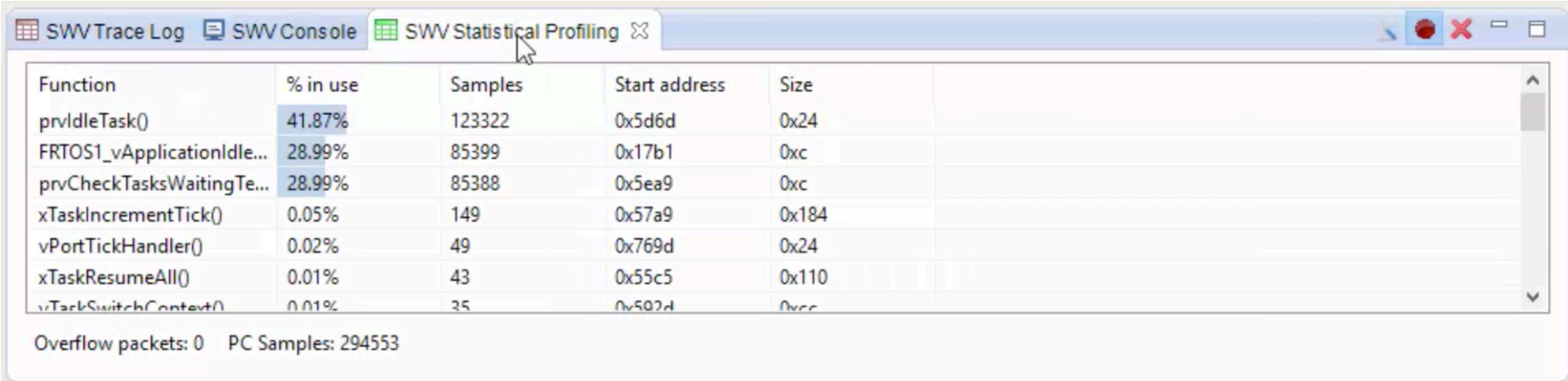


The screenshot shows the SWV Statistical Profiling window with a table of PC samples. The table has columns for Index, Type, Data, Cycles, Time(s), and Extra info. The data shows several PC samples with 0 cycles and 0.0 seconds of time.

Index	Type	Data	Cycles	Time(s)	Extra info
43549	PC Sample	0x5eae	0	0.0 s	
43550	PC Sample	0x17b4	0	0.0 s	
43551	PC Sample	0x5eb2	0	0.0 s	
43552	PC Sample	0x17b6	0	0.0 s	
43553	PC Sample	0x5eb2	0	0.0 s	
43554	PC Sample	0x17b6	0	0.0 s	

Overflow packets: 0

8 Pause



The screenshot shows the SWV Statistical Profiling window with a table of function usage. The table has columns for Function, % in use, Samples, Start address, and Size. The data shows the percentage of time spent in various functions.

Function	% in use	Samples	Start address	Size
prvIdleTask()	41.87%	123322	0x5d6d	0x24
FRTOS1_vApplicationIdle...	28.99%	85399	0x17b1	0xc
prvCheckTasksWaitingTe...	28.99%	85388	0x5ea9	0xc
xTaskIncrementTick()	0.05%	149	0x57a9	0x184
vPortTickHandler()	0.02%	49	0x769d	0x24
xTaskResumeAll()	0.01%	43	0x55c5	0x110
vTaskSwitchContext()	0.01%	35	0x592d	0xc

Overflow packets: 0 PC Samples: 294553

15

Additional Resources

- Download Course Material for
 - Updated C Doxygen Templates (Sept 2015)
 - Example source code
 - Templates
 - YouTube Videos
- Microcontroller API Standard
- EDN Embedded Basics Articles
- Embedded Bytes Newsletter
 - <http://bit.ly/1BAHYXm>



From www.beningo.com under






- Blog > Debugging Realtime Embedded Systems

The Lecturer – Jacob Beningo



Jacob Beningo
Principal Consultant

Social Media / Contact

-  : jacob@beningo.com
-  : 248-719-6850
-  : Jacob_Beningo
-  : Beningo Engineering
-  : JacobBeningo

EDN : Embedded Basics

CONSULTING

- Secure Bootloaders
- Code Reviews
- Architecture Design
- Real-time Software
- Expert Firmware Analysis

EMBEDDED TRAINING



www.beningo.com