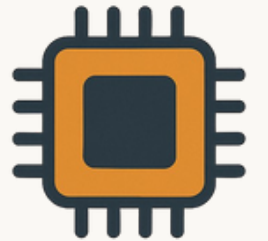

Embedded Debugging Techniques



*Hunt bugs
efficiently!!*

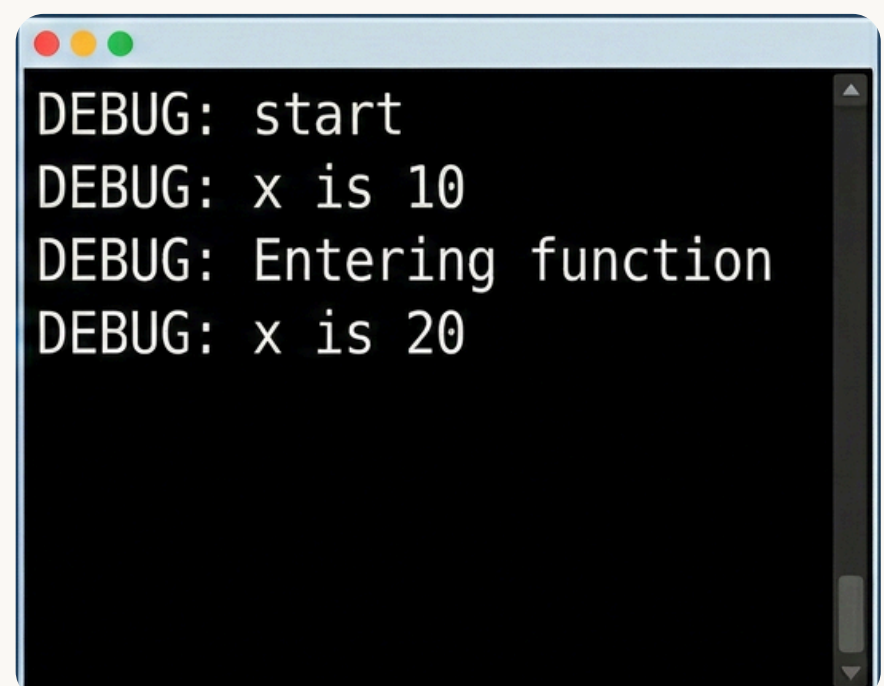


Ala Eddine Hammouda

Printf Debugging

Simple. Reliable. Still relevant

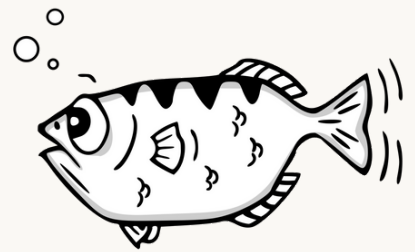
- The most primitive, yet universally understood method.
- **How it Works:** Injects code to send status logs or variables to a terminal via UART or RTT.
- **When to Use:** To check program flow or values without halting the CPU.
- **Where it shines:**
 - ✓ Extremely simple
 - ✓ Works without a debugger
 - ✓ Minimal setup required

A terminal window with a black background and white text. It has a title bar with three colored circles (red, yellow, green) on the left. The text inside the terminal reads: DEBUG: start, DEBUG: x is 10, DEBUG: Entering function, and DEBUG: x is 20. A vertical scrollbar is on the right side.

```
DEBUG: start
DEBUG: x is 10
DEBUG: Entering function
DEBUG: x is 20
```



GNU Debugger



GDB

- **GNU Debugger (GDB)** is a tool for debugging programs in languages like C and C++, allowing developers to inspect and control program execution.
- **What it requires**: a hardware probe like J-Link, ST-Link, or PEmicro to translate high-level software commands into the physical SWD/JTAG signals needed to control a microcontroller.

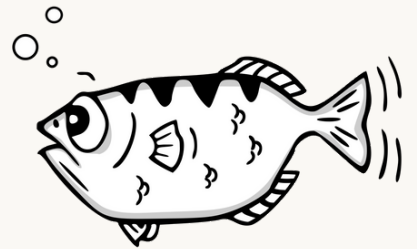


- ✓ Best suited for systems that can be safely halted when you need to understand the code behavior.



Ala Eddine Hammouda

Code Execution Control



GDB

Single-Stepping:

- ↻ **Step Over:** Executes the current line without entering any called functions.
- ↓ **Step Into:** Executes the current line and enters the called function.
- ↑ **Step Out:** Continues execution until the current function returns.

▶ **Pause/Resume:** Pauses or resumes program execution.

↺ **Restart:** Restarts the program from the beginning.

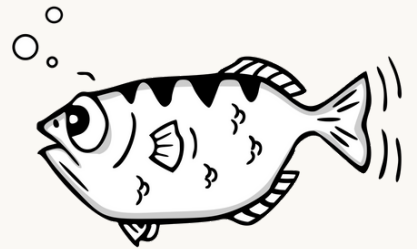
□ **Stop:** Ends the debugging session.

```
12
13
14
15
16
```

Breakpoints: Pause program execution when it reaches a specific line or function.



Inspecting System State



GDB

Variable Inspection: View or modify local and global variables.





```
✓ VARIABLES
  ✓ Local
    > msg: [20]
      counter: 8
      len: 13
    > Global
    > Static
```

Backtrace: Display the call stack to see how execution reached the current point.

```
✓ CALL STACK Paused on step
button_a_callback() board_init.c 336:1
HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
EXTI4_IRQHandler() board_init.c 360:1
```

Register & Memory View: Inspect CPU registers and raw memory addresses.

TERMINALPROBLEMS23RTOSOUTPUTMEMORYDEBUG CONSOLE

&(buf[20]) +     Status: Debugger attached, stopped

00000000200005bc	00	01	02	03	04	05	06	07	08	09	0a	0b	0c	0d	0e	0f
00000000200005e0	41	40	43	c2	45	c4	c7	c6	da	5b	58	59	de	5f	5c	dd
00000000200005f0	53	52	d1	d0	57	d6	55	d4	6c	6d	ee	6f	68	e9	6a	6b
0000000020000600	e5	e4	e7	66	61	60	e3	62	fe	ff	7c	fd	7a	7b	f8	f9
0000000020000610	f7	76	75	74	73	72	71	70	90	91	12	93	94	15	16	97
0000000020000620	99	18	1b	9a	1d	1c	9f	9e	02	03	80	01	86	87	84	85

Live Watch: Monitor variable values in real time.

Watchpoints: Halt execution when a variable changes.
Who keeps changing this variable?!

→ A watchpoint will do the job 😊

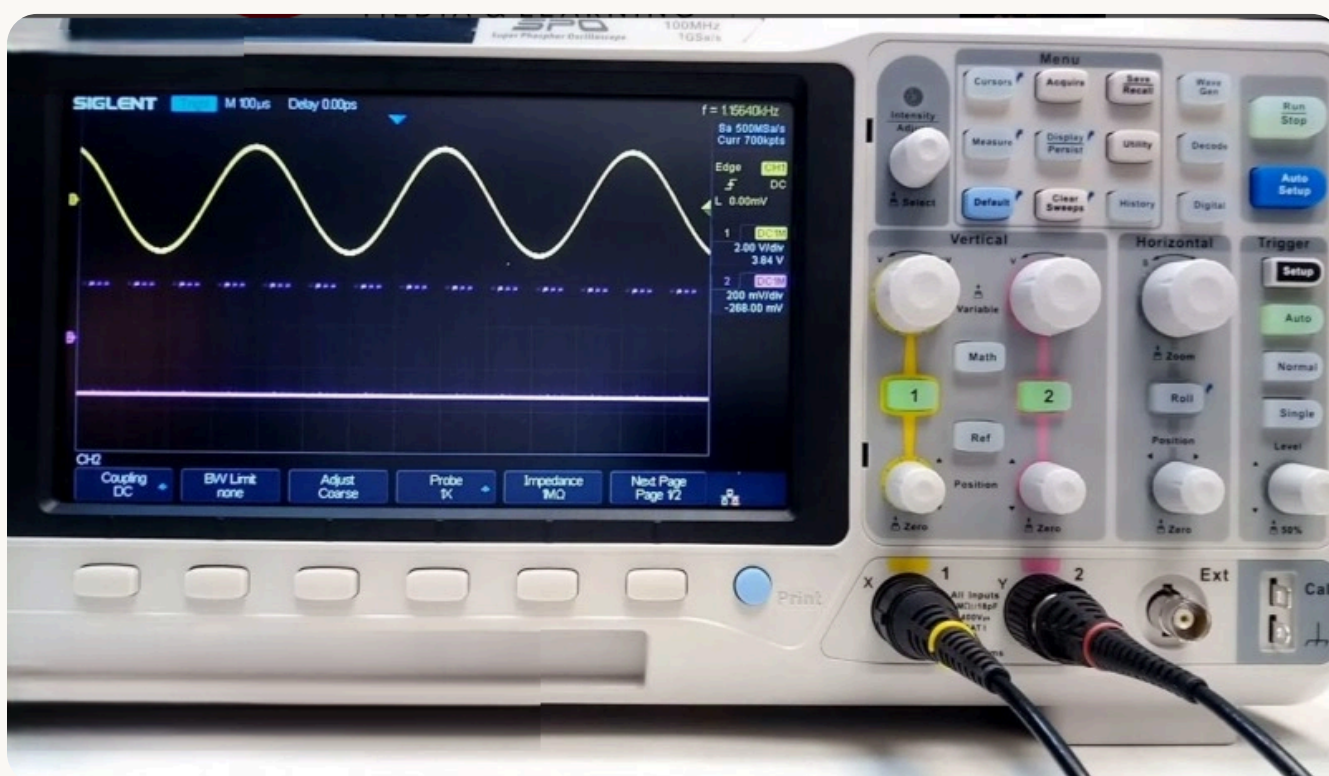


Ala Eddine Hammouda

Oscilloscopes

How it works: You attach oscilloscope probes to pins (GPIOs, buses, analog lines, etc.), and the scope displays voltage over time, letting you see waveforms, measure timing, and capture glitches.

When to use: Whenever you need to observe what the hardware is actually doing in real time, such as verifying a PWM signal's frequency and duty cycle.



Ala Eddine Hammouda

Logic Analyzers

How it works: Connect a logic analyzer to multiple digital lines (GPIOs, buses like I²C/SPI, UART TX/RX, etc.). It records the high/low states over time and can decode serial protocols. You then review the capture on a PC to see the exact sequence of events.



What it requires: A hardware interface like a Saleae Logic that samples voltage transitions and translates them into decoded signals.

When to use: When debugging multi-signal timing or digital protocol issues. For example, if an I²C read fails, a logic analyzer can show whether clock pulses and ACKs occur as expected.



Memory Profiling

How it works: The memory profiler monitors the application at runtime, recording memory allocations, deallocations, and usage patterns over time.

When to use: If the system crashes or behaves unpredictably after running for a while, a memory profiler can show where memory leaks or stack overflows occur.

Choosing the Right Tool:

- Segger SystemView (RTOS & Bare-Metal)
- Valgrind (Embedded Linux)
- Percepio Tracealyzer (RTOS)




→ These tools provide deep visibility into heap/stack usage, allocation lifetimes, and memory fragmentation.



Ala Eddine Hammouda

👉 Follow for practical embedded insights.

👍 Like &  Share if you learned something new!

Know other debugging techniques?
Share them in the comments! 😊



Ala Eddine Hammouda