Module — Debugging Tools & Techniques

1. Foundations: What "debugging" means on bare metal

Debugging is the process of observing and controlling program execution to find the *root cause* of unexpected behavior. On bare metal, there is no operating system (OS) to isolate faults: your code, device drivers, clocks, and linker map interact directly with silicon. This section defines core terms we will use throughout the module.

Key terms (expanded on first use): - **JTAG (Joint Test Action Group):** A multi-wire hardware debug and test interface. - **SWD (Serial Wire Debug):** A two-wire alternative to JTAG for Arm Cortex-M. - **Arm CoreSight:** The Arm on-chip debug/trace architecture that provides breakpoints, watchpoints, the **ITM (Instrumentation Trace Macrocell)** for lightweight printf-style tracing, the **DWT (Data Watchpoint and Trace)** for watchpoints and cycle counting, and the **ETM (Embedded Trace Macrocell)** for instruction tracing. - **ELF (Executable and Linkable Format):** The binary format (with symbols) loaded by your debugger. - **GDB (GNU Debugger):** The debugger front-end used by many IDEs (Integrated Development Environments). - **MCU (Microcontroller Unit):** Here, the NXP i.MX RT1050 with an Arm Cortex-M7 core. - **IDE (Integrated Development Environment):** e.g., NXP MCUXpresso IDE. - **MAP File:** Linker output listing every symbol's address and size—critical for post-mortem analysis.

Generic example: You flash a LED task to the EVKB-IMXRT1050 and nothing blinks. You attach a SWD probe, halt at `main()`, and step to see the GPIO clock not enabled—root cause: missing clock gate.

Avionics use case: After integrating ARINC-429 label processing for **ADC (Air Data Computer)** messages, the flight director annunciation sometimes freezes. With no OS, the only scheduler is interrupts. You attach via SWD, capture a **HardFault (hard fault exception)**, decode the stack to find a buffer overrun in the label ring buffer. The fault only occurs at specific airspeed update bursts—root cause: ISR overruns a shared buffer without proper bounds checking.

2. The NXP EVKB-IMXRT1050 debug stack at a glance

2.1 Hardware

- EVKB-IMXRT1050 board exposes a built-in debug probe (CMSIS-DAP), and also supports external probes (e.g., SEGGER J-Link) via the JTAG/SWD header.
- Cortex-M7 provides CoreSight features: hardware breakpoints, watchpoints, DWT cycle counter, ITM (if routed), and fault status registers.

2.2 Software

- **MCUXpresso SDK** for EVKB-IMXRT1050 (provided): board init (`BOARD_InitHardware()`), drivers like `fsl_gpio.h`, `fsl_lpuart.h`, `fsl_lpuart_edma.h`, delay helpers (`SDK_DelayAtLeastUs()`), and `fsl_debug_console.h` providing `PRINTF()`.
- IDE integrates **GDB** and a symbolized ELF loader. The same ELF is used by command-line tools (`arm-none-eabi-addr2line`, `objdump`) when doing post-mortem decoding.

Generic example: Use an on-board CMSIS-DAP probe to halt at `main()` and verify clocks/peripherals with memory view.

Avionics use case: Using the same probe you set a hardware watchpoint on the ARINC-429 Rx buffer write pointer to catch the exact instruction that overruns the ring buffer during maximum bus load.

---

3. Core techniques you will use every day

3.1 Breakpoints (software & hardware)

What they are: Points where execution halts. *Software breakpoints* replace an instruction in RAM with a “break” instruction. *Hardware breakpoints* use comparators in the CoreSight breakpoint unit and work even in Flash.

How to use on EVKB-IMXRT1050: - Set a hardware breakpoint at a function in Flash when you suspect an initialization path never runs. - Use conditional breakpoints to halt only when a predicate is true (e.g., `rxCount > bufSize`).

Generic example: Halt when `state == ERROR_CAL` to inspect sensor calibration.

Avionics use case: Conditional hardware breakpoint in `Arinc429_ProcessLabel203()` only when the SSM (Sign/Status Matrix) bits indicate “Failure” **and** airspeed label arrives within 5 ms of a mode change interrupt—this isolates a race between label parsing and mode logic.

3.2 Watchpoints (data breakpoints)

What they are: Hardware comparators that halt on a *data access* (read, write, or both) to a given address or range.

How to use: Set a write watchpoint on `arincRxRing[idx].word` to catch out-of-bounds writes. The DWT watchpoint counter is limited—use wisely.

Generic example: Halt on write to `motorDutyCycle` to see who violates a 0–100% clamp.

Avionics use case: Watchpoint on `g_Arinc429TxCtrl.parityMode` catches a stray memory overrun that flips odd/even parity, explaining intermittent parity faults flagged by downstream **LRU (Line Replaceable Unit)**.

3.3 Single-step, run-to-line, and backtrace

Single-step: Execute one instruction or one source line at a time. **Run-to-line:** Resume until the current line executes. **Backtrace:** Examine the call stack with function arguments and return addresses.

Generic example: Step through a CRC (Cyclic Redundancy Check) routine while inspecting variables.

Avionics use case: Backtrace after a fault shows `Arinc429_IsrRx` → `RingPush` → `memcpy` on a mis-sized buffer.

3.4 Inspecting memory-mapped registers

Use the IDE's peripheral viewer or GDB x (examine) commands to view registers for `GPIO`, `LPUART`, `EDMA`, `DMAMUX`, and clock gates in `CCM`.

Generic example: Verify `GPIOx_GDIR` sets LED as output.

Avionics use case: Confirm `LPUART` baud generator matches 100 kbps / 12.5 kbps requirements for an external ARINC-429 UART bridge interface.

---

4. Faults & exceptions on Arm Cortex-M7 (i.MX RT1050)

Cortex-M7 provides precise fault information. The common ones you will decode: - **HardFault:** Escalation of faults or unknown vectors. - **MemManage Fault:** Memory protection/stacking errors. - **BusFault:** Invalid bus accesses (e.g., wrong AHB port). - **UsageFault:** Undefined instruction, divide by zero, etc.

4.1 Capturing a fault with a C handler (post-mortem ready)

Provide a minimal assembly stub that forwards the active stack pointer (MSP or PSP) to a C routine so you can print or store the stacked registers.

```
// startup_fault.c — add to your project (uses CMSIS names)
#include "fsl_debug_console.h"
#include "core_cm7.h"

__attribute__((naked)) void HardFault_Handler(void)
{
    __asm volatile (
        "tst lr, #4\n"
        "ite eq\n"
        "mrseq r0, msp\n"    // r0 = MSP if exception from privileged thread
        "mrsne r0, psp\n"    // r0 = PSP if from unprivileged thread
        "b hardfault_handler_c\n");
}

void hardfault_handler_c(uint32_t *sp)
{
    volatile uint32_t r0  = sp[0];
    volatile uint32_t r1  = sp[1];
    volatile uint32_t r2  = sp[2];
    volatile uint32_t r3  = sp[3];
    volatile uint32_t r12 = sp[4];
    volatile uint32_t lr  = sp[5];
    volatile uint32_t pc  = sp[6];
    volatile uint32_t psr = sp[7];

    PRINTF("\r\nHARDFAULT! r0=%08x r1=%08x r2=%08x r3=%08x r12=%08x lr=%08x p
c=%08x psr=%08x\r\n",
           r0, r1, r2, r3, r12, lr, pc, psr);

    // Optional: store to a reserved OCRAM region for later retrieval
    // crash_store(pc, lr, r0, r1, r2, r3, r12, psr);

    __BKPT(0); // stop under debugger; remove in flight builds
    while (1) { __NOP(); }
}
```

How to map pc back to source: - In the IDE: open the Disassembly/Symbol view at pc. - From command line: `arm-none-eabi-addr2line -e evkbimxrt1050_yourapp.elf 0x<pc>`.

Generic example: A null pointer dereference in `DriverInit()` faults at a fixed address—quickly mapped to a missing `CLOCK_EnableClock()`.

Avionics use case: A burst of ARINC-429 labels triggers a fault inside `RingPush()`. The saved pc maps to `memcpy()` with an incorrect `dataSize` derived from a corrupted length field—found by examining `r0`/`r1`/`r2` (args) alongside the MAP file.

---

5. Timing-deterministic instrumentation (no OS)

5.1 DWT cycle counter for nanosecond-scale timing

Enable the DWT cycle counter and read it before/after critical sections.

```
#include "core_cm7.h"
static inline void dwt_start(void)
{
    CoreDebug->DEMCR |= CoreDebug_DEMCR_TRCENA_Msk;
    DWT->CYCCNT = 0;
    DWT->CTRL |= DWT_CTRL_CYCCNTENA_Msk;
}
static inline uint32_t dwt_cycles(void) { return DWT->CYCCNT; }
```

Usage:

```
dwt_start();
uint32_t t0 = dwt_cycles();
Arinc429_ProcessLabel203();
uint32_t dt = dwt_cycles() - t0;
PRINTF("Label203 processing cycles=%lu\r\n", (unsigned long)dt);
```

Convert cycles to time using the core clock (e.g., 600 MHz).

Generic example: Verify a PID (Proportional-Integral-Derivative) controller loop executes in < 50 µs.

Avionics use case: Guarantee that parsing and queueing of ARINC-429 label 203 (Airspeed/Mach) completes within 80 µs worst-case to avoid missing the next RX interrupt.

5.2 GPIO edge timing (external validation)

Wrap a critical section with GPIO set/clear and measure with a logic analyzer or oscilloscope—independent of the CPU clock's notion of time.

```
GPIO_PinWrite(EXAMPLE_LED_GPIO, EXAMPLE_LED_GPIO_PIN, 1);
critical_function();
GPIO_PinWrite(EXAMPLE_LED_GPIO, EXAMPLE_LED_GPIO_PIN, 0);
```

Generic example: Measure SPI transaction length on a logic analyzer.

Avionics use case: Validate that ARINC-429 label ISR service time never overlaps the next word boundary when the bus runs at 100 kbps under worst-case label density.

---

6. Logging & tracing without breaking real-time behavior

6.1 UART logging using NXP debug console (`PRINTF`)

The SDK's `fsl_debug_console.h` provides `PRINTF()` over the board's debug UART.

```
#include "fsl_debug_console.h"

void log_example(void)
{
    PRINTF("System up. Build:%s %s\r\n", __DATE__, __TIME__);
}
```

Use sparingly in ISRs (Interrupt Service Routines). Prefer buffering logs and flushing in the main loop.

6.2 ITM stimulus ports (if SWO is available in your setup)

If the Single Wire Output (SWO) pin is routed to your probe, the ITM can provide low-overhead printf. If not routed, stay with UART logging.

6.3 Flight builds vs. lab builds

- **Flight build:** Remove semihosting, disable `__BKPT()`, keep fault capture but store to reserved RAM/Flash instead of printing.
- **Lab build:** Enable rich logs and cycle timing prints to accelerate analysis.

Generic example: Ring buffer of 1 KB for logs, drained at idle.

Avionics use case: For ARINC-429 receive path, log only transitions of mode/state and corrupted words (bad parity, invalid SSM), not every word.

---

7. Reading the MAP file and using symbol tools

The MAP file is the truth about where code and objects live. Use it to: - Check stack/heap placement and size. - Identify large objects placed in OCRAM vs. DTCM (Data Tightly Coupled Memory). - Translate raw addresses (from crash logs) into symbols.

Command-line helpers: - `arm-none-eabi-nm -n your.elf | less` — sorted symbols. - `arm-none-eabi-objdump -d your.elf | less` — disassembly. - `arm-none-eabi-addr2line -e your.elf 0x60003478` — address → file:line.

Generic example: Confirm `RxRing` is in non-cacheable section using the `AT_NONCACHEABLE_SECTION` attribute (as in the SDK `lpuart_edma_transfer` example).

Avionics use case: After a field capture, you get only `pc` and `lr` from a post-mortem record. Use `addr2line` + MAP to pinpoint the exact function and line.

---

8. Debugging common drivers on the EVKB-IMXRT1050

8.1 GPIO (light-weight I/O sanity checks)

Reference SDK example: `driver_examples/gpio/led_output`. Typical causes for a non-blinking LED: 1) Pin mux not set to GPIO; 2) GPIO clock gate disabled; 3) LED logic inverted.

Generic example: Toggle the LED with `GPIO_PortToggle()` inside a low-priority loop and verify with a logic analyzer.

Avionics use case: Use a spare GPIO to bracket ARINC-429 ISR entry/exit and prove ISR service time margins.

8.2 LPUART (logging and bridges)

Reference SDK examples: `driver_examples/lpuart/interrupt_transfer` and `lpuart/edma_transfer`.

Generic example: Use the interrupt-driven echo (8-byte blocks) to validate basic connectivity and baud settings.

Avionics use case: Connect EVKB-IMXRT1050 to the ADK-8582 board over UART. Use `LPUART_TransferReceiveNonBlocking()` in the RX path and a small state machine to parse framed ARINC-429 words reported by the adapter. When parity error frames appear, set a watchpoint on the ring buffer write to catch corruption.

8.3 EDMA (enhanced DMA) + DMAMUX (DMA multiplexer)

Reference SDK example: `driver_examples/lpuart/edma_transfer`.

Failure modes: Wrong DMAMUX source, channel not enabled, cache coherency (buffers must be non-cacheable), or missing `EDMA_SetChannelMux()` on devices that require it.

Generic example: EDMA TX stops after first burst—diagnose missing DMAMUX enable.

Avionics use case: ARINC-429 burst logging to SD-card stalls because the RX DMA destination wasn't declared non-cacheable—fix by placing buffers in `AT_NONCACHEABLE_SECTION` and adding cache maintenance if needed.

---

9. Structured root-cause workflow

1) **Reproduce reliably:** Fix the stimulus (input patterns, timing). For ARINC-429, record a known label sequence and play back.
2) **Isolate the fault domain:** ISR vs. main, driver vs. algorithm.
3) **Instrument minimally:** Start with watchpoints and cycle counts; add logs only where necessary.
4) **Capture once, analyze offline:** Save PC/LR/PSR and a few key variables; decode with the MAP file.
5) **Verify the fix under worst-case:** Use GPIO timing or DWT to validate margins.
6) **Regress:** Add a unit test or assertion to prevent recurrence.

---

10. Best practices (avionics-grade)

- **Deterministic builds:** Separate *lab* and *flight* configurations. Flight builds remove semihosting and breakpoints; retain fault recording.

- **Assertions with care:** Use lightweight `ASSERT()` macros that compile out in flight builds but trip immediately in lab builds.
- **Hardware breakpoints economy:** Cortex-M7 has a limited number; prioritize the hottest suspects.
- **Use non-cacheable memory for DMA:** Follow SDK patterns (`AT_NONCACHEABLE_SECTION`).
- **Treat logs as safety artifacts:** Timestamp, include build hash, and store ring buffers for field returns.
- **Concurrency discipline:** ISRs must be bounded, re-entrant safe if needed, and keep shared data protected (disable/enable IRQs around critical updates or use double-buffering).
- **Traceability:** Tie each bug to a CAPA (Corrective And Preventive Action) record; keep ELF/MAP and source snapshot for future investigations.
- **DO-178C (Software Considerations in Airborne Systems and Equipment Certification) mindset:** Tool usage (e.g., code coverage, static analysis) should be justified; debug features that change behavior must be disabled in airborne software. (This is engineering guidance, not certification advice.)

---

11. Hands-on exercises (with solutions)

Important: All exercises use the provided EVKB-IMXRT1050 SDK code patterns. Include `board.h`, `app.h`, and relevant `fsl_*.h` headers exactly like the SDK driver examples. Call `BOARD_InitHardware()` before using peripherals, and use `PRINTF()` from `fsl_debug_console.h` for console output.

Exercise A — Blink + logic probe: verifying hardware init

Goal: Use breakpoints, peripheral view, and GPIO toggling to diagnose a non-blinking LED.

Steps: 1) Copy the `driver_examples/gpio/led_output` into your workspace as `lab_gpio_blink`. 2) Intentionally break the pin mux: comment out the LED pin mux in `pin_mux.c`. 3) Flash and run: LED won't toggle. 4) **Debug:** Set a breakpoint after `BOARD_InitHardware()`. Inspect IOMUXC registers; fix by re-enabling the mux or calling the generated `BOARD_InitPins()`. 5) Use a logic analyzer on the LED pin; verify a clean square wave.

Solution hint: The reference example initializes GPIO and uses `GPIO_PortToggle()` with `SDK_DelayAtLeastUs()` between toggles.

Exercise B — HardFault capture and PC→source mapping

Goal: Implement the HardFault handler stub, trigger a controlled fault, and map the `pc` back to source.

Steps: 1) Add `startup_fault.c` from Section 4.1 to your project and include `fsl_debug_console.h`. 2) In `main()`, intentionally call through a NULL function pointer inside a `#if DEBUG_FAULT` block. 3) Observe the printed registers, especially `pc` and `lr`. 4) Use IDE or `arm-none-eabi-addr2line -e <elf> 0x<pc>` to resolve the faulting line.

Solution snippet: See Section 4.1 listing. Ensure you've enabled the console via `BOARD_InitHardware()`.

Avionics scenario: Simulate an ARINC-429 Rx overflow by forcing `rxCount = bufSize + 1` before `RingPush()`; confirm the HardFault is now replaced by a clean bounds check.

Exercise C — DWT cycle timing of a critical function

Goal: Measure worst-case cycles for `Arinc429_ProcessLabel203()`.

Steps: 1) Add the DWT helpers from Section 5.1. 2) Wrap the processing call with `t0 = dwt_cycles()` and `dt = dwt_cycles() - t0`. 3) Convert to microseconds using your current core clock (e.g., 600 MHz → 1 cycle ≈ 1.667 ns). Print with `PRINTF()`. 4) Provoke worst-case: feed maximum label density from the ADK-8582 over UART.

Solution check: Confirm worst-case time is comfortably below your ISR budget (e.g., 80 µs).

Exercise D — Watchpoint to catch a buffer overrun

Goal: Use a DWT watchpoint to halt exactly where `arincRxRing` overflows.

Steps: 1) In the IDE, set a data write watchpoint on the address of `arincRxRing[bufSize]` (one past the end) or on the `writeIdx` variable itself. 2) Reproduce the failure (burst labels). 3) When halted, inspect the call stack and local variables.

Solution hint: Fix the off-by-one in the ring buffer push and add a size assert.

Exercise E — UART logging and EDMA pitfalls (SDK-based)

Goal: Convert the interrupt-driven LPUART echo to EDMA and fix a deliberate DMAMUX misconfiguration.

Steps: 1) Start from `driver_examples/lpuart/edma_transfer`. 2) Introduce a bug: comment out `DMAMUX_EnableChannel(...)` for RX. 3) Observe: TX runs once, RX never triggers. Use the EDMA and DMAMUX peripheral views to spot the disabled RX channel. 4) Fix the configuration; wrap TX/RX buffers in `AT_NONCACHEABLE_SECTION` macros (as in the example).

Solution excerpt (from SDK pattern):

```
#include "fsl_lpuart_edma.h"
#include "fsl_dmamux.h"
...
DMAMUX_Init(EXAMPLE_LPUART_DMAMUX_BASEADDR);
DMAMUX_SetSource(EXAMPLE_LPUART_DMAMUX_BASEADDR, LPUART_RX_DMA_CHANNEL, LPUAR
T_RX_DMA_REQUEST);
DMAMUX_EnableChannel(EXAMPLE_LPUART_DMAMUX_BASEADDR, LPUART_RX_DMA_CHANNEL);
```

Avionics scenario: Use EDMA to offload high-rate ARINC-429 adapter UART bursts so the CPU can maintain control loop deadlines.

Exercise F — Post-mortem crash record in OCRAM

Goal: Store a compact crash record to a reserved RAM area so that even without a debugger attached you can retrieve root-cause data.

Steps: 1) Define a `.noinit` or dedicated section in your linker script (e.g., `crashlog`) placed in OCRAM. 2) Implement `crash_store()` to write `{magic, pc, lr, psr, r0..r3, r12, timestamp}`. 3) On boot, check for `magic`; if present, print and then clear it.

Solution sketch:

```
__attribute__((section(".crashlog"))) static volatile struct {
    uint32_t magic, pc, lr, psr, r[5];
} g_crash;

static inline void crash_store(uint32_t pc, uint32_t lr,
                               uint32_t r0, uint32_t r1, uint32_t r2, uint32_
t r3, uint32_t r12, uint32_t psr)
{
    g_crash.magic = 0x43524153; // 'CRAS'
    g_crash.pc = pc; g_crash.lr = lr; g_crash.psr = psr;
    g_crash.r[0] = r0; g_crash.r[1] = r1; g_crash.r[2] = r2; g_crash.r[3] = r
3; g_crash.r[4] = r12;
}
```

Avionics scenario: Field return from flight test has only the crash record. You decode pc with `addr2line` and reproduce in the lab.

Exercise G — ARINC-429 receive path debugging (EVKB-IMXRT1050 ↔ ADK-8582 over UART)

Goal: Diagnose intermittent parity errors and missed labels during high-rate reception from the ADK-8582.

Setup: EVKB-IMXRT1050 `LPUARTx` connects to the ADK-8582 UART. The adapter streams framed ARINC-429 words with status (e.g., `0xXX XX XX <parity><SSM>`), plus simple text commands for baud/filters (vendor-specific; treat as opaque for this lab).

Steps: 1) Base project on `driver_examples/lpuart/interrupt_transfer` and refactor RX to push complete frames into a ring buffer (`AT_NONCACHEABLE_SECTION_INIT`). 2) Add a watchpoint on `g_Arinc429RxRing.writeIdx` and a cycle timer around `Arinc429_ProcessWord()`. 3) Drive maximum label density from the ADK-8582 at 100 kbps. 4) Observe occasional parity error frames. Confirm **your** parity is intact by computing parity in software and comparing to the adapter's metadata; if mismatch, suspect buffer corruption. 5) Root cause: a `memcpy()` uses a constant 8 bytes while frames are 10 bytes. The extra 2 bytes overrun the parity mode variable adjacent in memory—precisely what the watchpoint revealed. 6) Fix: correct the length, add bounds checks, and add a unit test that replays a recorded burst.

Validation: With GPIO bracketing and DWT timing, prove ISR service time margin ≥ 30% under worst-case.

---

12. Reference code snippets (all based on SDK conventions)

12.1 Minimal console + LED heartbeat (sanity template)

```c
#include "board.h"
#include "fsl_debug_console.h"
#include "fsl_gpio.h"
#include "app.h"

int main(void)
{
    BOARD_InitHardware();
    PRINTF("EVKB-IMXRT1050 debug template.\r\n");

    while (1)
    {
        SDK_DelayAtLeastUs(100000, SDK_DEVICE_MAXIMUM_CPU_CLOCK_FREQUENCY);
        GPIO_PortToggle(EXAMPLE_LED_GPIO, 1u << EXAMPLE_LED_GPIO_PIN);
    }
}
```

12.2 Interrupt-driven UART echo (excerpt, per SDK)

```
#include "fsl_lpuart.h"
static lpuart_handle_t s_handle;
static uint8_t s_rx[8], s_tx[8];

void LPUART_UserCallback(LPUART_Type *base, lpuart_handle_t *handle, status_t
status, void *ud)
{
    if (status == kStatus_LPUART_TxIdle) { /* ... */ }
    if (status == kStatus_LPUART_RxIdle) { /* ... */ }
}

int uart_echo_init(void)
{
    lpuart_config_t cfg;
    LPUART_GetDefaultConfig(&cfg);
    cfg.baudRate_Bps = BOARD_DEBUG_UART_BAUDRATE;
    cfg.enableTx = true; cfg.enableRx = true;
    LPUART_Init(DEMO_LPUART, &cfg, DEMO_LPUART_CLK_FREQ);
    LPUART_TransferCreateHandle(DEMO_LPUART, &s_handle, LPUART_UserCallback,
NULL);
    return 0;
}
```

12.3 EDMA setup for UART (correct pattern)

```
#include "fsl_lpuart_edma.h"
#include "fsl_dmamux.h"
AT_NONCACHEABLE_SECTION_INIT(uint8_t rxBuf[64]);
AT_NONCACHEABLE_SECTION_INIT(uint8_t txBuf[64]);

DMAMUX_Init(EXAMPLE_LPUART_DMAMUX_BASEADDR);
DMAMUX_SetSource(EXAMPLE_LPUART_DMAMUX_BASEADDR, LPUART_RX_DMA_CHANNEL, LPUAR
T_RX_DMA_REQUEST);
DMAMUX_EnableChannel(EXAMPLE_LPUART_DMAMUX_BASEADDR, LPUART_RX_DMA_CHANNEL);
```

---

13. Troubleshooting checklist

- **Nothing runs:** Check boot mode switches, power rails, and that the correct core image is flashed.
- **Breakpoints ignored in Flash:** Switch to hardware breakpoints; ensure you're not in XIP (Execute-In-Place) with caching anomalies.
- **UART garbage:** Mismatched baud, wrong clock source, or missing pin mux.
- **DMA transfers stall:** DMAMUX source/channel not enabled; buffers need to be non-cacheable; missing `EDMA_SetChannelMux()` on platforms that require it.

- **Random HardFaults:** Stack overflow (check linker script), ISR overruns, or memory corruption from overruns.
- **Timing regressions:** Extra logging in ISR, disabled compiler optimizations, or cache misses due to buffer placement.

---

14. What to keep (artifacts)

- The exact ELF and MAP used for any failure capture.
- Crash records retrieved from OCRAM/Flash.
- The log ring buffer snapshot (with timestamp and build hash).
- Probe configuration (SWD/JTAG, voltage, clock, trace routing notes).

---

15. Summary

You now have a full toolkit: breakpoints, watchpoints, cycle-accurate timing via DWT, GPIO bracketing, disciplined UART/ITM logging, symbol tools, and structured root-cause. You applied these to realistic avionics scenarios around ARINC-429 label handling over a UART-controlled adapter (ADK-8582). Carry these patterns into every lab: begin with minimal instrumentation, capture once, analyze offline with the MAP file, and verify fixes under worst-case timing.
