# STM32 IN DEPTH

## THE COMPLETE EMBEDDED DEVELOPER'S GUIDE

FROM BARE-METAL FUNDAMENTALS
TO PRODUCTION-READY FIRMWARE

- ARM CORTEX-M FUNDAMENTALS
- BARE-METAL PROGRAMMING
- PERIPHERALS MASTERY
- RTOS INTEGRATION AND REAL-TIME SYSTEMS
- SECURITY AND RELIABILITY
- OPTIMIZATION AND PRODUCTION DEPLOYMENT


SCOTT HANSELMANN

# STM32 in Depth: The Complete Embedded Developer's Guide

## From Bare-Metal Fundamentals to Production-Ready Firmware

Steve T. Publications

This book is available at
https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide

This version was published on 2026-07-13


This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

# Contents

**From Bare-Metal Fundamentals to Production-Ready Firmware** . . . . . **1**
About This Book . . . . . 1

**Introduction: The STM32 Universe** . . . . . **2**

**Chapter 1: ARM Cortex-M Architecture and the STM32 Platform** . . . . **4**
The ARM Cortex-M Processor Family . . . . . 4
Instruction Set: Thumb-2 and Pipeline Architecture . . . . . 6
Registers, Stack, and the Programmer's Model . . . . . 6
Nested Vectored Interrupt Controller (NVIC) . . . . . 7
System Control Block (SCB) and SysTick Timer . . . . . 8
STM32 Bus Matrix: AHB, APB, and Memory Mapping . . . . . 10

**Chapter 2: STM32 Families and Device Selection** . . . . . **14**
The Naming Convention Decoded . . . . . 14
Mainstream Series: F0, F1, F3, F4, F7 . . . . . 14
High-Performance Series: H7, H5, U5 . . . . . 14
Ultra-Low-Power Series: L0, L4, L5, U0 . . . . . 14
Specialized Series: G0, G4, N6, WB, WL, C0, C5 . . . . . 14
Selection Matrix: How to Choose Your MCU . . . . . 14

**Chapter 3: Development Environments and Toolchains** . . . . . **16**
STM32CubeIDE: The Official Integrated Environment . . . . . 16
STM32CubeMX: Peripheral Configuration and Code Generation . . . . 16
GCC Cross-Compiler and Makefile-Based Workflows . . . . . 16
Keil MDK-ARM and IAR Embedded Workbench . . . . . 16
OpenOCD, ST-Link Utilities, and Command-Line Tooling . . . . . 16
Project Templates and Build System Comparison . . . . . 16

**Chapter 4: CMSIS, HAL, LL Libraries, and Software Architecture** . . . . **18**
CMSIS Core: The ARM Standard Interface Layer . . . . . 18

Device-Specific CMSIS: Peripheral Register Definitions . . . . . 18
HAL Library: Abstraction with Trade-offs . . . . . 18
LL (Low-Layer) Library: Speed Meets Readability . . . . . 18
Building Your Own Hardware Abstraction Layer . . . . . 18
Software Architecture Patterns for Production Firmware . . . . . 18

**Chapter 5: Bare-Metal Programming, Memory, and Boot Process . . . . 20**
Memory Map: Flash, SRAM, Peripheral Registers . . . . . 20
The Linker Script: Defining Memory Sections . . . . . 20
Startup Code: Vector Table and Reset Handler . . . . . 20
Boot Modes: System Memory, Main Flash, SRAM . . . . . 20
First Bare-Metal Program: Blinking an LED Without Libraries . . . . . 20
Building a Minimal Project from Scratch . . . . . 20

**Chapter 6: Clock Configuration and GPIO . . . . . 22**
Clock Sources: HSI, HSE, CSI, LSI, LSE . . . . . 22
PLL Configuration and System Clock Tree . . . . . 22
Bus Prescalers and Flash Wait States . . . . . 22
GPIO Modes: Input, Output, Alternate Function, Analog . . . . . 22
Pull-Up/Pull-Down Resistors and Drive Strength . . . . . 22
Complete Project: Clock Setup + LED Blink + Button Read . . . . . 22

**Chapter 7: Interrupts, Timers, PWM, and Watchdogs . . . . . 24**
EXTI Lines and Interrupt Priority Grouping . . . . . 24
General-Purpose Timers: Basic Timebase . . . . . 24
Output Compare and PWM Generation . . . . . 24
Input Capture and Encoder Interface Mode . . . . . 24
Advanced Timers: Complementary PWM and Dead-Time . . . . . 24
Independent and Window Watchdog Timers . . . . . 24

**Chapter 8: DMA, ADC, DAC, and Analog Peripherals . . . . . 26**
DMA Architecture: Streams, Channels, and Priorities . . . . . 26
Memory-to-Memory and Peripheral Transfers . . . . . 26
ADC Modes: Single, Scan, Continuous, Injected . . . . . 26
ADC with DMA: Multi-Channel Sampling . . . . . 26
DAC Output and Waveform Generation . . . . . 26
Analog Comparators and Operational Amplifiers . . . . . 26

**Chapter 9: Communication Peripherals I – UART, SPI, and I2C . . . . . 28**
USART/UART: Asynchronous Serial Communication . . . . . 28

Hardware Flow Control and DMA for Serial . . . . . 28
SPI: Master Mode with DMA Transfer . . . . . 28
SPI: Slave Mode and Multi-Master Considerations . . . . . 28
I2C: Master Transmitter and Receiver Modes . . . . . 28
I2C: SMBus, Clock Stretching, and Troubleshooting . . . . . 28

**Chapter 10: Communication Peripherals II – CAN, USB, Ethernet, SDIO, QSPI . . . . . 30**
CAN Bus: Frame Types, Filters, and Bit Timing . . . . . 30
CAN FD and Dual CAN Controllers . . . . . 30
USB Device: CDC, HID, and Mass Storage Classes . . . . . 30
USB OTG Host and Device Modes . . . . . 30
Ethernet MAC with DMA (EMAC) . . . . . 30
SDIO, QSPI, and External Memory Interfaces . . . . . 30

**Chapter 11: Real-Time Operating Systems – FreeRTOS on STM32 . . . . 32**
RTOS Fundamentals: Tasks, Priorities, and Preemption . . . . . 32
Setting Up FreeRTOS with CMSIS-RTOS v2 . . . . . 32
Task Creation, Management, and Lifecycle . . . . . 32
Inter-Task Communication: Queues, Semaphores, Mutexes . . . . . 32
Event Groups and Software Timers . . . . . 32
RTOS Integration with HAL and Hardware Peripherals . . . . . 32

**Chapter 12: Low-Power Modes and Power Optimization . . . . . 34**
Sleep Mode: CPU Idle with Peripheral Activity . . . . . 34
Stop Modes: VCO Shutdown and Wake-Up Sources . . . . . 34
Standby and Shutdown Modes: Deepest Power Savings . . . . . 34
Voltage Scaling and Dynamic Frequency Scaling . . . . . 34
Tickless Idle in FreeRTOS . . . . . 34
Complete Project: Battery-Powered Sensor Node . . . . . 34

**Chapter 13: Bootloaders, Firmware Updates, and Flash Programming . 36**
System Memory Bootloader (ROM Loader) . . . . . 36
In-Application Programming (IAP) Architecture . . . . . 36
Custom Bootloader Design: Vector Table Relocation . . . . . 36
A/B Firmware Slots and Rollback Protection . . . . . 36
Device Firmware Upgrade (DFU) Protocol . . . . . 36
Over-the-Air (OTA) Update Strategies . . . . . 36

**Chapter 14: Debugging, Profiling, and Testing . . . . . 38**

ST-Link Debugger: SWD vs JTAG . . . . . 38
GDB Remote Debugging with OpenOCD . . . . . 38
Hardware Breakpoints, Watchpoints, and Trace . . . . . 38
ITM and Semihosting for Debug Output . . . . . 38
Performance Profiling: Cycle Counters and Cache Analysis . . . . . 38
Unit Testing and Hardware-in-the-Loop Strategies . . . . . 38

**Chapter 15: Security Features and TrustZone . . . . . 40**
Flash Protection: Readout Protection (RDP) Levels . . . . . 40
Memory Protection Unit (MPU) Configuration . . . . . 40
Arm TrustZone for Cortex-M33 Devices . . . . . 40
Hardware Cryptography: AES, HASH, TRNG, RSA/ECC . . . . . 40
Secure Boot and Trusted Execution Environments . . . . . 40
STM32Trust Framework and PSA Certified . . . . . 40

**Chapter 16: Performance Optimization and Production Best Practices . 42**
Compiler Optimizations: -O2, -Os, LTO, and Link-Time . . . . . 42
Cache Management: ICACHE, DCACHE, and Prefetch . . . . . 42
DTCM and CCM SRAM for Time-Critical Code . . . . . 42
Fault Handlers and HardFault Debugging . . . . . 42
Error Handling Patterns and Assert Strategies . . . . . 42
Production Checklist: From Prototype to Volume Manufacturing . . . 42

**Conclusion: The Embedded Developer's Path Forward . . . . . 44**

**References . . . . . 45**

# From Bare-Metal Fundamentals to Production-Ready Firmware

## About This Book

The STM32 family is the world's most popular 32-bit microcontroller platform, with over 15 billion units shipped and more than 4,000 part numbers spanning ultra-low-power sensors to AI-capable edge processors. This book takes you from the fundamentals of the ARM Cortex-M architecture through bare-metal register programming, peripheral mastery, real-time operating system integration, performance optimization, security hardening, and production deployment. Every concept is explained with clear technical prose, supported by fully commented C code examples that progressively increase in complexity. Whether you are a student encountering embedded systems for the first time or a seasoned firmware engineer looking to deepen your STM32 expertise, this book serves as both a learning path and a lasting reference.

# Introduction: The STM32 Universe

The story of modern embedded computing is inseparable from the story of the ARM Cortex-M processor. Since STMicroelectronics first announced the STM32F1 series in June 2007, this family has grown into the most widely deployed 32-bit microcontroller platform on Earth. More than 15 billion units have shipped across a portfolio that now exceeds 4,000 distinct part numbers. These chips control everything from toothbrushes and smartwatches to industrial robots and automotive powertrains. They run inside the Nintendo Game & Watch Super Mario Bros., they drive motor controllers in electric vehicles, and they form the real-time core of hybrid microprocessors that also run Linux.

What makes STM32 dominant is not any single feature. It is the combination of an open architecture built on ARM cores, a comprehensive and freely available software ecosystem, aggressive pricing across every performance tier, and a development experience that scales from hobbyist breadboards to volume production lines. STMicroelectronics has invested heavily in tooling: the free STM32CubeIDE integrated development environment, the STM32CubeMX peripheral configurator, the HAL and LL driver libraries, and an enormous collection of application notes, reference manuals, and example projects. Add to this the vibrant community around Nucleo boards, Blue Pill clones, and open-source toolchains, and you have a platform that lowers the barrier to entry while maintaining professional-grade capabilities.

This book is designed to help you navigate this ecosystem with confidence. We begin at the foundation: understanding what an ARM Cortex-M processor actually does when it executes your C code. You will learn how the 3-stage pipeline of a Cortex-M3 differs from the 6-stage superscalar pipeline of a Cortex-M7, how the Nested Vectored Interrupt Controller arbitrates between competing interrupt sources, and how memory-mapped I/O turns register writes into real electrical signals on physical pins.

From there, we move into practical territory. You will learn to select the right STM32 device from the sprawling portfolio, set up your development environment whether you prefer STM32CubeIDE, a GCC toolchain with Makefiles, or commercial IDEs like Keil MDK-ARM and IAR Embedded Workbench. We dissect the layered software architecture: CMSIS at the core, HAL and

LL libraries in the middle, and your own application code on top. You will write bare-metal programs that speak directly to hardware registers before graduating to library-assisted development.

The heart of the book covers peripherals in depth. Every GPIO mode, every timer configuration, every DMA transfer pattern, every communication protocol from UART to CAN FD is explained with working code and design rationale. We do not just show you how to call HAL_UART_Transmit(); we explain baud rate generation, oversampling strategies, hardware flow control, and the trade-offs between polling, interrupt-driven, and DMA-based approaches.

Advanced topics round out the coverage: FreeRTOS integration for multitasking firmware, low-power mode configuration for battery-operated designs, bootloader architecture for field firmware updates, hardware security features including TrustZone isolation, and production considerations like fault handling, performance profiling, and manufacturing test strategies.

Throughout the book, code examples use the STM32F4 series as the primary reference platform. The STM32F407 is chosen because it represents a well-rounded mid-range device: Cortex-M4F core with hardware floating-point, 1 MB flash, 256 KB SRAM, and a rich peripheral set. Where behavior differs significantly on other families, callouts note the variations. All code follows consistent conventions: register-level examples show the raw hardware interface, while HAL and LL examples demonstrate library-based approaches side by side.

By the time you finish this book, you will have a deep mental model of how STM32 microcontrollers work from silicon to software. You will be able to read a reference manual and translate it into working code. You will understand the trade-offs between abstraction levels and know when to reach for a HAL function versus when to write a register access directly. Most importantly, you will have the confidence to tackle any STM32-based project, from a simple LED blinker to a production-grade industrial controller with OTA firmware updates and hardware security enforcement.

The journey starts now.

# Chapter 1: ARM Cortex-M Architecture and the STM32 Platform

Before writing a single line of firmware, you need to understand the hardware that will execute it. The STM32 microcontroller is not just a blob of silicon that runs C code. It is a carefully engineered system built around an ARM Cortex-M processor core, surrounded by memory blocks, peripheral controllers, debug infrastructure, and a sophisticated bus matrix that routes data between all these components. This chapter establishes the architectural foundation for everything that follows.

## The ARM Cortex-M Processor Family

ARM (originally Advanced RISC Machines) does not manufacture chips. Instead, it designs processor cores and licenses them to semiconductor companies like STMicroelectronics, NXP, Texas Instruments, and dozens of others. Each licensee integrates the ARM core with their own peripherals, memory controllers, and analog blocks to create a complete microcontroller. The STM32 family is ST's implementation of this model.

The Cortex-M family targets microcontroller applications where deterministic behavior, low power consumption, and fast interrupt response matter more than raw throughput. This contrasts with the Cortex-A series (used in smartphones and tablets) which prioritizes performance at the cost of interrupt latency and power efficiency.

The STM32 portfolio uses several different Cortex-M cores:

**Cortex-M0 / M0+**: The entry-level cores, optimized for small die area and low cost. They implement the ARMv6-M architecture with a 3-stage pipeline (M0) or 2-stage pipeline (M0+). These support a subset of the Thumb-2 instruction set: all 16-bit Thumb-1 instructions plus a handful of 32-bit Thumb-2 instructions (BL, DMB, DSB, ISB, MRS, MSR). They lack hardware division and DSP instructions. STM32 families using these cores include the F0 series (up

to 48 MHz), G0 series (up to 64 MHz), L0 series (up to 32 MHz), C0 series (up to 48 MHz), and U0 series (up to 56 MHz).

**Cortex-M3**: The original mainstream core that launched the STM32 line. It implements ARMv7-M with a 3-stage pipeline, full Thumb-2 instruction set, hardware integer division (2 to 12 cycles depending on operands), and 12-cycle interrupt latency. It supports up to 240 external interrupts via the NVIC. The STM32F1 series uses this core at up to 72 MHz.

**Cortex-M4 / M4F**: A superset of the Cortex-M3 that adds DSP instructions (SIMD, saturating arithmetic, single-cycle MAC operations) and an optional hardware floating-point unit (FPU). When the FPU is included, the core is designated "M4F." The FPU implements the VFPv4-SP extension for single-precision IEEE-754 floating point. STM32 families using M4/M4F include the F3 series (up to 72 MHz, analog-focused), F4 series (up to 180 MHz), L4/L4+ series (up to 120 MHz, low-power), G4 series (up to 170 MHz, motor control), and WB/WB0 wireless series.

**Cortex-M7**: The high-performance core with a 6-stage superscalar pipeline capable of dual-issue load/store operations. It supports up to 64 KB instruction cache and 64 KB data cache, optional double-precision FPU (VFPv5), Tightly-Coupled Memory (TCM) blocks, and an Embedded Trace Macrocell (ETM) for cycle-accurate debugging. The STM32F7 series runs at up to 216 MHz, while the STM32H7 series reaches 400 to 600 MHz on a 40 nm process.

**Cortex-M33**: The security-focused core implementing ARMv8-M Mainline with TrustZone support. It partitions memory and peripherals into Secure and Non-Secure worlds at the hardware level, enabling trusted execution environments. Optional FPU, up to 16 MPU regions, and up to 8 Security Attribution Unit (SAU) regions. Used in STM32L5 (up to 110 MHz), H5 (up to 250 MHz), U5 (up to 160 MHz), C5 (up to 144 MHz), U3, and WBA series.

**Cortex-M55**: The AI-capable core implementing ARMv8.1-M Mainline with the Helium M-Profile Vector Extension (MVE). Helium enables single-instruction multiple-data (SIMD) processing on 128-bit vector registers, dramatically accelerating machine learning inference and signal processing workloads. Used in the STM32N6 series at up to 800 MHz.

The architectures are binary upward compatible: code compiled for Cortex-M0 runs unmodified on Cortex-M3, M4, and M7. Code compiled for Cortex-M3 runs on M4 and M7. This compatibility is a key advantage of the STM32 ecosystem: you can prototype on an inexpensive F0 device and migrate

to an F4 or H7 with minimal code changes.

## Instruction Set: Thumb-2 and Pipeline Architecture

Cortex-M processors execute only Thumb-2 instructions, a hybrid instruction set that mixes 16-bit and 32-bit encodings. The 16-bit Thumb-1 instructions provide high code density for common operations (register moves, simple arithmetic, short branches), while the 32-bit Thumb-2 instructions handle more complex operations (long-range branches, memory accesses with large offsets, multiply-accumulate) without sacrificing too much density.

This design contrasts with legacy ARM cores that could execute both 32-bit ARM and 16-bit Thumb instructions. Cortex-M processors have no ARM state: they always run in Thumb state, which simplifies the hardware design and reduces silicon area.

The pipeline architecture determines how many instruction stages exist between fetch and completion:

- **Cortex-M0+**: 2-stage pipeline (fetch, execute). Simplest pipeline, lowest power.
- **Cortex-M0 / M3 / M4 / M33**: 3-stage pipeline (fetch, decode, execute). Adds branch speculation in M3/M4.
- **Cortex-M55**: 4 to 5-stage pipeline with Helium vector processing.
- **Cortex-M7**: 6-stage superscalar pipeline with dual-issue capability for load/load and load/store pairs. Branch target address cache (BTAC) reduces branch penalty.
- **Cortex-M85**: 7-stage pipeline, the deepest in the Cortex-M family.

Deeper pipelines can sustain higher clock frequencies but suffer more from branch misprediction penalties. The Cortex-M7's superscalar design allows it to achieve up to 2.5 DMIPS/MHz (Dhrystone millions of instructions per second per megahertz), compared to approximately 1.25 DMIPS/MHz for the Cortex-M4 and roughly 1.0 DMIPS/MHz for the Cortex-M3.

## Registers, Stack, and the Programmer's Model

The Cortex-M programmer's model presents 32 general-purpose registers (R0 through R15), though several have special roles:

- **R0 to R12**: General-purpose registers available for all code.
- **R13 (SP)**: The stack pointer. Cortex-M maintains two separate stack pointers: the Main Stack Pointer (MSP) used by exception handlers and startup code, and the Process Stack Pointer (PSP) optionally used by thread-mode tasks in an RTOS context.
- **R14 (LR)**: The link register, holding return addresses for subroutine calls. A special "EXC_RETURN" value in LR tells the hardware to restore exception context upon function return.
- **R15 (PC)**: The program counter. Reading it returns the address of the next instruction (plus 4 on Cortex-M3/M4/M7 due to the pipeline).

The processor operates in two modes:

- **Thread mode**: Normal application execution. Can run at privileged or unprivileged level.
- **Handler mode**: Active exception/interrupt handling. Always runs at privileged level.

Thread mode is where your main() function and RTOS tasks execute. Handler mode is entered automatically when an interrupt fires, with the hardware pushing a stacked context (R0-R3, R12, LR, PC, xPSR) onto the stack before calling the interrupt handler. This automatic stacking takes just 12 cycles on Cortex-M3/M4 and saves significant software overhead compared to legacy processors.

The program status register (xPSR) holds condition flags (N, Z, C, V), exception number, and execution state information. On M4F cores with FPU, additional floating-point status registers (FPSCR) and 32 double-precision floating-point registers (S0-S31 / D0-D15) are available.

## Nested Vectored Interrupt Controller (NVIC)

The NVIC is perhaps the most distinctive feature of the Cortex-M architecture. Unlike traditional interrupt controllers that use a fixed priority scheme, the NVIC supports fully nestable interrupts with configurable priorities.

Key NVIC features:

- **Priority grouping**: The 8-bit priority field is split into preemption priority bits and subpriority bits. Higher preemption priority interrupts can preempt lower-priority ones that are currently executing. Subpriority determines execution order when interrupts of equal preemption priority arrive simultaneously.
- **Tail-chaining**: If an interrupt finishes and another pending interrupt with equal or higher priority exists, the hardware skips the full unstacking/restacking sequence, saving 6 cycles.
- **Late arrival**: If a higher-priority interrupt arrives while the hardware is stacking registers for a lower-priority interrupt, the already-fetched registers are written to the higher-priority handler's stack frame, avoiding wasted work.
- **Interrupt disable**: The PRIMASK register can disable all interrupts except NMI and HardFault. FAULTMASK disables everything including NMI (but not HardFault). BASEPRI allows selective disabling of interrupts below a certain priority threshold, which is essential for RTOS context switching where the scheduler must be atomic but high-priority hardware interrupts must still fire.

The NVIC supports 1 to 240 external interrupts on Cortex-M3/M4/M7 and up to 480 on Cortex-M33/M55/M85. STM32 devices typically implement 60 to 100+ interrupt lines, each mapped to a specific peripheral event (UART receive complete, timer overflow, DMA transfer finish, etc.).

```
/* Example: Configuring NVIC priority grouping and setting
 * an interrupt priority using CMSIS */
#include "stm32f4xx.h"

void configure_interrupt_priorities(void)
{
    /* Set priority grouping: 2 bits preemption, 2 bits subpriority
     * This gives 4 preemption levels (0-3) and 4 sublevels (0-3) */
    NVIC_SetPriorityGrouping(2);

    /* Set USART2 interrupt to preemption priority 1, subpriority 0
     * With grouping=2, the top 2 bits of the 4-bit priority field
     * are preemption, bottom 2 bits are subpriority.
     * Priority value = (1 << 2) | 0 = 0x04 */
    NVIC_SetPriority(USART2_IRQn, 4);

    /* Enable the USART2 interrupt in the NVIC */
    NVIC_EnableIRQ(USART2_IRQn);
}
```

## System Control Block (SCB) and SysTick Timer

The System Control Block provides access to core-level functionality:

- **VTOR (Vector Table Offset Register)**: Allows relocating the interrupt vector table from its default location at address 0x00000000 to any aligned address in memory. Essential for bootloader implementations where the application firmware resides at a flash offset.
- **AIRCR (Application Interrupt and Reset Control Register)**: Controls system reset, bus fault behavior, and endianness (though endianness is typically fixed in silicon).
- **SHPR registers (System Handler Priority Registers)**: Set priorities for system exceptions like MemManage, BusFault, UsageFault, and SysTick.
- **SCR (System Control Register)**: Controls SLEEPDEEP and SLEEPONEXIT bits that determine which low-power mode the processor enters on WFI/WFE instructions.
- **CPACR (Coprocessor Access Type Register)**: On M4F cores, enables access to the FPU by setting coprocessor 10 and 11 full access.

The SysTick timer is a 24-bit downcounter built into every Cortex-M core (except some minimal M0 implementations). It generates an interrupt when the counter reaches zero, then reloads from a programmable value. This makes it ideal for:

- Providing a periodic timebase for HAL_Delay() and similar functions
- Serving as the RTOS tick source for FreeRTOS task scheduling
- Implementing simple timeouts in bare-metal code

```c
/* Example: Configuring SysTick for a 1 ms tick at 168 MHz SYSCLK */
#include "stm32f4xx.h"

void systick_1ms_init(void)
{
    /* SysTick runs from the processor clock (168 MHz on STM32F407)
     * Reload value = (168,000,000 / 1,000) - 1 = 167,999 */
    SysTick->LOAD  = 167999;

    /* Clear current value to force immediate reload */
    SysTick->VAL   = 0;

    /* Enable SysTick interrupt in NVIC (priority 255 = lowest)
     * Use processor clock (not external reference)
     * Enable the counter */
    SysTick->CTRL  = SysTick_CTRL_CLKSOURCE_Msk |
                     SysTick_CTRL_TICKINT_Msk    |
                     SysTick_CTRL_ENABLE_Msk;
}

volatile uint32_t sys_tick_counter = 0;

void SysTick_Handler(void)
{
    sys_tick_counter++;
}
```

## STM32 Bus Matrix: AHB, APB, and Memory Mapping

STM32 microcontrollers use an Advanced High-performance Bus (AHB) as the system backbone, with Advanced Peripheral Buses (APB) for lower-speed peripherals. The exact bus topology varies by family, but the general pattern is consistent.

On the STM32F407, the memory map looks like this:

| Address Range | Description |
|---|---|
| 0x0000 0000 to 0x1FFF FFFF | Alias region (mirrors boot source) |
| 0x0800 0000 to 0x0FFF FFFF | Internal Flash memory (1 MB on F407) |
| 0x1000 0000 to 0x107F FFFF | CCM SRAM (Core-Coupled Memory, 64 KB on F407) |
| 0x1FFF 0000 to 0x1FFF 7FFF | System memory (bootloader ROM) |
| 0x2000 0000 to 0x203F FFFF | Main SRAM (192 KB) + TCM SRAM (64 KB) on F407 |
| 0x4000 0000 to 0x5FFF FFFF | Peripheral register space |
| 0x6000 0000 to 0x9FFF FFFF | External memory space (FSMC/FMC) |
| 0xA000 0000 to 0xBFFF FFFF | Reserved / external SRAM |
| 0xC000 0000 to 0xDFFF FFFF | SDRAM space (FMC on F429/439) |

The peripheral register space at 0x4000 0000 is organized by bus:

- **APB1** (up to 42 MHz on F4): Low-speed peripherals including USART2/3, I2C1/2, SPI2/3, TIM2-7, CAN1/2, USB OTG FS, RTC
- **APB2** (up to 84 MHz on F4): Higher-speed peripherals including USART1/6, SPI1/4, TIM1/8-11, ADC1-3, SDIO
- **AHB1** (up to 168 MHz on F4): High-speed peripherals including GPIOA-G, DMA1/2, USB OTG HS, RNG, CRC
- **AHB2** (up to 168 MHz on F4): Dedicated to DCMI and OCTOSPI on some devices
- **AHB3** (up to 168 MHz on F4): FMC memory controller

Each peripheral's registers are memory-mapped: reading or writing a specific address performs a register access. For example, writing to the GPIOA output data register at address 0x4002 0014 changes the electrical state of physical pins PA0 through PA15.

```c
/* Example: Direct register access to toggle LED on PA5
 * This is what "bare-metal" programming looks like */
#include "stm32f4xx.h"

/* Enable GPIOA clock by setting bit 0 in RCC AHB1 peripheral
 * clock enable register */
void led_baremetal_init(void)
{
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOAEN;  /* Enable GPIOA clock */

    /* Configure PA5 as general-purpose output
     * MODER register: bits [11:10] for pin 5
     * Value 0b01 = General purpose output mode */
    GPIOA->MODER &= ~GPIO_MODER_MODE5;     /* Clear existing config */
    GPIOA->MODER |= GPIO_MODER_MODE5_0;    /* Set to output mode */

    /* OTYPER: push-pull (default, bit 5 = 0) */
    /* OSPEEDR: 2 MHz medium speed (bits [11:10] = 0b01) */
    GPIOA->OSPEEDR &= ~GPIO_OSPEEDER_OSPEEDR5;
    GPIOA->OSPEEDR |= GPIO_OSPEEDER_OSPEEDR5_0;

    /* PUPDR: no pull-up/pull-down (bits [11:10] = 0b00) */
    GPIOA->PUPDR &= ~GPIO_PUPDR_PUPDR5;
}

void led_toggle(void)
{
    /* Toggle bit 5 in the output data register
     * ODR holds the current output state of all pins */
    GPIOA->ODR ^= GPIO_ODR_OD5;
}

int main(void)
{
    led_baremetal_init();

    while (1)
    {
        led_toggle();
        /* Simple delay: busy-wait approximately 500 ms at 168 MHz */
        for (volatile uint32_t i = 0; i < 42000000; i++);
    }
}
```

This bare-metal approach gives you complete control but requires careful attention to the reference manual. Every register field, every bit position, every clock enable sequence must be correct. In subsequent chapters we will see how CMSIS header files and HAL/LL libraries abstract away much of this complexity while still letting you understand what happens underneath.

The bus matrix architecture has important implications for performance and design. DMA controllers connect to AHB buses as bus masters, meaning they can transfer data without CPU involvement. The CPU core connects via its own AHB master port. When both try to access the same memory region simultaneously, the bus matrix arbitrates based on configured priorities. Understanding this arbitration is crucial when designing high-throughput data paths involving ADC sampling, DMA transfers, and real-time control loops running concurrently.

The CCM (Core-Coupled Memory) SRAM deserves special mention. It is accessible only by the CPU core, not by DMA or other peripherals. This makes it ideal for storing time-critical stack frames, RTOS control structures, or data that must not be modified by DMA. On the STM32F407, 64 KB of CCM SRAM sits at address 0x1000 0000, entirely separate from the main SRAM at 0x2000 0000.

With this architectural foundation in place, we can now explore the STM32 device portfolio and learn how to choose the right chip for a given application.

# Chapter 2: STM32 Families and Device Selection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## The Naming Convention Decoded

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Mainstream Series: F0, F1, F3, F4, F7

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## High-Performance Series: H7, H5, U5

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Ultra-Low-Power Series: L0, L4, L5, U0

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Specialized Series: G0, G4, N6, WB, WL, C0, C5

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Selection Matrix: How to Choose Your MCU

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# Chapter 3: Development Environments and Toolchains

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## STM32CubeIDE: The Official Integrated Environment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## STM32CubeMX: Peripheral Configuration and Code Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## GCC Cross-Compiler and Makefile-Based Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Keil MDK-ARM and IAR Embedded Workbench

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## OpenOCD, ST-Link Utilities, and Command-Line Tooling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Project Templates and Build System Comparison

# Chapter 4: CMSIS, HAL, LL Libraries, and Software Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## CMSIS Core: The ARM Standard Interface Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Device-Specific CMSIS: Peripheral Register Definitions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## HAL Library: Abstraction with Trade-offs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## LL (Low-Layer) Library: Speed Meets Readability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Building Your Own Hardware Abstraction Layer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Software Architecture Patterns for Production Firmware

# Chapter 5: Bare-Metal Programming, Memory, and Boot Process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Memory Map: Flash, SRAM, Peripheral Registers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## The Linker Script: Defining Memory Sections

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Startup Code: Vector Table and Reset Handler

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Boot Modes: System Memory, Main Flash, SRAM

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## First Bare-Metal Program: Blinking an LED Without Libraries

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Building a Minimal Project from Scratch

# Chapter 6: Clock Configuration and GPIO

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Clock Sources: HSI, HSE, CSI, LSI, LSE

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## PLL Configuration and System Clock Tree

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Bus Prescalers and Flash Wait States

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## GPIO Modes: Input, Output, Alternate Function, Analog

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Pull-Up/Pull-Down Resistors and Drive Strength

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Complete Project: Clock Setup + LED Blink + Button Read

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# Chapter 7: Interrupts, Timers, PWM, and Watchdogs

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## EXTI Lines and Interrupt Priority Grouping

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## General-Purpose Timers: Basic Timebase

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Output Compare and PWM Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Input Capture and Encoder Interface Mode

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Advanced Timers: Complementary PWM and Dead-Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Independent and Window Watchdog Timers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# Chapter 8: DMA, ADC, DAC, and Analog Peripherals

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## DMA Architecture: Streams, Channels, and Priorities

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Memory-to-Memory and Peripheral Transfers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## ADC Modes: Single, Scan, Continuous, Injected

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## ADC with DMA: Multi-Channel Sampling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## DAC Output and Waveform Generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Analog Comparators and Operational Amplifiers

# Chapter 9: Communication Peripherals I – UART, SPI, and I2C

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## USART/UART: Asynchronous Serial Communication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Hardware Flow Control and DMA for Serial

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## SPI: Master Mode with DMA Transfer

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## SPI: Slave Mode and Multi-Master Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## I2C: Master Transmitter and Receiver Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## I2C: SMBus, Clock Stretching, and Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# Chapter 10: Communication Peripherals II – CAN, USB, Ethernet, SDIO, QSPI

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## CAN Bus: Frame Types, Filters, and Bit Timing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## CAN FD and Dual CAN Controllers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## USB Device: CDC, HID, and Mass Storage Classes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## USB OTG Host and Device Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Ethernet MAC with DMA (EMAC)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## SDIO, QSPI, and External Memory Interfaces

# Chapter 11: Real-Time Operating Systems – FreeRTOS on STM32

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## RTOS Fundamentals: Tasks, Priorities, and Preemption

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Setting Up FreeRTOS with CMSIS-RTOS v2

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Task Creation, Management, and Lifecycle

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Inter-Task Communication: Queues, Semaphores, Mutexes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Event Groups and Software Timers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## RTOS Integration with HAL and Hardware Peripherals

# Chapter 12: Low-Power Modes and Power Optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Sleep Mode: CPU Idle with Peripheral Activity

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Stop Modes: VCO Shutdown and Wake-Up Sources

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Standby and Shutdown Modes: Deepest Power Savings

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Voltage Scaling and Dynamic Frequency Scaling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Tickless Idle in FreeRTOS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Complete Project: Battery-Powered Sensor Node

# Chapter 13: Bootloaders, Firmware Updates, and Flash Programming

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## System Memory Bootloader (ROM Loader)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## In-Application Programming (IAP) Architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Custom Bootloader Design: Vector Table Relocation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## A/B Firmware Slots and Rollback Protection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Device Firmware Upgrade (DFU) Protocol

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Over-the-Air (OTA) Update Strategies

# Chapter 14: Debugging, Profiling, and Testing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## ST-Link Debugger: SWD vs JTAG

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## GDB Remote Debugging with OpenOCD

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Hardware Breakpoints, Watchpoints, and Trace

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## ITM and Semihosting for Debug Output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Performance Profiling: Cycle Counters and Cache Analysis

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Unit Testing and Hardware-in-the-Loop Strategies

# Chapter 15: Security Features and TrustZone

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Flash Protection: Readout Protection (RDP) Levels

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Memory Protection Unit (MPU) Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Arm TrustZone for Cortex-M33 Devices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Hardware Cryptography: AES, HASH, TRNG, RSA/ECC

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Secure Boot and Trusted Execution Environments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## STM32Trust Framework and PSA Certified

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# Chapter 16: Performance Optimization and Production Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Compiler Optimizations: -O2, -Os, LTO, and Link-Time

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Cache Management: ICACHE, DCACHE, and Prefetch

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## DTCM and CCM SRAM for Time-Critical Code

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Fault Handlers and HardFault Debugging

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Error Handling Patterns and Assert Strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

## Production Checklist: From Prototype to Volume Manufacturing

# Conclusion: The Embedded Developer's Path Forward

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.

# References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/stm32indepththecompleteembeddeddevelopersguide.