Chapter 8
The Right Start

The startup code is called by the microcontroller hardware after reset and is the first code to execute before calling the main() subroutine. The startup code predominantly consists of initialization code and may include, among other things, CPU-initialization, zero-clear RAM initialization, ROM-to-RAM static initialization and static ctor call initialization. The compiler’s default startup code is often tightly bound to the compiler’s runtime libraries and may not be available as source code. In addition, even if the source of the startup code is available, it can be hard to understand because it may be written in assembly and cluttered with a multitude of options required for supporting a variety of chip derivatives. This chapter describes how to implement a custom startup code and its initializations written predominantly in C++, from reset to main().

8.1 The Startup Code

It can be preferable to write a custom version of the startup code. This makes it possible to include specialized initialization mechanisms for I/O pins, oscillators, watchdog timers, etc. These might otherwise be postponed to an unduly late time, such as in the main() subroutine. The flowchart of a custom startup code is shown in Fig. 8.1.
We will now examine the main parts of this startup code going step-by-step through a real example. The code below shows the implementation of the startup code for the 32-bit target in the reference project of the companion code.

```c
extern "C" void startup()
{
    // Set the stack pointers.
    __asm volatile("movs r1, #0");

    // Initialize I/O pins, oscillators and watchdog.
    mcal::cpu::init();

    // Initialize statics from ROM to RAM.
    // Zero-clear non-initialized static RAM.
    crt::initRam();
    mcal::wdg::trigger();

    // Call all ctor initializations.
    crt::initCtors();
```