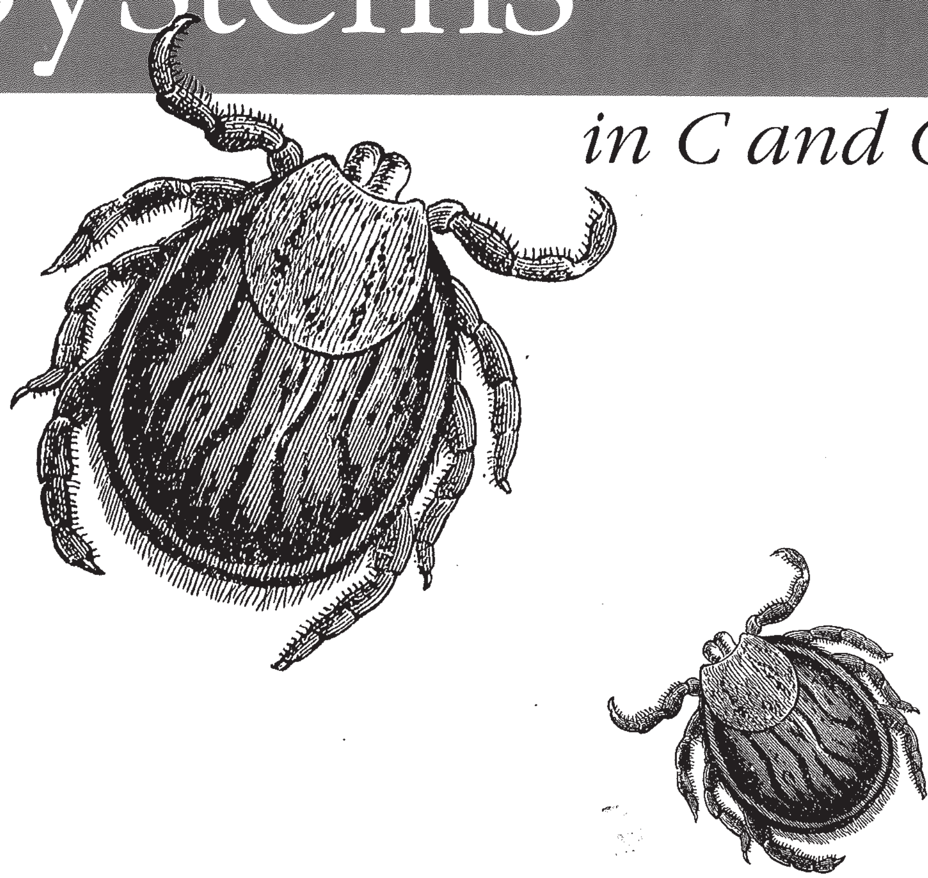
Thinking Inside the Box

Programming Embedded Systems

in C and C++



O'REILLY®

Michael Barr

Programming Embedded Systems in C and C++

Michael Barr

O'REILLY®

Beijing · Cambridge · Farnham · Köln · Paris · Sebastopol · Taipei · Tokyo

Programming Embedded Systems in C and C++
by Michael Barr

Printed in the United States of America.

Published by O'Reilly & Associates, Inc., 101 Morris Street, Sebastopol, CA 95472.

Editor: Andy Oram

Production Editor: Melanie Wang

Printing History:

January 1999: First Edition.

ISBN: 1-56592-354-5 [6/00]
[M]

In this chapter:

- *Types of Memory*
- *Memory Testing*
- *Validating Memory Contents*
- *Working with Flash Memory*

6

Memory

> ***Tyrell:** If we give them a past, we create a cushion for their emotions and, consequently, we can control them better.*
>
> ***Deckard:** Memories. You're talking about memories.*
>
> — the movie *Blade Runner*

In this chapter, you will learn everything you need to know about memory in embedded systems. In particular, you will learn about the types of memory you are likely to encounter, how to test memory devices to see if they are working properly, and how to use Flash memory.

Types of Memory

Many types of memory devices are available for use in modern computer systems. As an embedded software engineer, you must be aware of the differences between them and understand how to use each type effectively. In our discussion, we will approach these devices from a software viewpoint. As you are reading, try to keep in mind that the development of these devices took several decades and that there are significant physical differences in the underlying hardware. The names of the memory types frequently reflect the historical nature of the development process and are often more confusing than insightful.

Most software developers think of memory as being either random-access (RAM) or read-only (ROM). But, in fact, there are subtypes of each and even a third class of hybrid memories. In a RAM device, the data stored at each memory location can be read or written, as desired. In a ROM device, the data stored at each memory location can be read at will, but never written. In some cases, it is possible to overwrite the data in a ROM-like device. Such devices are called hybrid memories because they exhibit some of the characteristics of both RAM and ROM. Figure 6-1 provides a classification system for the memory devices that are commonly found in embedded systems.

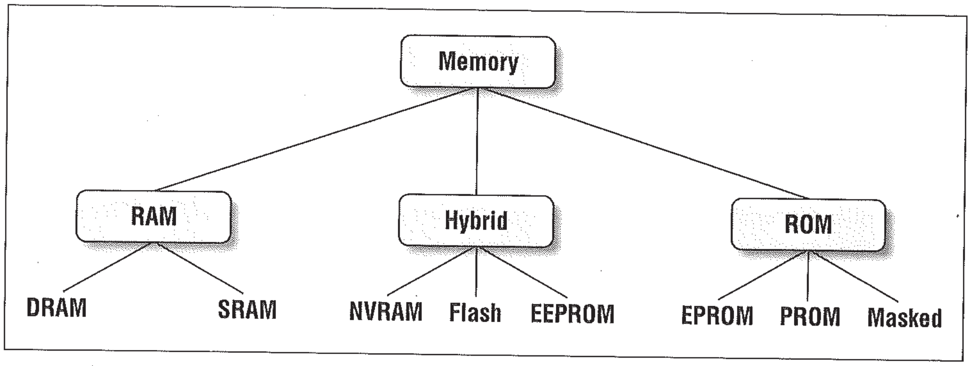


Figure 6-1. Common memory types in embedded systems

Types of RAM

There are two important memory devices in the RAM family: SRAM and DRAM. The main difference between them is the lifetime of the data stored. SRAM (static RAM) retains its contents as long as electrical power is applied to the chip. However, if the power is turned off or lost temporarily then its contents will be lost forever. DRAM (dynamic RAM), on the other hand, has an extremely short data lifetime—usually less than a quarter of a second. This is true even when power is applied constantly.

In short, SRAM has all the properties of the memory you think of when you hear the word RAM. Compared to that, DRAM sounds kind of useless. What good is a memory device that retains its contents for only a fraction of a second? By itself, such a volatile memory is indeed worthless. However, a simple piece of hardware called a DRAM controller can be used to make DRAM behave more like SRAM. (See the sidebar "DRAM Controllers" later in this chapter.) The job of the DRAM controller is to periodically refresh the data stored in the DRAM. By refreshing the data several times a second, the DRAM controller keeps the contents of memory alive for as long as they are needed. So, DRAM is as useful as SRAM after all.

When deciding which type of RAM to use, a system designer must consider access time and cost. SRAM devices offer extremely fast access times (approximately four times faster than DRAM) but are much more expensive to produce. Generally, SRAM is used only where access speed is extremely important. A lower cost per byte makes DRAM attractive whenever large amounts of RAM are required. Many embedded systems include both types: a small block of SRAM (a few hundred kilobytes) along a critical data path and a much larger block of DRAM (in the megabytes) for everything else.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | **sales@docketalarm.com** | **1-866-77-FASTCASE**

fastcase®
Smarter legal research.