Novos™

Environments for Embedded Systems

Operating Systems for Embedded Applications

Matching the OS to the Application

by

Tom Barrett
Embedded Environments Co.
2018

Embedded Environments Co.
www.novos.us

Copyrights

Disclaimers

Trademarks

Version #	Date	Author	Revised Sections
1.0	23-May-2018	ATB	Initial release

Contents

Figures

Tables

Examples

PART 1

Architecture benefits, Application Tasks and Real-Time Needs

Chapter 1

Introduction

There has always been a variety of opinions amongst embedded system developers about the best system architecture for their applications. The subject has endured for over forty years and continues to this day. The debate tipped in favor of using a defined software architecture primarily because of the growth in the complexity of embedded applications.

Developing a complex application that incorporates a communications protocol, file-based storage, graphical output along with data acquisition and control is virtually impossible to do without some organizational structure on which to build the application code. But even in small systems, a good argument can be made for an organized structure for application development.

Those for whom a system architecture is not a big consideration tend to be developers of small or very simple applications. There are still a lot of these applications around due to the steady decline in the prices of microcontrollers and the scale of peripheral devices integrated into them. With the Internet of Things (IoT) starting to burgeon, there is an increasing likelihood of growth in that segment of the embedded systems world.

The subject of system architectures in general is large and literature about them is extensive, far beyond the space available here. Many system architecture implementations exist for a broad spectrum of applications but this document will confine the discussion to an overview of those system architectures primarily used in embedded systems.

Any products directly mentioned in this document are the trademarks of their developers.

Scope

In the world of embedded systems, there are many system architectures, or, environments, that provide a foundation on which the application developer designs and implements the software that becomes the application. As befits the applications themselves, some system architectures are better suited to a given type of application than others. For example, the cruise control on an automobile has tight timing needs whereas a smart thermostat does not. They are both considered embedded applications but their architectural needs differ greatly.

The intent of this document is to provide a top-level, generic overview of system architectures commonly used in embedded systems. Some of them have real-time capabilities and some do not. Nevertheless, they are useful in a broad range of embedded applications where their capabilities match the needs of the application.

The information provided in this document is general and does not pertain explicitly

to any of the Novos family of operating system environments for embedded systems. If the reader wants to delve more deeply into the subject, the source code for the Novos operating system environments is freely available at www.ee-novos.com and can serve as a sound basis for further exploration.

Benefits of a System Architecture

Whether large and complex or small and simple, a well-designed and documented system architecture provides a number of tangible and/or financial benefits to the developer. It

- simplifies application development and improves developer productivity by utilizing high level abstractions (i.e., objects) to ease the handling of complex functions,
- provides a solid development infrastructure that provides consistency and repeatability,
- abstracts away the complexities of the processor, making it easier to migrate the application to another processor to meet pricing or competitive requirements,
- optimizes use of system resources and improves product reliability, maintainability and quality of the application, and,
- manages resources and serves and the point of integration for middleware.

A system architecture provides a way to bring all those elements together into a platform that allows the application developer to begin development at a much higher point, which enables a shorter time-to-market with higher reliability and lower risk. Most importantly, valuable engineering time is directed towards domain-specific endeavors that advance the system development rather than being spent trying to create an architecture for the application. In the final analysis, it is the domain-specific work that yields the financial return for the application developer, not the work on developing an ad hoc system architecture.