Handling complexity in embedded application development: an example of real time MP3 streaming over Bluetooth

Author:

Trish Messiter, Clarinox Technologies Pty Ltd, 8 Bayside Cres, Hampton VIC, AUSTRALIA, Ph:+61 3 9502 0728, Fx:+61 3 9502 0729, trish@clarinox.com

Abstract

Continuing to use the traditional approach applied to embedded design into the modern complex designs can become chaotic and ad hoc. This paper looks at the issues confronting modern embedded application design and the embedded eco system that exists to enable the combination of standard hardware, proven RTOS, middleware and applications to present a solution for these issues.

Introduction

Today more microprocessors around the globe are used in embedded systems than in PCs and those already large numbers are increasing at a phenomenal rate.

Traditionally the requirements placed upon embedded systems are quite different to those applied for desktop computing. This is because embedded systems are, in general, designed to accomplish a very specific task or group of tasks there is no single characterization that applies to the whole gamut of embedded systems. However some combination of the variables of robustness, small size and weight, real-time requirements, long life cycle and low price could be expected to figure in the design criteria for most embedded systems.

Less tolerance for malfunctions in some cases may be simply a convenience and cost issue, such as the lack of permanent I/O connections which makes debugging more difficult, or it can far more serious such as the failure a mission critical component which could have extreme consequences.

Real time requirements combine the constraint of time and correctness - not only does the computation need to be correct but it also needs to be at the correct time. This requires an estimate of the worst case performance, which on complicated architectures can be difficult, and leads to overly conservative estimates. Mission Critical systems as a class have a significant requirement for real time operation in order to meet external I/O and control stability requirements.

Low price translates to reduced resources such as processor speed and memory size which in turn constrains software development and execution. Often embedded devices are very sensitive to cost. A variation of even a few cents per device can be significant due to the huge multiplier of production quantity combined with the higher percentage of total system cost it represents.

These constraints, though difficult, were manageable, if the application is simple and small enough to and run without the need of an underlying operating system. This changes once there is a need to manage many variables such as serial, USB, TCP/IP, Bluetooth, Wireless

LAN, trunk radio, multiple channels, data and voice, enhanced graphics, multiple states, multiple threads, numerous wait states and so on.

This increase in complexity has made the use of a proven robust and networked RTOS complete with file system the most efficient course in most cases. For many designs however this is not sufficient and further middleware, protocols and drivers are required to achieve the desired aims.

Application and Supporting Infrastructure

This paper discusses the implementation of streaming of MP3 Audio data via Bluetooth, and how the various challenges were addressed by the use of appropriate infrastructure: an x86 processor with QNX Neutrino RTOS and Clarinox Technologies embedded framework with incorporated Bluetooth protocol.

A distinguishing feature of Bluetooth is the sophisticated service discovery mechanism that allows for devices to establish connection, transfer data and disconnect without requiring user intervention. User approval may however be required for authentication and security purposes.

Bluetooth also caters for many devices to co-exist in the same space. These devices may be part of the same piconet, they may be connected in scatternet topology, or may have no connection to each other at all.

The standard defines all layers of the OSI protocol stack layer, which translates to a focus on application development inherent in the specification design.

The Bluetooth reasonably low power rates of around 1mW Class 3 (100mW Class 1) during transmission make it suitable for use for handheld devices. Added to this is various low power modes utilized during idle periods which can be under the control of the application.

All these attributes suggest Bluetooth is ideal for networking of all sorts of electronic devices within 100m of each other however; the use of Bluetooth is not without challenges such as limited bandwidth, high degree of error rates, and the time-varying nature of the radio link.

The Bluetooth protocol stack consists of several layers, starting from the physical layer, comprising the Radio, BaseBand, Link Manager, Logical Link Control and Adaptation Protocol, and Host Controller Interface as shown below, in Figure1.

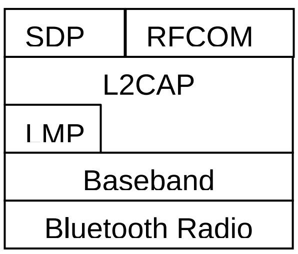

Figure 1: Protocol stack of Bluetooth

The functions of the various layers mentioned are:

- Baseband is the physical layer where Bluetooth performs all low-level data processing including basic Forward Error Correction (FEC) and Automatic Request for Transmission (ARQ), packets handing, data whitening, hop selection and security.

- o Link Manager Protocol (LMP) [9] handles link control, power-sensitive states changing, and data encryption.
- o Logical Link Control and Adaptation Protocol (L2CAP) [9] provides both connection-oriented and connectionless data services to upper layer protocols with segmentation and reassembly operation. L2CAP only supports ACL links with packet size up to 64Kbytes. L2CAP and LMP serve as a Media Access Control (MAC) layer.
- o Host Controller Interface (HCI) [9] provides a uniform interface method to access hardware capabilities. It is responsible for transmitting data between L2CAP and baseband through a physical bus (e.g., USB, RS232 and PCI), using LMP.

Hardware and software infrastructure that was capable of supporting the relatively large code base of the Bluetooth stack, required profiles and application were required.

The Eden x86 processor, from VIA technologies was selected due to the reasonably high performance, low power and passive cooling attributes. The Eden is based on the C5P Nehemiah core and delivers enhanced digital media performance and is designed to operate with passive cooling due to its ultra low power consumption. More information can be obtained from VIA technologies web site listed in the references.

The QNX Neutrino RTOS provides robust microkernel architecture complete with POSIX Support, IP Networking Technologies and file systems which were the main criteria for this application. An overview is provided in Figure 2, adapted from QNX web site.

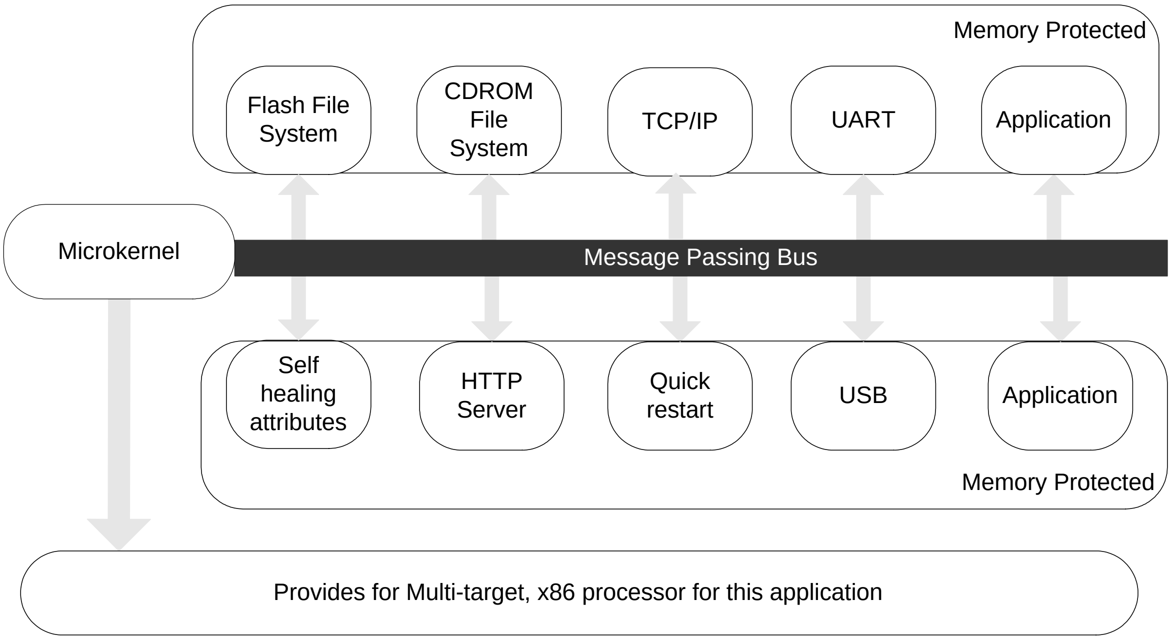

Figure 2: QNX Neutrino Microkernel overview

The Clarinox Technologies embedded framework with incorporated Bluetooth protocol provides the necessary infrastructure for:

- o Threading
- o Timers
- o Semaphores
- o Mutexes
- o Dynamic memory management without fragmentation
- o Inter-process message passing

- Event/Message handling
- Finite state machine
- Serial device driver encapsulation
- USB device driver encapsulation
- TCP/UDP Socket encapsulation

This functioned as an extension to the debugging tools and Board Support Package or Hardware Adaptation Layer provided by the RTOS to enable the handling of complex multi threaded applications. Some examples of the provisions include:

- dynamic memory management. ClarinoxSoftFrame provide a simple and effective memory management module to replace C style (malloc/free) or C++ style (new/delete) calls. Inside these calls are the smart, and adaptive memory and pool management that does not result with memory fragmentation, yet fast and efficient.
- standard libraries and stream libraries guarantee are provided that the code works the same way on each platform.
- Debug mode function profiling, function entry/exit tracing and indented display of threads and functions with timestamp would prove the value of presenting information. Especially, while trying to find a problem that happens in a system with a large number of threads and protocols running concurrently

The infrastructure is depicted in Figure 3.

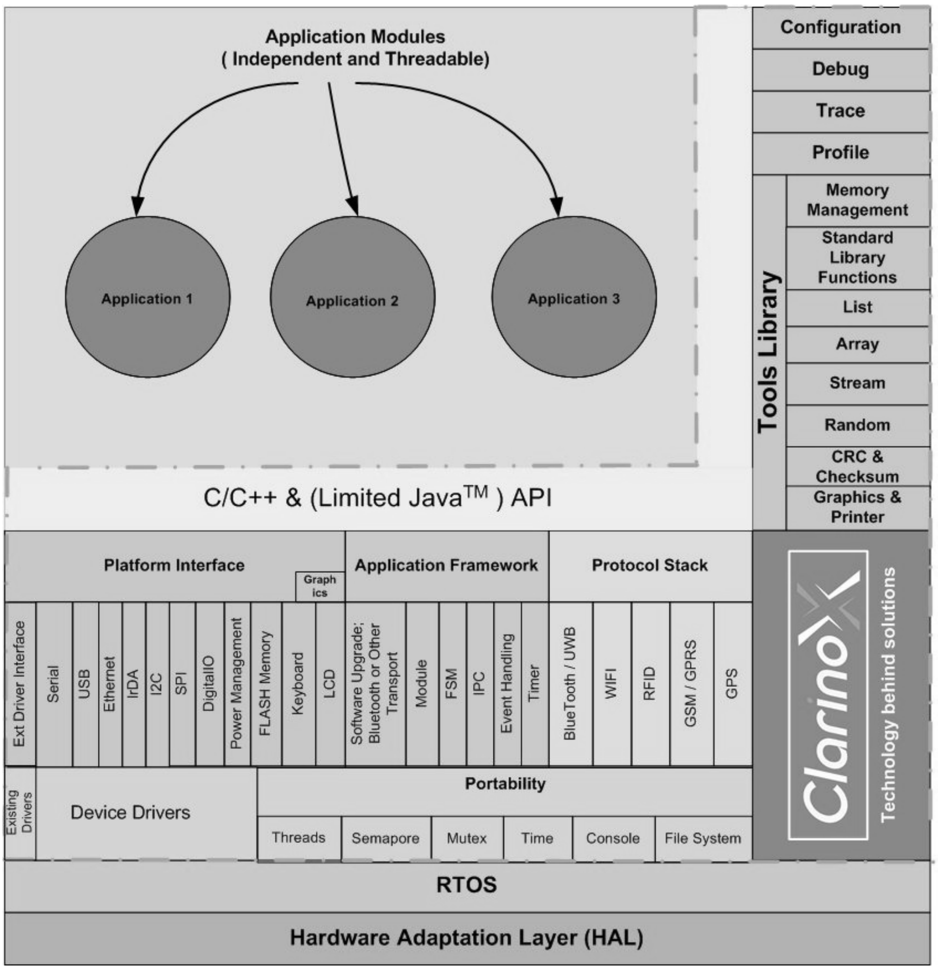

Figure 3: ClarinoxSoftFrame structure

Together these elements enabled this development to proceed smoothly with reduced errors, reduced development time and reduced complexity.

Results

The real time streaming of MP3 over Bluetooth placed high demands on both hardware and software. The successful solution used an eco system where the use of standard hardware was combined with proven RTOS (robust, with file system, networking etc) and embedded middleware attaching to RTOS and hardware to provide the object oriented encapsulation of service based object oriented application development.

References & Further Suggested Reading

http://nicta.com.au/director/research/programs/ertos.cfm
http://ptolemy.eecs.berkeley.edu/~kienhuis/dacSlides/intro.pdf
http://www.ece.cmu.edu/~koopman/iccd96/iccd96.html
Ling-Jyh Chen, Rohit Kapoor, Kevin Lee, M. Y. Sanadidi, Mario Gerla, Audio Streaming Over Bluetooth, An Adaptive ARQ time out approach
MPEG Layer3 Bit stream Syntax and decoding, www.mp3tech.org
MPEG Layer 3 Frame Header Documentation, www.mp3tech.org
MP3 Play Lib, mpg123 player www.mp3tech.org
RTP/RTCP RFC 1889 www.ietf.org/rfc/rfc1889.txt
Staffan Gadd & Thomas Lenart, A Hardware Accelerated MP3 Decoder with Bluetooth Streaming Capabilities
Specification of Bluetooth System Core www.bluetooth.org (in particular Advanced Audio Distribution Profile / Audio Video Distribution Transport protocol specification)
Wang Xiaohang, Video Streaming over Bluetooth a survey
VIA Technologies: http://www.via.com.tw/en/products/processors/eden-n/
QNX: http://www.qnx.com/products/rtos/glance.html
Singh, Mritunjay, Trish Messiter, Gokhan Tanyeri MP3 Audio streaming Over Bluetooth, WiCon Aisa 2005
Messiter, Trish, The place of Wireless, Radio Communications, 2004
Clarinox Technologies: www.clarinox.com