DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

Open Journal of Electrical and Electronic Engineering

Short Communication Open Access

Software driven approach for Embedded Devices

Hrvoje Dodig*, Joško Šoda and Ivana Golub

University of Split - Department of Maritime Electrical and Information Technology, Faculty of Maritime Studies, Ruđera Boškovića 37, Split, Croatia

***Corresponding Author:** Hrvoje Dodig, University of Split - Department of Maritime Electrical and Information Technology, Faculty of Maritime Studies, Ruđera Boškovića 37, Split, Croatia, Email: hdodig@pfst.hr

ORCID: Hrvoje Dodig: https://orcid.org/0000-0001-7910-9583
Joško Šoda: https://orcid.org/0000-0002-5825-8026
Ivana Golub: https://orcid.org/0000-0002-1420-3197

Received Date: *Feb 24, 2020 /* ***Accepted Date:*** *Mar 03, 2020/* ***Published Date:*** *Mar 05, 2020*

Abstract

This paper presents the possible new design paradigm that emerged during the author's design of an embedded communication device for Croatian Navy. Prior to codesign techniques that emerged in 1990's the traditional embedded design methodology involved problem specification, separate hardware and software specification, integration, and the system test as the final step in the embedded device design. Such an approach can potentially lead to numerous iterations and can increase the cost of the development cycle because there are no guarantees that separately developed software will work well with separately designed hardware. Codesign techniques, on the other hand, delay the decision to which components of hardware or software will be used for embedded system until late stages of embedded design process. At the time of the invention of the codesign techniques this seemed as perfectly balanced approach between design of hardware and software spending about equal time in the design of both hardware and software components. However, since the 1990's the design of embedded devices has changed; nowadays the most working hours are spent in the design of software while the design of hardware requires less working hours due to extensive choice of IC's and supporting electronic circuits, and due to advancement of EDA software tools. In favor of the software-driven approach presented in this paper, it should be noted that nowadays, there is a large number of freely-available software components and libraries which, if properly utilized, greatly expedite the development of the software part of the embedded system design. Therefore, perhaps it is a suitable time for a new paradigm shift where the design of the hardware is completely dictated by the design of software, and the design of the hardware is simply the matter of selecting proper IC's and other electronic circuitry that supports the software. In this paper, we present an example of the embedded design using this software-driven design strategy. By the end of this paper, it is shown that software-driven design not only allows the rapid prototyping of embedded devices, but it reduces the possibility of design errors, as well.

Keywords: Embedded design; Hardware-software codesign; Software driven design; ARM technology

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

Cite this article as: Hrvoje Dodig, Joško Šoda, Ivana Golub. 2020. Software driven approach for Embedded Devices. OJEEE. 1: 01-08.

Introduction

An embedded system is the special purpose computer system used for single purpose, such as domestic appliance, parking car counting device or a smart phone. Embedded system can even be the subsystem of the larger system such as medical device or a number of embedded systems can be interconnected such as in shipborne communication system.

Prior to codesign techniques the traditional embedded design methodology involved problem specification, separate hardware and software specification, and finally the integration and the system test. Such approach did not allow, for example, early fault identification of the system's architecture. Furthermore, some hardware tests could not be performed without fully functioning software. This led to potentially many iterations of hardware development until the product is in the final production stage.

In early 1990's the hardware/software codesign emerged as a new discipline for embedded system design [1]. Because of the technological advances an adoption of codesign techniques became the necessary tool for embedded system companies. Codesign techniques involve steps such as specification, which is a common specification that specifies both hardware and software components. Other steps in codesign are synthesis and convexification where software and hardware are tested simultaneously to verify both hardware and software components.

In the embedded system design the larger portion of time is allocated to the development of software than it is to the design of the hardware [2]. Furthermore, the software design tends to be more complicated then hardware design because of the required functionality. This is especially true when GUI (graphical user interface) design is required. The hardware design, in most cases, relies simply on selecting the appropriate components to achieve software design requirements. In today's IC market the electronic component manufacturers provide IC's with almost any standard functionality.

Furthermore, nowadays, there is a large number of software tools available and variety of code libraries and code elements that can be reused. Thus, the right combination of software tools can potentially lead to rapid development of product and could lead to better product in terms of reliability and quality. The question that really needs to be answered here is what components (IC's) do support the software components and libraries?

For the afore mentioned reasons, instead of codesign technique, in this paper we present the software driven embedded design of shipborne communication system. We define the software driven embedded design as the design methodology that combines the hardware and software in such way that software requirements dictate the final decision on the hardware components.

Communication system requirements and design

In maritime design of shipborne communication device many considerations that are related to the operating environment of the device had to be taken into account. For example, the display of the device had to support low light conditions such as in engine room and it had to be visible while exposed to direct sunlight. Because the engine room is

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

noisy environment the care had to be taken in order to discriminate the human voice from the environment sounds. The design had to be electrically and mechanically robust in order to support conditions with increased salinity and temperature. Moreover, because of the cost constraint relatively inexpensive display is used and user interface (UI) system called µWin was developed to support such non-standard display. At the same time the system had to encrypt/decrypt large number of UDP packets while simultaneously processing microphone input and speaker output. For all these reasons, the software design in this case is order of magnitude more complicated than hardware design. Therefore, we have decided to take the software driven approach in our development which in turn dictates the hardware development.

Functional Requirements on the Ship's Communication Device

The functional concept of the shipborne communication device is shown in figure 1. At the core of the embedded system there is a CPU powerful enough to deliver the desired level of functionality. Because the device might be installed outside of ship's chambers, one of the important requirements is that display should be visible in sunlight conditions, therefore, limiting the choice of displays to OLED technology.

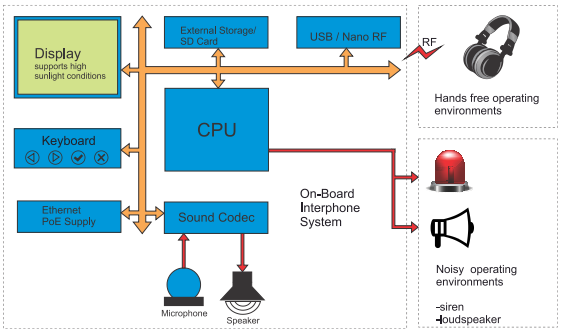

Figure 1: The functional layout of the communication device.

The device is intended to be connected to ship's LAN network which offers an opportunity to simplify the device's power system and the installation of the device by utilizing ethernet's PoE (Power over Ethernet) capability. This approach effectively eliminated the need for device's batteries, for wall electric power socket and for device's power supply circuitry.

Because the communication device is intended to operate in noisy and dark environment of the engine room there are several considerations to take into account: first in order to enhance the visibility of display in the engine room conditions the additional requirement for the display is to have back-light functionality. Second, because of noisy environment and because the engineers in engine room should operate the device hands-free, we have added wireless USB-Nano headphones with built-in noise cancellation. Third, again because of the noisy and dark engine room environment we have decided to add siren to visually alert the crew, and the possibility of external loud speaker for those occasions when ship's captain or officer need to speak to several crew members at once.

The network layer is another important aspect to consider since the communication device is intended to operate as VoIP (Voice over IP) device which is connected via LAN to ship's communication central. Furthermore, because the requirement for the device is to support one-to-one conversation, one-to-many conversation, group conversations and conference calls each of these functionalities had to be carefully implemented. Related to this, since the communication device is currently installed on operating ships the special care had to be taken about the network security and packet encryption.

Finally, it was our intent to make the device user friendly and flexible and to allow some degree of user's custom ability. This means that to some degree and with ship's officer permission, the crew members are allowed to change ring tones, the layout of certain menus, to add other crew members to phone book and similar and to play the background music. All this information is stored on SD card which serves two purposed: first the crew's customization information is stored on SD card and second it allows the software updates since

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

the parts of the operating system and software is stored on SD card. Furthermore, in the unlikely event of device's fault some important service data is stored on SD card which allows the service engineers to inspect this data and to detect the cause of the fault.

Software System Layout

The general layout of software components for ship's interphone system design is shown in figure 2, where software components are shown in top-down order according to the level of complexity and interdependency. Hardware is mentioned simply to clarify the necessity of hardware abstraction layer (HAL) which is the software component, in most cases, available from CPU manufacturer. Because of the situation on the market, our goal was to reuse the code components which are freely available from various sources. Because one of the constraints of the project was to reduce the cost, we decided to use Free RTOS because we only needed support for minimal threading model. Furthermore, the minimum version of Free RTOS takes only 86 bytes of RAM, therefore too advanced CPU should not be required.

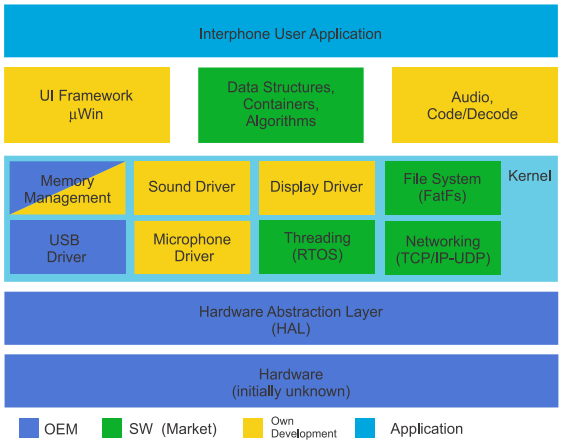

Figure 2: The layout of software components of embedded interphone design. Free software available from different sources is marked with green color.

In the design of software components there is a need for standard data structures, containers and algorithms such as lists, queues, stacks, trees and networks. This is especially true in GUI design where menu's, drop-down lists are placed in tree structures. Thus, in order to expedite the production of ship's intercom system the natural choice to support these structures was STL. The standard template library (STL) is the set of ready-to-use templatized standard algorithms, containers and functionalities [3] and in order for STL to work the programming language must be C++. This placed another restriction on the choice of CPU, that is, the processor that was at the core of our embedded communication system design had to be selected with the aim to support C++ programming language. Since nowadays most of the ARM processors support GCC for ARM which is C++ compiler, the natural choice of CPU was the one that supports ARM technology.

In figure (2) the schematic representation of the software system is shown where the parts that come from original equipment manufacturer (OEM) are shown in purple color, software components that are freely available from the market are shown in green color and software components that were missing and needed in-house development are marked with yellow color. On top of previously mentioned software layers comes the application layer which is responsible for integration and control of all these components and for user interface and ship's intercom functionality. The Figure (2) was created after market analysis which software components are freely available and easy to integrate into ship's intercom system and one of its purposes is to serve as guideline for later hardware component selection.

Thus, after analysis, the following software components had to be developed to ensure the intended functionality of the communication device:

- Microphone driver - the driver that converts analog microphone sound signals to digital signals with acceptable sampling frequency and with enough quantization levels to ensure the audio quality. Furthermore, to ensure the voice quality the device is required to have two microphones for stereo sound.

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

- Sound driver - this is two channel stereo driver which is responsible for reproduction of voice audio at range of audio frequencies.
- Display driver - displays the characters on the screen at desired pixel location, displays the rudimental 2D graphics such as lines, circles, rectangles and round rectangles and allows the bitmap drawing. In addition, it regulates the level of backlight.
- Memory management - since for small embedded systems the available RAM is limited resource one of the main concerns was to avoid memory fragmentation that naturally occurs when memory is allocated dynamically. Memory fragmentation was completely eliminated by employing some special memory management algorithms.
- Audio Code/Decode - this driver is responsible for decoding encrypted audio that comes from LAN network and to provide the audio data to sound driver at required playback frequency. Furthermore, it codes the data from microphone and it prepares and encrypts the data packets for transmission over the LAN network.
- UI Framework (µWin) - this driver is essentially the windows management system similar to windows management system on commercial operating systems such as MS Windows. Its main task is to maintain z-order, perform windows clipping, windows drawing by implementing painter's algorithm and to respond to external events such as keyboard or inbound call.

Due to large number of tasks that are required to be performed simultaneously the processor to be selected is required to support the large number of DMA (Direct Memory Access) channels. If the large number of DMA channels are supported in hardware then the processor core would be relieved of tasks such as: copying the buffers to audio playback ship, copying the data from microphone to audio buffer, LAN packet retrieval, writing buffers to display and similar. For example, if DMA is used then the task of drawing the pixel to the display is reduced to the task of writing the data to designated memory area used for DMA access to display. Finally, the user interaction with keyboard is intended to be implemented via standard interrupt mechanism available on all modern processors.

Hardware component selection

Due to software requirements and analysis presented in previous section the natural choice for core processor is processor based on ARM technology. The ARM technology is the set of processor architecture guidelines that are provided by Arm Holdings Ltd. in order to unify processor architecture and instruction sets across various processor manufacturers [4]. The instruction set unification came in the form of Thumb and Thumb-2 instruction sets [5]. This approach guarantees the code compatibility and reusability on processors that come from different manufacturers and these guidelines are issued almost yearly. Most of the main processor manufacturers have adopted this standard and nowadays it is estimated that majority of processors on the market support ARM technology which evaluates to about 15 billion units sold only in 2015 [6]. This means that nowadays the ARM technology is present in most of the mobile phones (especially those based on Android), tablets, television sets, computers, home appliances and similar.

One interesting consequence of this architecture unification is that the compiler tools are unified as well. For example, GCC for ARM is freely available compiler for ARM and it was used in this project for software development. Furthermore, the debugging and programming of ARM processors was unified as well. In the old days, the code developer for microprocessor, in order to program effectively had to buy an expensive processor emulator for particular microprocessor device and then develop and debug the code on that emulator (for example with Keil µVision). Then the developer had to use chip programmatic device (which also could be expensive) to program that particular microprocessor chip. The need for this emulation software was effectively eliminated with the emergence of ARM technology because ARM technology provides

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

simple unified hardware interface for programming and debugging called JTAG interface. The hardware debugging removed the need for expensive emulator software because all the debugging, setting break points, following the code flow is now done in hardware. The task of GUI debugger is now to simply follow the code flow and to communicate via JTAG interface. This allowed for free ARM debuggers such as System Workbench for STM32 which are au par with commercial counterparts such as Keil µVision. Additionally, because the GCC for ARM was chosen as the compiler tool of choice the access for code and standard algorithm libraries such as STL is guaranteed. All of these reasons amounted to decision that processor based on ARM technology is our processor of choice.

Before the emergence of ARM technology, the microprocessors came in two flavors: central processing units (CPU's) and microcontrollers [7]. The purpose of the CPU is to execute program code and to communicate with peripherals such as USB, RS232, A/D or D/A module, etc. [8]. On the other hand, microcontrollers are small devices which were able to execute the code, however, microcontrollers had peripherals integrated, as well [9,10]. This means that on the single chip both CPU and peripherals such as USB, RS232 were integrated together. For this reason, the microcontrollers are referred to as SoC or system on chip. Of course, this integration of CPU and peripherals came at the price of lower CPU processing capabilities and the intent of the microcontroller is to operate in small embedded devices such as home appliances.

With the emergence of ARM technology, the need for microcontrollers was recognized and this gave birth to ARM Cortex M microprocessors. The Cortex M series of ARM microprocessors are packed with peripherals such as: LAN, CAN, RS232, IrDA, A/D, D/A, USB 2.0, USB 3.0, TFT display support, SD card interface, CCD camera interface and similar. The first Cortex M was Cortex M-0 while most modern ARM Cortex M microprocessor are designated as ARM Cortex M-7.

Since our communication device was designed with the intent to be relatively cheap ARM Cortex M series was natural choice for device's development because it allows to reduce the number of other electronic components (i.e. for USB and LAN communications etc.). At the time of device's design (2015) an award winning STM32F4 processor caught our attention: it was 32bit ARM Cortex M4 processor, packed with various peripherals and with the full range of freely available software ecosystem such as peripheral driver’s graphics drivers and similar. Its DMA system was very simple to set up by simply designating the region of memory used for DMA and, furthermore, almost all peripheral devices could be connected to DMA (such as microphone, sound codec, graphic display, LAN, USB). This alleviates the processing burden from microprocessor allowing it to spend the most of the processing time executing application stack (GUI, user requests etc.). On top of this, STM32F4 is equipped with floating point unit, runs at 180 MHz and is equipped with DSP instruction set, hardware calendar (required to display device's time) and had remarkable 225 DMIPS instruction through output.

Once the core CPU for ship's communication device was selected using the software criteria the remaining hardware components had to be chosen. This was relatively straightforward task, for example, the chosen sound codec was stereo codec WM8731 from Wolfson microelectronics. The chip came with integrated headphone driver, stereo output to speakers and stereo input from microphone. Since one of the requirements for software sound and microphone driver was programmable sampling/playback rate WM8731 was the codec of the choice because it supported this functionality in hardware for audio sampling frequencies from 8kHz to 96kHz. With WM8731 the sampling rate can be set simply by setting appropriate chip's register. Furthermore, one of the requirements for communication device was to have the mute button. Again, WM8731was chip of choice because muting is controlled by register and the task of muting/unmuting is reduced to set or

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

clear appropriate bit in the register. Furthermore, WM8731 works seamlessly with SMT32F4 DMA buffers using I2S (Inter-IC sound) which is supported in hardware by STM32F4 microprocessor. The microphone's used were electret condenser microphone CMR-2747PB-A coupled to low noise audio amplifier MCP6021 in order to match the microphone output impedance, therefore, minimizing the signal loss from microphone.

Another important point in hardware development of the ship's communication device was interfacing LAN to STM32F4. Although STM32F4 provides ethernet interface we still had to adjust electrical levels of ethernet signaling in order for core processor to receive and send UDP packets required for VoIP. This was achieved with DP83848 ethernet transceiver from Texas Instruments, the chip which adapts the physical level of LAN signals to the expected signal levels from core processor which are expected to be at 3.3V.

Furthermore, because the communication device was expected to operate in harsh environment, that is, increased temperature in the engine room and in cold environments if placed on outside areas of the ship the military grade components were selected to ensure device's functionality in harsh environments. The final product, the PCB of which is shown in figure (3), was tested in climate chambers and for water-tightness under increased pressure conditions.

Conclusion

In this paper we have presented the design process of ship's communication device, the development of which was almost completely driven by the software. By designing the software functionalities first and by making use of ARM technology we were able to start the software design process long before the electronic components of the device were selected. The advantage of such design process is that the waiting time for hardware component acquisition and assembly is eliminated and this time is allocated into software design process which takes significantly more time than the hardware design. Once the core processor was selected to meet the requirements of the software design the choice of other components was dictated mostly by the core processor (which again was chosen by software) and software requirements. This approach allowed us to start software development cycle before-hand and therefore reduce overall product design time. The approach of software design first was greatly supported by the availability of free software components and it would probably not be possible in 90's and 80's. Finally, the cost of the design in terms of the equipment and in terms of developer software tools was significantly reduced by choosing freely available software tools and by choosing appropriate technologies.

Figure 3: Finished PCB artwork of ship's communication device.

Refrences

1. Teich J. 2012. Hardware/Software Codesign: The Past, the Present and Predicting the Future. Proceedings of the IEEE. 1411-1430. Ref.: https://bit.ly/2PFtfZu

DOI: https://doi.org/10.36811/ojeee.2020.110001 OJEEE: March-2020: Page No: 01-08

2. Kraeling M, Oshana R. 2013. Software Engineering for Embedded Systems, 2nd ed, Waltham, USA: Elsevier. Ref.: https://bit.ly/3amQRdl
3. Musser DR, Derge DJ, Sanni A. 2009. STL Tutorial and Reference Guide: C++ Programming with the Standard Template Library, 3rd ed. New York: Addison-Wesley. Ref.: https://amzn.to/3aoj6sm
4. Sloss A, Symes D, Wright C. 2004. ARM System Developer's Guide, New York: Morgan Kauffman. Ref.: https://bit.ly/2PFtEuY
5. Seal D. 2000. ARM Architecture Reference Manual, 2nd ed., New York:Addison-Wesley. Ref.: https://amzn.to/2IdeGbN
6. ARM Holdings Investor Relations, The financial reports of ARM Holdings, available online at: https://www.arm.com/company/investors/financial-results, retrieved 15. March 2015.
7. Betker MR, Fernando JS, Whalen SP. 1997. The History of the Microprocessor for Bell Labs Technical Journal. Autumn. 29-56. Ref.: https://bit.ly/3cjN1Ua
8. Alpert D, Avnon D. 1993. Architecture of the Pentium Microprocessor for IEEE Micro. 13: 11-12.
9. Mazidi MA, Mazidi JG, McKinlay RD. 2006. The 8051 Microcontroller and Embedded Systems, New York:Pearson/Prentice Hall.
10. Uma RDK. 2010. The 8051 Microcontrollers: Architecture, Programming & Applications, New York: Pearson.