International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 14 (2015) pp 34544-34548

Software Oriented Development Of Junior MCU Model System

Victor A. Rusakov
Department of Cybernetics, National Research Nuclear University MEPhI (Moscow Engineering Physics Institute), Kashirskoe highway, 31, Moscow, 115409, Russia, VARusakov@mephi.ru

Abstract- This article describes the approach for younger models (8-bit) of the microcontrollers where the focal point of the embedded systems development is the application software. Controllers play role here as a matching hardware components with complicated, but typical functionality. The description of the HAL (hardware abstraction layer) as a way to ensure the portability of code is developed in accordance to small resources of the junior models. The information is provided regarding the experimental justification of the approach, also some issues related to the language aspect of such approach development are disclosed.

Keywords- software, microcontroller, hardware abstraction layer, portability.

1. Introduction

Almost any modern device contains control embedded system based on microcontroller (MCU).Typical MCU is an integral microchip, where the following is located: CPU with general purpose registers, RAM and program memory, as well as several different types of interface modules: analog-digital converters(ADC), analog comparators, different input-output (i/o) interfaces UART, USB, SPI, I2C, CAN.

The use of microcontrollers allows to place the logic of the device in the software (SW) of one or more MCU.

The presence of various purpose analog and digital interface modules of on-chip MCU, in addition to the processor node, can reduce the number of required electronic components on the system board. As a result it becomes possible to simplify the hookup of the device, to increase its reliability and to improve consumer characteristics - reducing the size and weight and decreasing the power consumption.

The most important stages of the development cycle (see fig.1) are: the development of the main algorithm, the selection of the MCU type and decomposition of the system into hardware and software components. Incorrect decisions made on the initial stages of the project lead to the ultimate delaying of the development/ implementation cycle and increase the price of the system.

Lets reveal the main challenges designers are faced in the earlier stages of project development.

Development of the main algorithm. The development or selection of the control algorithm is an innovative task. The choice of the algorithm depends on the functional requirements as well as on the developer's preference and experience. In real life, two different developers can approach the same task in different ways.

Selection of the microcontroller type varies depending on several circumstances:

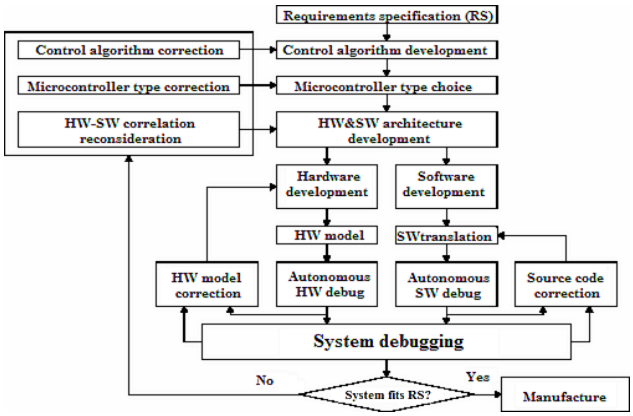

Fig.1 The typical development cycle of MCU system

- The existence of the large number of microcontrollers types and models with different characteristics. MCU resources should be sufficient for the new algorithm implementation. Keep in mind that in the future the functional requirements of the device could be extended that will lead to the software modification. Subsequently some extra resources are required that is hard to plan in advance. Take into consideration that as MCU resources are growing the price is rising as well.
- The selection of the MCU is dictated by designer's personal preferences, knowledge and experience. This means that a designer is using a narrow range of microcontrollers, since the assimilation of a new family of MCU requires considerable efforts for learning of a new device, as well as spending money on the acquisition and development of debugger firmware and IDE (integrated development environment).
- Software incompatibility of different types of microcontrollers. The incompatibility is caused by differences in the debugger firmware and IDE. As a consequence if you want to transfer programs between different microcontroller's families you have to rewrite a large piece of the code.

Development of a hardware and software architecture. At this stage of the development it is determined which functions will be assigned to the MCU software and which functions - to MCU hardware and additional electronic components. In this case the decomposition into hardware and software components depends on the MCU type that was chosen at previous step.

As we can see how it was described from the standard approach, despite of the presence of software components in the development cycle, just system hardware is conceptually leading. Obviously there are limitations in this approach for a wide class of systems, This is because of

historical reasons apart from already mentioned personal technological preferences. First simple MCUs and their peripherals were expensive by today standards, and the smallness of their resources didn't allow to develop functionally complex and relatively expensive application software.

Erroneous selection of the MCU could manifest at the final stage of system debugging (Fig.1). If it is detected that the system doesn't satisfy the requirements specification, the work has to start from the beginning. This leads to the delaying of the development/ implementation cycle and increases the price of the whole system. The increase of the price is the most critical for short-run or highly specialized products.

The development price for the large circulation of the product is recovered easily due to the large number of released products. At the same time even for the large scale products in case of real competition in the electronics market, it is required:

- The reducing of the price for the production unit. A consumer will choose the cheapest one among the products with similar characteristics
- The reducing of the time spent on the product development. The winner is the manufacturer, who will issue the new product to the market first. The decreasing of the time for the development also reduces the cost price of the product.
- The improvement of consumer's characteristics. In the struggle for the market manufacturers release new devices with improved characteristics such as higher performance, smaller size and power consumption. This motivates manufacturers who produce electronics to explore and use new MCU.

The competition between MCU manufacturers, relying on the achievements of microelectronics technology, eliminates the differences of hardware service levels for different microcontrollers of the same class while maintaining the inheritance and special characteristics of the development and implementation for MCUs of different manufacturers**.**

Such leveling means that all functional originality of the system is concentrated in the software. Therefore the most efforts should be concentrated on the development of the software system components with the subsequent selection of the controller with resources adequate to the debugged software.

The portability and hardware abstraction layer

The existence of multiple types and models of microcontrollers poorly compatible with each other and as a result the need to ensure portability of the code is a big obstacle of this development approach.

The problem of software portability between different hardware platforms is solved by using the hardware abstraction layer (HAL) [1, 2]. Such functional layer is a component of the low level of the operating systems (OS) and interacts directly with the hardware of a computer system.

The OS doesn't work with the hardware directly, but uses the services provided by the HAL at its upper interface. The presence of the HAL in the operating system allows you to install the OS on the computers with different architectures only by HAL replacement.

Figure 2 shows a simplified, generalized architecture of the HAL for such operating systems as Windows NT, Linux, ECOS. As shown, HAL provides management services by using interruptions and timers, as well as provides access services for IO registers. Applications and device drivers interact with hardware by using these services.

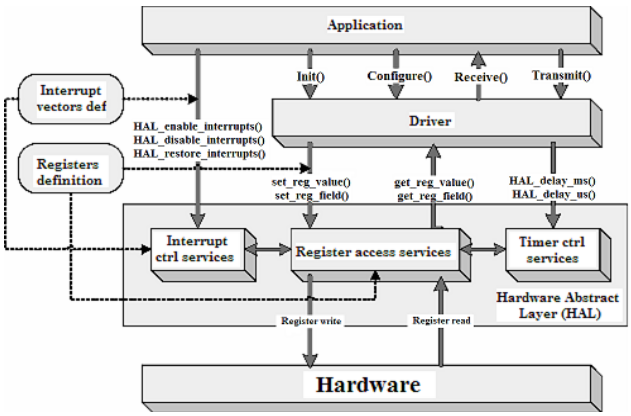

Fig.2 The Architecture of a hardware abstraction layer (HAL) of an operating systems

Architecture that is shown in the fig.2 mainly characterizes large systems, where the connection of peripheral devices is performed by using standard interfaces, such as PCI, USB, COM, PS/2 in the presence of a limited number of types for standard peripheral devices such as sound cards, Ethernet adapters, modems, keyboards, joysticks, etc.

Many different devices can be connected at the same time on the same bus, therefore HAL, (e.g. HAL of the Windows NT), allows to assign logical addresses according to the type of file handles for the devices on the bus. Hence drivers do not need to monitor the devices on the bus and their actual physical address.

The HAL feature that includes all HW-dependent code is the need to create such a layer for each type of target platform. In the course of its design it is necessary to form the upper interface, sufficient to the applications and to the operating system working with the equipment. This interface should not be changed when porting to another platform, otherwise the portability of the code will be deteriorated.

Hardware abstraction and low-power microcontrollers

Low-power microcontrollers are characterized by properties such as:

- Limited resources. Small memory size - from hundreds of bytes to hundreds of kilobytes, and relatively low clock frequency - to tens of megahertz.
- The variety of embedded devices and peripheral bus interfaces that have a substantial difference in the hardware implementation for various types of microcontrollers.
- The work in an exceptionally diverse hardware environment.

These circumstances prevent the use of the HAL with the architecture displayed on Fig.2 as a tool for providing hardware abstraction and portability of the software for the 8-bitMCUs.

The analysis of architectures and configurations of such MCUs has revealed the characteristics of their hardware that are very important for the subsequent development [3].

The presence of many built-in (internal) devices such as USART, ADC, analog comparators, PDM is typical for microcontroller as well as the presence of many peripheral interfaces and buses such as SPI, I2C, parallel bus, USB etc. that allowed the external peripheral devices to be connected. The control of the peripheral bus can be implemented by the bus controller that is built into the MCU, as well as can be programmatically implemented by using the universal parallel I/O port. So the drivers that interact directly with the hardware can be divided into internal drivers of MCU devices and peripheral bus drivers.

External devices can also be connected to some internal I/O MCU devices. The coupling of two devices (internal and external) will be formed. For example, terminal, RFID reader or other similar device requiring adequate management can be connected to the UART (RS-232). Therefore, the coupled driver can be added above the level of the drivers of the internal devices.

In a similar way the device on the peripheral bus can be controlled by the driver of the peripheral device. Such a driver needs to "know" the registers of "its" device, but needs to know nothing about the peripheral bus to which the device is connected. The access to the registers of the device connected to the bus is provided by the service of the driver corresponding to the peripheral bus.

Due to the described functionality , the driver for any peripheral device will be able to work with it no matter what interface the device is using to connect to the MCU and how this interface was implemented: by using peripheral interface controller or by the universal parallel port.

The proposed HAL architecture is presented in Fig.3. The drivers of the coupled and the peripheral devices are conventionally made at the application layer. The drivers of the internal devices and peripheral buses are placed in the HAL because exactly their application interface provides the platform-independent access to the hardware MCU for the upper levels of the software.

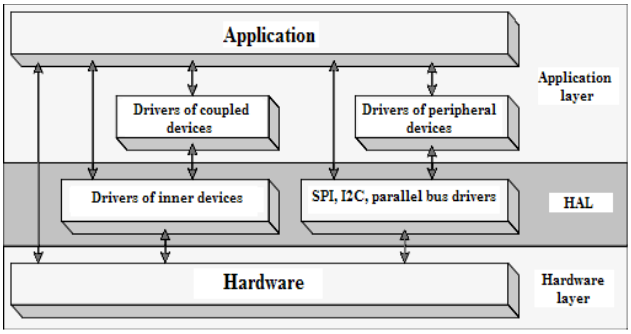

Fig.3 The proposed HAL architecture of microcontrollers

The application interface of the driver allows configuring easily the I/O devices, to arrange data admission and transmission hiding the features of the MCU hardware and thereby realizing the abstracting of I/O devices for microcontrollers.

In practice, this abstraction can be implemented through the libraries of routines (drivers) that support standard devices and interfaces such as: USART, 1-wire, EEPROM, ADC, etc. These libraries are often included in compilers (for example such as: Code Vision AVR [4, 5], CCS PIC [6], HI-TECH PICC [7]) or developed [8] by single enthusiasts. The presence of a standardized drivers application interface for similar devices in combination with the availability of the libraries of such drivers would allow the migration of the application code between different platforms without modification.

However, the studying of existing developments revealed that there are no such libraries supporting IO for microcontrollers of different types.So if it's necessary to transfer the application code to the MCU of another family, you will have to pick up a library with a similar application interface and to rewrite the application code for such a library, or to create your own library for application**.**

The drivers of the architecture in Fig.3 provide the initialization of the devices in different modes. Also the initialization algorithm of any mode is performed at the stage of program execution. With manual MCU programming (i.e. without using the drivers) the device initialization can be executed directly, which can save some memory. On the other hand, in some rare cases, the choice of the operation's mode of the device is really required during the program execution.

In comparison with the traditional code, driver function calls require more resources and longer execution due to the need of saving /restoring calling context, however this problem can be solved by using inline functions.

2. Experiments

To verify the proposed architectural solutions and the availability of resources that describe them, it was decided to provide the portability of a simple application that uses the ADC and UART as part of the MCU and the external real-time clock chip. There were implemented some drivers to support i/o such as: the ADC driver, the asynchronous serial interface UART driver, SPI bus driver and the driver of RTC chip DS1305 [9] as an external peripheral device on the SPI bus.

Two modern RISC microcontrollers were used as test platforms for developed drivers: MCU PIC18F252 [10] from the Microchip PIC family and MCU atmega8 [11] from the Atmel AVR family. Selected microcontrollers are widely used in simple devices and characterized by the low price, similar characteristics and set of hardware resources such as: a 10- bit ADC for 6 channels, USART unit, SPI bus controller, and are delivered in the same housing DIP28.

A universal high level language - C was chosen as a programing language. The lack of available compiler that supports both target MCUs became a problem. Therefore, the development and compilation of the code for MCU AVR was

performed by applying IDE Atmel AVR Studio 4 – the compiler AVR-GCC in the package WINAVR, and for MCU PIC –applying IDE-MPLAB 7.11 – the compiler HI-TECH C Pro 9.64. As a result there were implemented two independent driver packages for MCU AVR Atmega8 and PIC18F252.

The designed drivers allow performing initialization of devices in various modes, to turn on/off the device, to perform reading (UART, ADC, SPI) and data (UART, SPI) transferring. The clock driver provides the basic functions for setting and reading the date/time, for starting and stopping clocks. The clock driver uses driver of the SPI interface for communication with the chip.

The interface of the designed drivers permits the initialization of devices in various modes (including the specific target platform) to increase the flexibility of their usage.

As experimentally confirmed, the application of the designed drivers allows the following:

- The possibility to transfer the application code that uses modules ADC, USART, SPI between the controllers of the PIC and AVR families.
- The portability of the driver for DS1305 timer between different platforms using the SPI driver. And it is easy to add the support for 3-wire interface available for DS1305.

The realization of the full potential of the proposed approach requires means of the development support, permitting the following:

- To provide the descriptions of hardware resources of the microcontrollers and their corresponding HAL descriptions as components of development environment.
- To provide the control of shared hardware resources by the MCU device drivers. During the application's development you may experience hard-to-find conflicts between devices, due to the multiplexing of the MCU outputs. Such conflicts may reveal themself in unpredictable behavior of the system's periphery.
- To provide a consistent description and debugging of the application code followed by snap to specific MCU using already existing translators.
- To provide the automated selection of the target controller whose resources match the debugged software [12].

The special language and the appropriate development support environment could become such means.

3. Conclusion

Specific functionality of the relatively simple embedded system is accumulated in the application software, therefore the software development should be the core of the current development process of such system. The selection process of the MCU with its standard hardware functionality should be automated. Such approach to the system development requires the software portability.

The architectural mean to provide software portability is HAL – hardware abstraction layer. The adaptation of HAL configuration for small resources of 8-bit MC has been produced.

There were implemented some versions of HAL for the two test platforms in accordance with the proposed architecture for the application using ADC and UART in the MCU and an external clock chip. The universal language C has been used as a development tool, and common microcontrollers from Microchip and Atmel were used as the platform**s**. The portability of the applications has been confirmed experimentally.

Along with mentioned above the practice of creating HAL components revealed the insufficiency of the available tools for displaying the full potential of the proposed approach. It exposed the need to create new means of development support. A specialized language should become the basis of this process. The use of the specialized language will allow to put a routine part of the development process of the embedded systems on the development environment.

References

[1] Tanenbaum E., 2002, Contemporary operating system. 2-nd ed. – StPbg., Piter, 1040p.
[2] http://ecos.sourceware.org/docs-1.3.1/ref/ecos-ref.b.html. The eCos Hardware Abstraction Layer (HAL).
[3] Rusakov V.A., 2014, Software-in-centre 8-bit microcontroller embedded system development. Proceedings of the International Scientific-Practical Conference "Contemporary problems and ways of their solution in science, transport, production and education", vol.6, Technical Sciences, pp 64-69.
[4] http://www.hpinfotech.ro/html/cvavr_features.htm. HP Info Tech. Code Vision AVR.
[5] Lebedev M.B., 2008, Code Vision AVR: book for the beginners. – Moscow, Dodeca-XXI, - 592p.
[6] http://www.ccsinfo.com/content.php?page=compilers. Code Optimizing C Compiler for Microchip PIC® DSCs. Custom Computer Services Inc.
[7] http://www.htsoft.com. Embedded C Compilers and Tools for Software Development: HI-TECH Software. Microchip Technology Inc.
[8] .http://easyelectronics.ru/avr-uchebnyj-kurs-biblioteka-dlya-lcd-na-baze hd44780.html. Electronics for all. AVR. HD44780 based LCD library.
[9] DALLAS SEMICONDUCTOR DS1305 Serial Alarm Real Time Clock (RTC), 1997. – 20 p.
[10] Microchip PIC18Fxx2 Data Sheet 28/40-pin High Performance, Enhanced FLASH Microcontrollers with 10-bit A/D, 2002 – 330 p.
[11] Atmel 8-bit AVR Microcontroller with 8K Bytes In-System Programmable Flash ATmega8/ATmega8L, 2006 – 300 p.
[12] Rusakov V.A., Kramin A.G., 2010, Microcontroller based embedded system development problems. Proceedings of the XIX International Scientific and Technical Seminar "Modern technologies in control, automation and information processing", pp 207-208.

Victor A. Rusakov:
Ph.D. on Computers, Computer Complexes, Systems and Networks (National Research Nuclear University MEPhI, 1986)
Senior Sci Fellow on Computers, Computer Complexes, Systems and Networks (1990)
Assistant Professor on Cybernetics (1994)