IJIREEICE

ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Nitte Conference on Advances in Electrical Engineering NCAEE-2016

NMAM Institute of Technology, Nitte

Vol. 4, Special Issue 2, April 2016

Developing Platform Independent Applications

Maisie Fernandes[1], Sangeeta Mahaddalkar[2], Manikandan Kumaraguru[3]

Electrical and Electronics Department, Goa College of Engineering, Farmagudi, Ponda Goa, India[1,2]

Research & Development Department, Siemens Ltd., Verna Industrial Estate, Verna, Goa, India[3]

Abstract: Embedded Technology has been used widely for the power system protection and monitoring devices. There is also an increasing complexity added to the embedded systems. The useful life of a product, is likely to be extended, and its user base increased, if it can be migrated to various platforms over its lifetime. It is seen that much of the development time is wasted in redeveloping the application software which can be saved if the applications built are platform independent. Developing a Hardware Abstraction Layer (HAL) can ensure the much of the software reuse when ported across embedded hardware platforms having different architectures. The paper proposes the concept of HAL for two underlying Microcontroller platforms (16bit & 32bit architecture) targeting applications to power system protection. The paper discusses how a HAL is developed for the UART module of the two platforms.

Keywords: Hardware Abstraction Layer (HAL), UART, Application Program Interface (API), Middleware.

I. INTRODUCTION

Embedded technology is seen to have advanced widely in the recent years. Highly sophisticated embedded systems are employed in great numbers in the electrical power distribution process at all levels. The number of embedded system components is increasing rapidly and these components with their varied tasks cover the entire chain of the electrical power delivery process from production to consumption.

A key criterion for the characterization of an embedded system or system component is its ability to react to process events or conditions within a deterministic time frame. The main function of these systems is to protect the power system components, control the power flow, and monitor the process, as well as the condition of its equipment. Power system automation devices are integrated in communication networks for the exchange of information between several such devices, as well as with supervisory systems.

In embedded technology, software is becoming more and more important than ever before due to its high complexity in the entire system. Much of the highly detailed, difficult development is spent on interfacing the software onto the hardware. Because of the market requirements, reducing design time is one of the key issues in the design process. Software portability and reuse are effective ways to address this problem.

However, on account of a change of the hardware architecture of the underlying platform, porting application requires a lot of efforts in adjustment, modification and debugging. In this paper we see how the software portability is achieved.

The paper is organized in the following manner: Section I Introduction, Section II Architecture of an Embedded System, Section III Portable Software Architecture, Section IV The Hardware Abstraction Layer, Section V Controllers used for Implementation, Section VI Developing HAL for the UART Module and Section VII gives a brief discussion on the paper.

II. ARCHITECTURE OF AN EMBEDDED SYSTEM

An understanding of the embedded system architecture can help in embedded systems design and can also resolve the challenges faced when designing a new system.
Most common of these challenges include:

- Defining and capturing the design of the system
- Development cost limitations
- Determine system integrity like reliability and safety.

Without defining or knowing any of the implementation details, the architecture of an embedded system defines the infrastructure of a design, possible design options, and design constraints. The architecture can be used to accurately estimate and reduce costs through its demonstration of the risks involved in implementing the various elements, allowing for the mitigation of the risks. [1] Finally the various structures of architecture can then be leveraged for designing future products with similar characteristic, thus allowing design knowledge to be reused, and leading to a decrease of future design and development cost.

DOI 10.17148/IJIREEICE/NCAEE.2016.16

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Nitte Conference on Advances in Electrical Engineering NCAEE-2016
NMAM Institute of Technology, Nitte
Vol. 4, Special Issue 2, April 2016

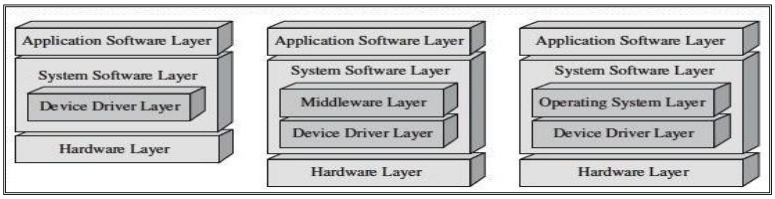

Figure1: Embedded System Architecture

In general, the embedded system architecture is divided into two main layers:

A. Hardware layer

Various hardware components need to be incorporated along with the processor to form an embedded system. Eg: peripherals such as UART, timers, ADC etc.

Major hardware components of most of the boards classified into categories as follows:

- CPU, the master processor
- Memory, where system software is stored
- Input devices, input slave processors
- Output devices, output slave processors
- Data pathways/buses, interconnects the other components

B. Software layer

The software layer is dived into the application software and the system software.

The application software layer consists of the application program and the system software supports this application. It consists of: Device drivers

- Middleware
- Operating system (OS)

The figure 1 shows the different software layer in embedded model.

- Application software layer

III. PORTABLE SOFTWARE FOR DIFFERENT ARCHITECTURE

In most MCU application, the connection between the code running on the core and the hardware peripherals is different. This makes it difficult to create a standard abstract layer that will translate between different MCU's. Adding an abstract layer will improve development processes and make it easier to reuse the code across multiple platforms. The HAL abstraction through the use of well defined HAL API's makes easier the software portability and flexibility.

Application Program Interface (API) is a set of routines, protocols, and tools for building software applications. The semantics of API specifies how software components should interact.

Figure 2 shows the software architecture to achieve platform portability.

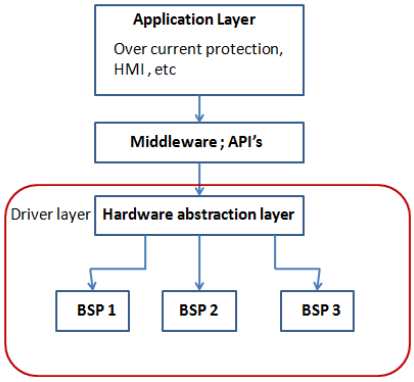

Figure 2: Software Architecture

The components are as follows:

A. Application layer:

This layer is independent of the hardware; it is managed and run by the system software. Some considerable applications for power protection systems are:

- Over current protection function

DOI 10.17148/IJIREEICE/NCAEE.2016.16

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Nitte Conference on Advances in Electrical Engineering NCAEE-2016
NMAM Institute of Technology, Nitte
Vol. 4, Special Issue 2, April 2016

- MODBUS RTU communication
- HMI interfacing

B. Middleware layer:

This layer is developed to mediate between the application layer and device driver. It is an API library that provides system abstraction to application software to increase application portability.

C. Driver layer:

This layer consists of the hardware abstraction layer (HAL) and the (board support package (BSP) for each hardware platform.

D. Hardware Abstraction Layer:

Abstraction layers enable a device driver to interact with a hardware device at a general, or abstract, level rather than at a detailed hardware level. The abstraction layer defines common routines to handle interrupts, address translations, memory read/write, and clocking functions.

E. Board support packages (BSP):

BSP is the set of software used to initialize the hardware devices on the board and implement the board specific routines. A Board Support Package (BSP) provides code and configuration items to support board-specific hardware. A typical BSP can include:

- Boot loader support
- Memory map support
- Clock system support
- Interrupt controller support
- System timer support
- Power management , etc

IV. THE HARDWARE ABSTRACTION LAYER

Conceptually the presence of HAL ensures portability of the software across different platforms. In an embedded system, there are supporting components around the core which serve different purposes. Examples include, interrupt controller, clock, timers, UART, ADC, SPI, etc. Although implemented in different ways on different hardware platforms, they basically perform the same function. For instance, hardware peripherals like UART or SPI follow a general protocol but they may have different register access methods or buffer size depending on the different Microcontroller platforms. By using HAL we can abstract the hardware component into basic functions and hide the details of the way these functions are implemented from the upper layer of software. Thus HAL interacts with the device in a more general way, providing the developer with a key advantage i.e. independence from both the device driver and the BSP. Hence, developers and customers save time and money on the integration and this enables greater software reuse.

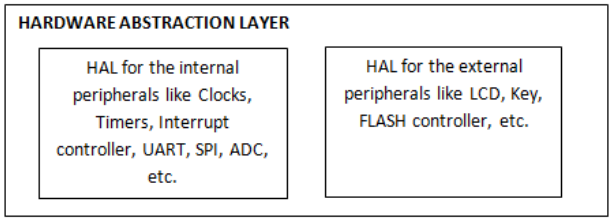

Figure 3: HAL for the internal and external peripherals

As seen in Figure 3 the HAL can be developed for the internal peripherals like Clocks, Timers, interrupt controller, UART, SPI, etc. and the external peripherals like LCD, Keys, Flash controller, etc.

V. CONTROLLERS USED FOR THE IMPLEMENTATION

The current application targets power system protection. The concept of HAL is developed and configured to target two microcontrollers from Texas Instruments one with Von Neumann architecture and another with Harvard Architecture namely,

- MSP430F5529 (MSP430F5xx family)
- MSP432P401R (MSP432P4xx family)

The concept of HAL is carried on these controllers belonging to two different architectures.

DOI 10.17148/IJIREEICE/NCAEE.2016.16

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Nitte Conference on Advances in Electrical Engineering NCAEE-2016
NMAM Institute of Technology, Nitte
Vol. 4, Special Issue 2, April 2016

A brief comparison of the two is shown below in table 1:

TABLE I
COMPARISON BETWEEN MSP432P4XX & MSP430F5XX [2] [3]

MCU architecture	Harvard	Von Neumann
MCU	MSP432P4xx	MSP430F5xx
Bus type	AHB	16-bit, msp430 bus
Instruction set	RISC, Thumb,Thumb2	RISC
Pipeline	3-stage pipeline	None
Clock Frequency	up to 48MHz	25 MHz
Flash/ROM	256KB/32kB	512KB
RAM/SRAM	64kB	66KB (RAM)
Digital I/O	84	59
DMA	8 Ch	3 Ch
UART	4 mod.	2 mod.
I2C	4 mod.	2 mod.
SPI/SSI	8 mod.	4 mod.
Gen purpose timer	4(16-bit), 2(32-bit)	4(16-bit)
USB	….	2.0

VI. DEVELOPING HAL FOR THE UART MODULE

One of the most common interfaces used in embedded systems is the universal asynchronous receiver/transmitter (UART). The MSP430 provides a module called the USCI (universal serial communications interface), and The MSP432 provides a module called the EUSCI (Enhanced Universal Serial Communication Interface).
Considering the UART application on both the microcontrollers, the HAL consists of the standard APIs and is developed for the following:

1. To access the respective device drivers of the hardware peripheral (UART module) of the two microcontroller platforms. As seen in table 1, MSP430F5xx has only two UART modules whereas the MSP432xx family has 4 UART modules. The HAL API's provide basic features for UART configuration and enabling the UART.
2. To perform I/O device configuration/access i.e. to configure the corresponding pins as UART Transmit (Output) and Receive (Input) pins.
3. For the interrupt vector management, it is seen that the MSP432xx family supports the Nested Vectored Interrupt Control (NVIC) unlike the MSP430xx series. The APIs provide common functionalities for interrupt handling (e.g. Interrupt enable/disable, Master interrupt enable/disable, etc).
4. The UART configuration requires the clock initialisation so; HAL developed for the clocks takes into account the different signals available on both the controllers and provide the required clock to run the UART module. (Functions in the HAL include clock input select, clock enable/disable, etc).

Figure 3 shows an how the HAL works taking an example of a serial communication application that requires initialization of UART and transmission of data frame.

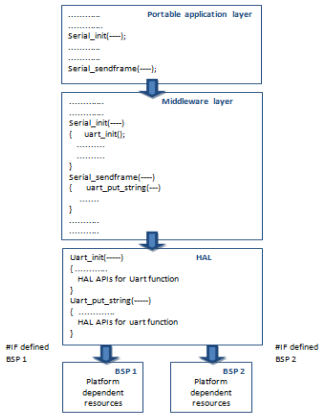

Figure 4: Illustration of the HAL for UART

IJIREEICE
ISSN (Online) 2321 – 2004
ISSN (Print) 2321 – 5526
International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering
Nitte Conference on Advances in Electrical Engineering NCAEE-2016
NMAM Institute of Technology, Nitte
Vol. 4, Special Issue 2, April 2016

The serial communication application requires a frame of data bytes to be transmitted using UART. So the top most layer; i.e. the application layer requires the peripheral used for serial communication to be initialized (UART initialization).

Different parameters based on requirement are passed on from portable application layer to the middleware layer. In the figure 3 the function serial_init( ), passes the required initialization parameters to the middleware APIs. The parameters from the middleware layer are passed on to the respective API defined in abstraction layer. The abstraction layer consists of API's specific to the peripheral devices used by the middleware layer. As seen in figure 3 the function serial_init (), further uses uart_init ( ) function that is defined in abstraction layer. Based on the underlying platform definition, the APIs in the abstraction layer calls the respective driver/BSP. The hardware dependent routines are then carried out by the BSP.

VII. DISCUSSION

1. It is seen that the abstraction layer interface forms the bridge between the controller independent application layer and the hardware dependent layer (drivers/BSP).
2. HAL can be developed for the internal peripherals like Clocks, timers, UART, ADC, etc. and the external peripherals like LCD, keys etc. and it contains the basic set of abstract features.
3. The concept can be used across different controllers from different vendors and varied architecture.
4. Developing a HAL helps in the software reuse and thus the products develop can be used across multiple platforms and thus reduce the time to market these products.

REFERENCES

[1]. Tommy Noergaard, 'Embedded system architecture a comprehensive guide for engineers and programmers.
[2]. MSP430F5xx and MSP430F6xx User's guide, Texas Instruments, revised May 2015.
[3]. MSP432P4xx User's guide, Texas Instruments, revised April 2015.
[4]. Embedded power protection: Embedded applications in power system automation, by Kornel Scherrer.
[5]. Developing Portable Software by James D. MooneyLane Department of Computer Science and Electrical Engineering, West Virginia University
[6]. "FAQ: What is a Board Support Package?" By Ron Fredericks, Wind River, August 20, 2001, updated November 20, 2002

DOI 10.17148/IJIREEICE/NCAEE.2016.16