Implementation Of Led Driver For Commercial Applications Based On Arm9

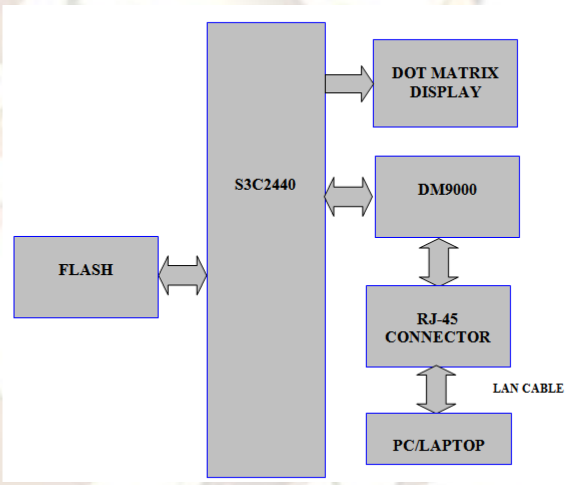
[1]G.Sravani, [2] B. Karunaiah, [3]Prof K V Murali Mohan

[1]M. Tech Student, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301.
[2]Professor, ECE, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301
[3]Professor,HOD of ECE Dept, Holy Mary Institute of Technology & Science, Bogaram (V), Keesara (M), R. R Dt.- 501301

Abstract –

In Real time development 32-bit CPU's which are widely used satisfies high speed processing, but they need a platform which makes it to run systems steadily and speedily. Stability and security plays an important role in embedded field. Based on the price, architecture and security Linux is one platform which plays an important role in embedded applications .The new version kernel has more benefits which are useful for embedded applications. Since Linux is open source we can get it easily and also the modified features in scheduling makes it more useful in embedded applications. This paper mainly concentrates on the boot-loader which is useful for system startup and loading of the system software's which defines the operating procedures of the target platform before the operating system kernel is run. Its only task is to initialize the hardware and software environment into an appropriate state.In this paper we are using S3C2440 ARM9 high performance processor. Early embedded applications are simple in order to achieve only some specific operations, such as specialized applications like industrial control. Since these are simple applications so there was no need of operating system. But as the applications complexity increased and functions became more complex, and the over all system performance also needed to be managed, so these simple applications became insufficient. And, the need to choose an embedded operating system becomes an inevitable trend. But Linux was originally developed as an Operating System for desktop computers, and the OS to be used in embedded systems will have some differences. So we need to transform Linux into Embedded Linux. Since it is the embedded operating system, we need to use Cross compiler to CROSS_COMPILE Linux source code according to the target. Before compiling we have to configure the source code according to the target board configuration. On completion of successful compilation we will get the Embedded Linux Kernel. Then we need to transfer the kernel Image to the target board.

In Embedded Linux, the application program doesn't have direct access to the devices attached to the target board. We need to use Device Drivers. So we will be making one device driver to control an LED according to our application program.



Block diagram

***Keywords* - embedded system; Boot Loader; Linux 2.6; transplant; LED driver**

I. INTRODUCTION

With the Internet's development and post-PC era, embedded systems have become the focus of the current IT industry. As embedded system is of low power consumption, small size, performance, reliability, industry oriented applications, high and prominent feature of the current, it has been widely infiltrated into science, engineering, military technology, business culture and the arts, entertainment, as well as people's every aspect of daily life and so on. We can imagine that some people may have never had any contact with computers, but cannot imagine that he never had any contact with embedded systems. Because embedded systems are everywhere, from the family's washing machines, refrigerators, bicycles, cars, to the office of the remote conferencing system, etc., this is part of embedded products. Currently embedded technology has become a hot research and application. Early embedded applications is relatively simple, generally only in order to achieve some specific functions to meet the needs of a specific occasion, such as used in highly specialized types of industrial control, aircraft and missiles and other weapons and equipment in, etc. In these systems, running software is a simple control loop, so in general do not use the operating system, but as the application areas of embedded systems has expanded its functions more complex, so entirely by the programmer to manage the overall system capacity is clearly insufficient, And for each additional one feature is necessary to redesign system, resulting in a tremendous waste of resources and duplication of effort. Meanwhile, along with the development of computer technology and integrated circuits, hardware provided conditions are getting better and better, so choose an embedded operating system has become an inevitable trend. Embedded systems embedded operating system is to support the work of the operating system. It is the nature of knowledge systems and technology with general-purpose operating system, there is not much difference, generally used for more complex embedded system software development [1]. Be able to effectively manage a complex system resources to complete the process management, processor scheduling, memory management, device management, interrupt handling tasks such as operating systems; be able to hardware virtualization, allowing developers to the driver from the busy port and maintenance freed; able to provide library functions, drivers, tools and applications. Embedded operating system is important to the operation of an embedded system, environment and development platform, whether it is efficient, stable, secure and so will have a direct bearing on the success or failure of embedded systems has become an embedded system design and development priorities. But Linux was originally developed for desktop machines, which is used in embedded systems have some differences, such as memory capacity and limited compared to desktop computers, so how to transform Linux into a small capacity, high stability and easy the development of embedded operating system becomes a critical issue. This also means that the embedded Linux operating system, used in digital products and industrial control fields of a lot of work needs to be done

II. INTRODUCTION TO S3C2440

Embedded operating system and drivers transplant, need to be transplanted to the target hardware platform is more in-depth understanding. This optional use of Samsung's embedded development platform based on ARM9 core S3C2440 microprocessors, is designed for PDA, Internet equipment, and handheld devices such as the development of high-performance, low-power microprocessor. This chapter analyzes the main characteristics of S3C2440 microprocessor, the system's overall structure and characteristics, and a brief introduction of the system of storage space allocation.

Introduction 2.1.1 S3C2440 Processor

Samsung’s S3C2440A is designed to provide hand-held devices and general applications with low-power, and high-performance microcontroller solution in small die size. To reduce total system cost, the S3C2440A includes the following components. The S3C2440A is developed with ARM920T core, 0.13um CMOS standard cells and a memory complier. Its low power, simple, elegant and fully static design is particularly suitable for cost- and power-sensitive applications. It adopts a new bus architecture known as Advanced Micro controller Bus Architecture (AMBA).The S3C2440A offers outstanding features with its CPU core, a 16/32-bit ARM920T RISC processor designed by Advanced RISC Machines, Ltd.

The ARM920T implements MMU, AMBA BUS, and Harvard cache architecture with separate 16KB instruction and 16KB data caches, each with an 8-word line length. By providing a complete set of common system peripherals, the S3C2440A minimizes overall system costs and eliminates the need to configure additional components. The integrated on-chip functions that are described in this document include:

- Around 1.2V internal, 1.8V/2.5V/3.3V memory, 3.3V external I/O microprocessor with 16KB I-Cache/16KB DCache/
- MMU
- External memory controller (SDRAM Control and Chip Select logic)
- LCD controller (up to 4K color STN and 256K color TFT) with LCD-dedicated DMA
- 4-ch DMA controllers with external request pins
- 3-ch UART’s (IrDA1.0, 64-Byte Tx FIFO, and 64-Byte Rx FIFO)
- 2-ch SPI’s
- IIC bus interface (multi-master support)
- IIS Audio CODEC interface
- AC’97 CODEC interface
- SD Host interface version 1.0 & MMC Protocol version 2.11 compatible
- 2-ch USB Host controller / 1-ch USB Device controller (ver 1.1)
- 4-ch PWM timers / 1-ch Internal timer / Watch Dog Timer

- 8-ch 10-bit ADC and Touch screen interface
- RTC with calendar function
- Camera interface (Max. 4096 x 4096 pixels input support. 2048 x 2048 pixel input support for scaling)
- 130 General Purpose I/O ports / 24-ch external interrupt source
- Power control: Normal, Slow, Idle and Sleep mode
- On-chip clock generator with PLL

III. EMBEDDED SYSTEM BOOT PROCESS AND BOOTLOADER TRANSPLANT

This chapter is rooted in the embedded development board of the Boot Loader systematic theoretical explanation.

Boot loader to run after the system power-on the first paragraph of the code. PC, the boot loader is located in the hard disk from the BIOS and the MBR in the OS Boot Loader component. BIOS Upon completion of the hardware detection and resource allocation will be the hard disk MBR in the Boot Loader to read the system RAM, and then jump to kernel entry point to the operation, which started operating. In embedded systems, generally do not like the BIOS firmware like that, so start the task of loading the entire system is entirely completed by the Boot Loader.

Boot Loader is used to complete the system startup and system software loads the operating procedures of the target platform stored in the nonvolatile storage medium, before the operating system kernel to run, its task is to initialize the hardware device through the establishment of memory space maps, the system's hardware and software environment into an appropriate state, for the final call to the operating system kernel to prepare the right environment, then, Boot Loader will have served its purpose and its life cycle also stop there. Can be seen, it is the underlying hardware and the upper application software, a middleware between the software, the completion of processors and peripheral circuits to be running the initialization work; can shield the underlying hardware differences, so that the upper application software easier to write and transplantation ; not only have similar PC-BIOS function, but also has some debugging features.

Most of Boot Loader contains two different modes of operation.

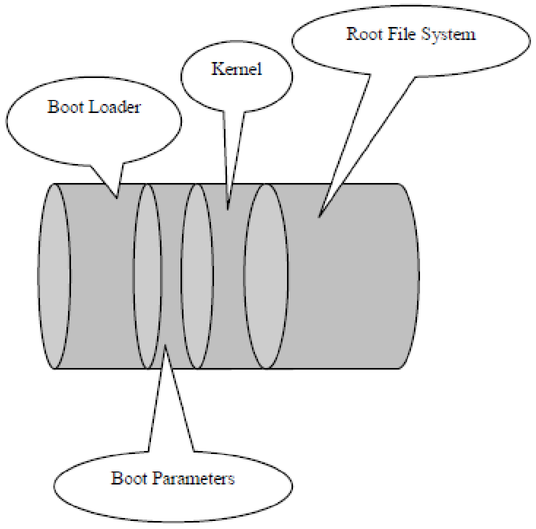


Figure 1. Typical solid-state storage device structure of spatial distribution

(1) Boot loader (Boot Loading) model: namely, autonomy (Autonomous) mode. Boot Loader from the target machine to solid-state storage devices will be loaded into RAM to run the operating system, the entire process and no user intervention. The Boot Loader mode is the normal operating mode, so when the embedded product releases, Boot Loader obviously have to work in that mode.

(2) Download (Down Loading) mode: Boot Loader on the target machine through the serial port connection or network connection communication means such as downloading files from the host. Downloaded file from the host is usually the preferred Boot Loader has been saved to the target machine's RAM, and then Boot Loader was written to FLASH on the target machine class solid-state storage device. This mode is usually the first to install the kernel with the ROOT FS was used; In addition, the future system update will also use this mode. Working in this mode, the Boot Loader to its end users usually provides a simple command interface[6].

After the system power-on or reset, CPU is usually from a pre-arranged by the manufacturer's address on the instruction fetch. CPU-based embedded systems usually have built a solid-state storage devices (such as: ROM, E2PROM, or FLASH, etc.) are mapped to this prearranged address. Thus, in the system after power, CPU will be the preferred implementation of the Boot Loader program. Shown in Figure 1 shall be also equipped with a Boot Loader, the kernel boot parameters, the kernel image and ROOT FS image of a typical solid-state storage device space allocation chart[7].

IV. SOFTWARE PLATFORM

Due to the tireless efforts of a large number of software elite, Linux in embedded systems applications more and more widely. 2.6 kernel in a large number of new features to make them more suitable for embedded systems. It is based on tried and innovative considerations, the system uses a 2.6.11 kernel, as well as YAFFS file system. The system's hardware circuit is based on SMDK2440, and its related knowledge points, have been the other detail. The following introduces the software environment migration tasks.

(1) Access to Linux kernel source file and its corresponding patch Linux kernel's official website is www.kemel.org. Any change on the kernel to update this site has been subject to.

(2) To obtain the source code YAFFS http://husaberptoby-churchill.com/h can obtain the necessary source code.

(3) Start transplantation Extract the source files, and marked with the corresponding patch. We assume that all documents are saved in the

```
/home/yourmame directory
#cd /home/yourmame
#tar-zxvf linux-2.6.11.12.tar.gz
#cp -a ./linux-2.6.11.12 /usr/src
#cp patch-2.6.11.12.tar.gz /usr/src/
#mkdir /usr/local/arm/
#tar -jxvf arm-linux-gcc-3A1.tar. 6z2
#cp ./3.4.1 /usr/local/ann/
#tar-zxvf yaffs.tar.gz
#cp -a ./yaffs fusr/src/linux-2.6.11.12/fslyaffs
#cd /usr/sre/linux-2.6.11.12/
#zcat../patch-2.6.11.12.tar.gz!patch -pl -f
```

V. LED MATRIX DRIVER DEVELOPMENT

A. OverviewLinux Driver

After determining the basic functions of the kernel, the remaining work is to develop the user's specific device driver. With the support of device driver, hardware devices can work properly. The driver actually controls the I / O devices, which contains a series of functions that are used to control the peripheral. Linux will be the right function of the operation of equipment in the form of a unified interface to the file expression. Therefore, in Linux, all of the devices are files; the file can be achieved through the operation of the device control. Such as the application requires the use of equipment, with the system call open ()
opens a device file, and build up the target device connection. For the implementation of the application process is concerned, to establish a connection on the performance of an already open file. We can use after VFS Virtual File System calls related to the device handler in. A typical driver has the following features:

(1) By a series of functions and data structure, it has to communicate with hardware devices ,at the same time it has to follow the Unified interface provided by operating system kernel.

(2) Self-contained components that can be dynamically added to the operating system kernel or removed by the kernel.

(3) It is to manage the user program and data flow between peripherals and control flow.

(4) Belongs to the kernel part customized through the device file to deal with the user program.

B. LED Driver Development

(1) Character device driver based on the right LED control. Showed that some use is a 5X light-emitting diode dot-matrix, we are frequently asked for issemination of information, display of Chinese characters dot matrix LED display usually has a number of blocks of LED dot matrix display modules, 8X8 dot matrix display modules, each 64 independent of the light-emitting diodes, in order to reduce the pin and easy to package a variety of LED matrix display module have adopted the form of an array row cloth, that the ranks of the intersection of the line wherever there display LED. Thus, LED dot matrix display module of the display driver can only use dynamic driving method, each line can only lit LED (of the form yang dot matrix LED display module), or an LED (total Yin Type LED dot matrix display module).

(2) I/O Interface

In this development board, the entire LED display module type as an I/O for control. DATA [0 ... 7], DATA [8 ... 12] system data lines corresponding to the low 16-bit, LED_LOCK signal from the system bus write signal and address signals derived by the combination of simple logic in the CPLD board to complete , control of the display module's I / O address 0x20000000.

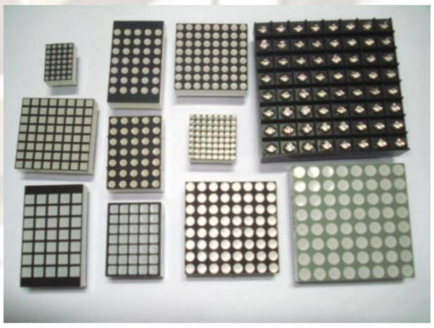
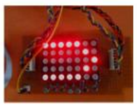
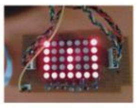
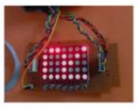


Figure 3 – LED Module

VI RESULTS



VII CONCLUSION

Embedded technology development is inseparable with operating systems, computer architecture, VLSI and other basic subjects, at the same time, embedded systems development provide a broader development platform for communications, navigation and other applied sciences.

The design selected GX-ARM9experimental system of Beijing's innovative company as the hardware platform, through the initial design, post-commissioning, transplant the embedded system Boot Loader based on S3C2440and the operating system Linux 2.6 kernel, as well as LED dot-matrix driver development.

As the new 2.6 kernel has begun to make new improvements in the new driver programme, we can used new functions, in order to achieve the flexibility of the module.

REFERENCES

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

[2] Song, Yanzhao. Embedded operating system resentation and the principle of selection. .2005,23 Industrial control computer (5): 41- 43

[3] Zhong Xichang. Embedded operating system in China's development status and pre-A Information Technology and Standardization .2006,7 (6): 6-10.

[4] Yang Rong, Wang Lin Dou, information appliances based on embedded Linux operating system, analysis and application. East- 101 [5] Jiao Quan, Huang Rural Health, Bao-jun. U-Boot .Porting the S3C2410. Electronic design applications, 2006,3:126-128

[6] Zhang Jin, JIANG Wei. U-Boot boot process analysis And transplant step. Electric Power Automation Equipment, 2005,25 (7): 68-71

[7] Bill Weinberg. Embedded Linux Is A Hit In Wireless Entertainment[J] Wireless System Design,]an 2003.29-32.

G.Sravani Completed B.Tech in Electronics and communication Engineering from St.Ann's College of Engineering and Technology,Chirala,JNTU Kakinada and pursuing M.Tech in Embedded systems from Holy Mary institute of Technology and science, JNTU Hyderabad. My interested areas are Embedded systems

Karaunaiah B is pursuing PhD from Andhra University, Visakhapatnam, Andhra Pradesh in the field of Antennas.Completed M. Tech in ECE with specialization Electronic Instrumentation **from Andhra** University, Visakhapatnam in 2003.

B. Tech in Electronics and Communication Engineering from Bapatla Engineering college, Nagarjuna University, Guntur, Andhra Pradesh in 1998.Currently working as Professor in the ECE Department at Holy Mary Institute of Technology **and** Science (College of Engineering), Hyderabad. Areas of interest **are** optical communication, microwave communication, radar engineering, . antenna & wave propagation, fundamentals of electronic devices, basic electronics, EM field theory, satellite and digital communication.