TELKOMNIKA, Vol. 11, No. 9, September 2013, pp. 4918~4924
ISSN: 2302-4046

The monochrome LCD Driver Design and Implementation based on Framebuffer

Li Zhi-hui*, Xian Jinlong
College of Information Science and Engineering, Henan University of Technology, Zhengzhou, 450001, China
* Corresponding author, e-mail: duzh@tsinghua.edu.cn

Abstract

Recently, the LCD applications have developed very rapidly, especially the application of the monochrome LCD in certain areas. This paper, on hardware, introduced the connection between monochrome LCD HEM160160-07C2 and ARM9-based AT91SAM9261; on software, respectively introduced monochrome LCD driver implementation based on GPIO and based on Framebuffer. By the two mode analysis, the mode based on Framebuffer is better, ultimately has been validated in the project experiment.

Keywords*: ARM9, linux, framebuffer, monochrome LCD*

1. Introduction

In recent years, the LCD because of its advantages of low operating voltage, low power consumption, ease of integration, long life, portability, large amount of display information and electromagnetic radiation pollution, has increased rapidly in the development [1]. According to the classification of the color, LCD can be divided into two kinds of monochrome screen and a color screen. Although the application of the color screen is more and more widely, but the pixel requirements are generally not high in some certain areas, such as pure numbers, pure characters, pure text and icon. In addition, most of high-resolution monochrome dot-matrix LCD has its own controller [2]. Taking the power meter reading for example, the LCD is need to adapt to harsh environments and maintain a longer operating time, and need low power consumption, high contrast, so monochrome LCD is necessary. Generally, ARM9 comes with LCD controller, but the controller does not support the controller in monochrome screen. So need to write the appropriate drivers according to the requirements of the specific controller, which increases the difficulty of system development in some extent. This article describes how to use the ARM9 processor comes with its own LCD controller to realize the monochrome screen driver under Linux. In the experiment, the model of ARM9 is AT91SAM9261, and the model of monochrome screen is HEM160160-07C2.

2. Design of the Interface between LCD and ARM

HEM160160-07C2 is a high contrast black and white FSTN LCD screen, widely used in environmental protection, medical, power and military industries. It supports 3.3V logic supply, parallel communication, display points 160*160, and operating temperature range -25 □ ~ 75 □. The LCD controller in HEM160160-07C2 is uc1698 which is a color LCD driver supporting color panel160 * 128, 160 COM and 384 SEG. So the RGB median and order can be set according to that. A pixel consists of a COM and three SEG corresponding to the R, G, B. In other words,, the written values of RGB will be reflected in the three SEG interface. In the hardware connection, a pixel is only connected to a SEG and COM. For the written data in the software, uc1698 still considered to be "R G B", respectively reflected the three SEG, and to achieve the control three pixels. HEM160160-07C2 total gives twenty signal lines (three useless), and its external interface signals are described as shown in Table 1.

Received April 4, 2013; Revised May 29, 2013; Accepted June 12, 2013

Table 1. HEM160160-07C2 External Interface Signal Description

Pin No.	Symbol	Function
1	VSS	Ground
2	AO	Command and Data register select signal
3	WR	In 8080 mode, use as Write enable input. “L” active
4	RD	In 8080 mode, use as Read enable input. “L” active
5	CS	The chip select signal. “L” active
6	RST	Reset input pin
7	VDD	Support voltage for logic circuit +3.3V
8-15	D0~D7	Display data signal
16	BLK	Support voltage for backlight Ground 0V
18	BLA	Support voltage for backlight +3.3V
17, 19, 20	NC	…

The internal LCD controller in AT91SAM9261 is used to control the transmission of image data, consists of two main blocks, the DMA controller and LCD controller core (LCDC core), the DMA controller reads the display data from an external memory through the AHB master interface. The LCD controller core formats the display data. The LCD controller core controller core continuously pumps the pixel data into through the LCD bus (LCDD [23:0]). This bus is timed by the LCDDOYCK, LCDDEN, LCDHSYNC, and LCDVSYNC signals. The LCD controller interface signals consist of 24 data lines and 5 control signal line shown in Table 2. By programming the LCD ,a variety of different needs can be achieved, such as resolution up to 2048*2048; pixel depth (1,2,4,8,16,24 bits per pixel.); Data line width (4,8, 16 or 24) and interface Timing.

Table 2. LCD Controller External Interface Signals and Description

Name	Description	Type
LCDCC	Contrast control signal	Output
LCDHSYNC	Horizontal synchronous signal	Output
LCDDOTCK	LCD clock signal	Output
LCDVSTNC	Vertical synchronous signal	Output
LCDDEN	Data enable signal	Output
LCDD[23:0]	LCD Data Bus output	Output

The AT91SAM9261 comes with a LCD controller, but the controller does not support the uc1698 controller in HEM160160-07C2. Therefore directly using the internal LCD controller in AT91SAM9261 can not be to drive HEM160160-07C2. The method used here is to use a combination of the LCD controller and GPIO to drive the LCD dynamic display.

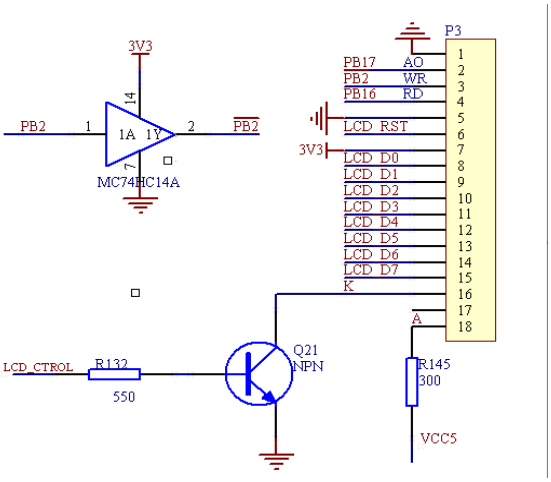

Figure 1. LCD External Interface Circuit

The external interface logic level of HEM160160-07C2 is 3.3V, compatible with the ARM, thus can be directly connected to the generic GPIO of AT91SAM9261. The original LCDDOYCK LCDDEN, LCDHSYNC, and LCDVSYNC signal port in AT91SAM9261, used to control the color LCD, can complete the initialization of the LCD through configuring internal register which is set to the generic GPIO port. The 8-bit data interface of HEM160160-07C2 can be connected to low eight bits of data bus of AT91SAM9261 internal LCD controller by configuring the data line width for 8-bit to realize parallel data transmission. Because of only write operation on the LCD operation, the write enable port of HEM160160-07C2 can connect to the LCDDOYCK signal port to achieve write clock synchronization. Due to both send and receive waveforms triggered in different ways, a non-gate is need in the middle. The other external interfaces can be controlled by other generic GPIO. The specific connection is shown in Figure 1.

3. Development of Monochrome LCD Driver under Linux

The driver is a class of procedures to drive the hardware work properly, and written for specific hardware. The device driver under Linux for the hardware provides a standardized interface to the user program, and hides the details of the work of the equipment [3]. Linux system equipment is divided into three types: character devices, block devices and network equipment. The LCD device driver involved in this paper is a character device driver. The character device drivers can be completed as follows:

(1) Completion of the loading function of the character device driver. Using the function register_chrdev () to achieve the applications and device registration of main and minor device number.

(2) Fill member functions of the structure file_operations in the character device driver. The member functions in the structure is the interface between character device drivers and kernel, which define a variety of operation of equipment, such as the open operation--open (), write operation--write (), read operation--read (), close operation--close ().

(3) Completion uninstall of the character device driver. Using the function unregister_chrdev() to uninstall the character device driver.

3.1. Implemented of Monochrome LCD Driver based on GPIO

The simple design of LCD driver without using the internal LCD controller in AT91SAM9261, is to directly read and write the GPIO. The driver based GPIO provides two types of interfaces to the LCD: Data signal interface; Function signal interface. For the internal uc1698 controller of HEM160160-07C2, the data signal interface is DB0 ~ DB7, and the function signal interface is the write enable signal (WR), read enable signal (RD), channel selection signal (AO) and chip select signal (CS). There are two general methods to achieve LCD driver based on the GPIO: One is that the read and write timing is completed in the application, while GPIO driver is responsible for the writing control and data register. The other is to control the read and write timing in the GPIO driver. Taking the second for example, LCD driver based on GPIO includes four following functions: write data function--lcd_WRData(), write command function--lcd_WRCommand(), display function--lcd_Display(), initialization function--lcd_Init().The completion lcd_WRData() and lcd_WRCommand() is based on the timing of uc1698 controller. The completion of lcd_Init() and lcd_Display() is based on calling lcd_WRData() and lcd_WRCommand(). Among them lcd_WRCommand() function is implemented as follows:

```
void lcd_WRCommand (unsigned int para)
{
    CSLow;   // CS enable
    AOLow;   // Select command register
    RDHigh;   // Read disable
    WRHigh;   // Pulled Write Enable level
    AT91C_BASE_PIOB->PIO_ODSR = (para<<5);
                    // 8 bits parallel data output (PB5-PB12)
    WRHigh;   // Down the write enable level to send data
     CSHigh;    // CS disable
}
```

lcd_WRData () is similar to lcd_WRCommand ().Application by calling the member function ioctl () in the structure file_operations, and ultimately by lcd_init () and lcd_Display () to complete the LCD display. The program structure is shown in Figure 2:

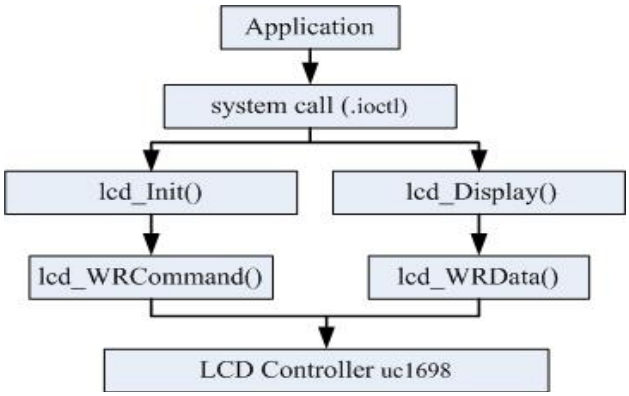

Figure 2. The Program Structure based on GPIO

However, in practical applications, because of the disadvantage of poor versatility, slow speed, and low execution efficiency, LCD driver based on GPIO is not suitable for multi-tasking, multi-threaded concurrent operations. For this reason, LCD controller in AT91SAM9261 can be used to achieve the LCD driver. And Linux has specifically provided a frame buffer (Framebuffer) for the LCD display devices.

3.2. Implemented of Monochrome LCD Driver based on Framebuffer

The Framebuffer provided by Linux is an interface to the display device, a device of abstracting video memory, thus shields the underlying differences in graphics hardware [5]. Therefore, the writing of driving program can be achieved to directly operate the display buffer to complete the location and storage of physical display buffer. In other words, the substance of writing the LCD driver is to write frame buffer driver.

The frame buffer device is belongs to the character device, so the driver uses a “file layer-driver layer” structure. At the file level, it provides the perfect operation interface of device file to the application of the above, such as read(), write(), mmap(), and ioctl() operation. The mmap () is the most crucial part to achieve frame buffer device function. It can directly map the video memory to the user process address space. The structure fb_fops is initialized as follows:

```
static struct file_operations fb_fops={
   .owner = THIS_MODULE,
   .read = fb_read, // read operation
   .write = fb_write, // write operation
   .ioctl = fb_ioctl, //control operation
   .mmap = fb_mmap, // mapping operation
   .open = fb_open, // open operation
   .release = fb_release, // close operation
};
```

In the driver layer, a very important structure is fb_info, which is the driver interface for frame buffer device defined by of the Linux, contains a complete description of attributes and operations of the frame buffer device. It not only contains the underlying operating functions, but also contains some data which record device state. The fb_info is an interface left by lower device to upper layer, including all of the information of the frame buffer device.

The structure is to complete the registration and cancellation to kernel through a pair of functions provided by the file layer. The functions are shown as follows:

*int register_framebuffer(struct fb_info *fb_info)*

*int unregister_framebuffer(struct fb_info *fb_info)*

The structure fb_info is defined as follows:

```
struct fb_info{
    int node;
    int flags;
    struct fb_var_screeninfo var;
    struct fb_fix_screeninfo fix;
    struct fb_monspecs monspecs;
    struct work_struct queue;
    struct fb_pixmap pixmap;
    struct list_head modelist;
    struct fb_ops *fbops;
    ……
    };
```

The member fb_var_screeninfo records the display control information which user can modify, such as the pixels of each row and column. The member fb_fix_screeninfo records controller parameters which users can not modify, such as the physical address and length of the buffer. When the frame buffer device do mapping operation, means that, gets physical address of the buffer from the structure. In the structure fb_info, the most important nested structure is fb_ops which is a function pointer pointing to the underlying operation. When the application process call ioctl () to get or set the display parameters, in fact, is to call the functions contained in fb_ops. The structure fb_ops is initialized as follows:

```
static struct fb_ops atmel_lcdfb_ops = {
    .owner          = THIS_MODULE,
    .fb_check_var   = atmel_lcdfb_check_var, / / Check the LCD parameters
    .fb_set_par     = atmel_lcdfb_set_par, / / Set the display mode
    .fb_setcolreg   = atmel_lcdfb_setcolreg, / / Set register corresponding to display mode
    .fb_pan_display = atmel_lcdfb_pan_display,
    .fb_fillrect    = cfb_fillrect, / / Draw a rectangular area
    .fb_copyarea    = cfb_copyarea, / / Copy display data to another area
    .fb_imageblit   = cfb_imageblit, / / Display picture
};
```

The interface of the LCD controller in AT91SAM9261 is not compatible with the external interface of HEM160160-07C2. The HEM160160-07C2 does not support LCDDEN, LCDHSYNC and LCDVSYNC signal port used for color LCD in AT91SAM9261. Therefore, directly using the LCD controller in AT91SAM9261 can not complete the monochrome LCD Driver based on Framebuffer. Through the above analysis, combining the both, implement of the monochrome LCD Driver based on Framebuffer is possible. Specifically, after seeding a full screen data, directly using the way based on GPIO to Complete the LCD initial position. Then the frame buffer provided by Framebuffer maps video memory to the user space, and the underlying driver reads data directly from the frame buffer using the LCD controller in AT91SAM9261. In this way, the monochrome LCD Driver based on Framebuffer can be achieved.

The main work on the program [6-8]:

(1) Initialize the LCD screen and the LCD controller. First setting the PIO disable register PIO_PDR = 0, the PIO enable register AT91C_BASE_PIOB-> PIO_PER = 0x00001FF7, the peripheral select register AT91C_BASE_PIOB-of> PIO_ASR = 0, makes the LCD Controller Interface work under normal GPIO mode. In this mode, using write command operations to initialize the LCD. Then to initialize the LCD controller, and set LCD control register according to LCD parameter.

(2) Write the member functions. Mainly to write the underlying operating functions in fb_ops(). Must be noted that HEM160160-07C2 is monochrome LCD, and its external data interface is 8 bits. But its own drive uc1698 is Color LCD drive, can support 160*128 color panel, 16-bit three Display: R: G: B = 5:6:5. So, the LCD display mode can be set to be single color, single scan and 8-bit interface width.

(3) Fill member function in structure file_operations. Read / write (.read /.write) operation, which is equal to read / write buffer. Here mainly to write the buffer. Input / output control (.ioctl) operation can be read or set the parameters of the display device, such as display mode, the data width of the bus, the screen size.

(4) Mapping (.mmap) function. Kernel space and user space of Linux, usually work in protected mode. Each application process has its own virtual address space, can not directly

access the physical buffer address. Linux provides the address mapping operation in the file operations interface to map file to user space. So that users can access the buffer by reading and writing this address, then draw on the screen.

(5) DMA interrupt handling. Because HEM160160-07C2 does not have LCDHSYNC, LCDVSTNC interface to synchronize horizontal and vertical with the LCD controller in AT91SAM9261, LCD need to reposition when finished the first screen data. Therefore, after the DMA sent a screen data, a frame end interrupt will produce. At this time, the program under the normal GPIO mode completes the reposition of the LCD. Its program structure shown in Figure 3:

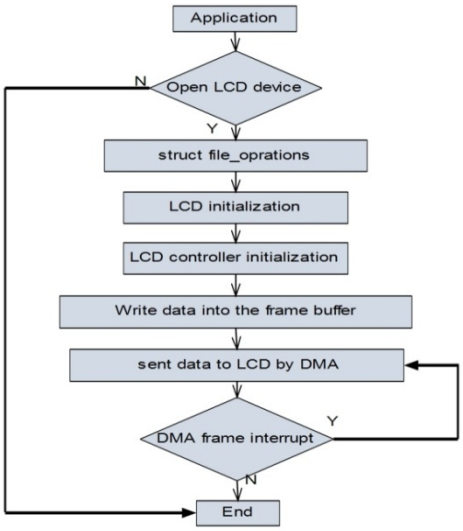

Figure 3. Monochrome LCD Program Structure based on Framebuffer

4. Experimental Results

The monochrome LCD Driver based on Framebuffer mentioned in this article have been applied in the project experiment of meter reading system. After finished LCD driver, use the command “insmod” to load it into the kernel. By writing a simple test application, achieve device open, device close, memory map and input and output control operation, ultimately complete monochrome LCD normal display.

Figure 4. Monochrome LCD display in the experiment

References

[1] Wen Xianguang. Design of Low Power and High Performance LCD Driver. Zhejiang University. 2005: 1- 5.
[2] Wang Yanpin, Feng Bingjun, Yuan Guoshun. A LCD Driver Circuit with Key Input Function Design. *Microelectronics & Computer*. 2009; 26(1): 34-37.
[3] RUBINI A, CORBET J. *Linux Device Drivers*. Oreilly & Associates Inc. 2005: 9-12.
[4] Zhang Jiawei, Zhou Andong, Luo Yong. Design of LCD Driving in Linux Based on ARM11 Embedded System. *Chinese Journal of Liquid Crystals and Displays*. 2011; 26(5): 660-664.
[5] Chang Yunjie Zhang Weiyong. The design of LCD divers based on Framebuffer. *Office Informatization*. 2008; (24): 30-31.
[6] Chen Tao, Yu Xuecai, Zhu Liangxiao, et al. Design and Implementation of the LCD Driver in the Embedded Linux. *Electronic Science and Technology*. 2010; 23(11): 22-24.
[7] Chen Chaoying, Huang Weizhi. *Design of LCD Driver Base d on S3C2440A and Linux*. Proceedings of 2010 Asia-Pacific Conference on Information Theory (APCIT 2010). 2010.
[8] Zhu Baohua. Development explain of Linux device driver. Posts and Telecom Press. 2008.