Development of BSP under WinCE and Its Application in Inkjet Printer

Hongwen Wang, Panyun Lei, Ning Le and Zhangliang Guo
School of Control Science and Engineering, Hebei University of Technology, Tianjin 300130, China

***Abstract*—Based on the powerful data processing capability of S3C2440 core board and the good performance of WinCE operating system in the capability that the system can be transplanted, cut and has a good real-time performance, a control system of high resolution inkjet printer has been designed. With analyzing the hardware platform of high resolution inkjet printer, this paper has mainly studied the transplant of BSP and the customization of inkjet printer operating system under WinCE. And the framework and development process of BootLoader, OAL, device driver, and configuration file structure has been also discussed. The principle of code reuse is used to simplify the development of BootLoader, OAL. At the same time, a driver of nozzle has been developed. Experiments show that self-customized WinCE operating system can run stably on the hardware platform of inkjet printer, and makes the inkjet printer realize the printing function.**

Keywords-high resolution inkjet printer; windows CE 5.0; BSP; XJ128 nozzle driver

I. Introduction

The technology of inkjet printing presents the trend of high speed printing, low maintenance rate and humanized operation. While the domestic inkjet printer is mainly constituted by low speed controller such like MCU, which cannot satisfy the current social demand [1]. Therefore, to develop an inkjet printer operating system which is efficient, quick and simple is imperative. The application of embedded system in printing industry provides a software support for the development of domestic inkjet printer.

II. Hardware System Design of Inkjet Printer

A. General Design of Hardware Circuit

The core of inkjet printer hardware platform designed in this paper is S3C2440 processor, and the processor is responsible for controlling the operation of the whole system. The hardware structure of inkjet printer control system is shown in Figure 1.

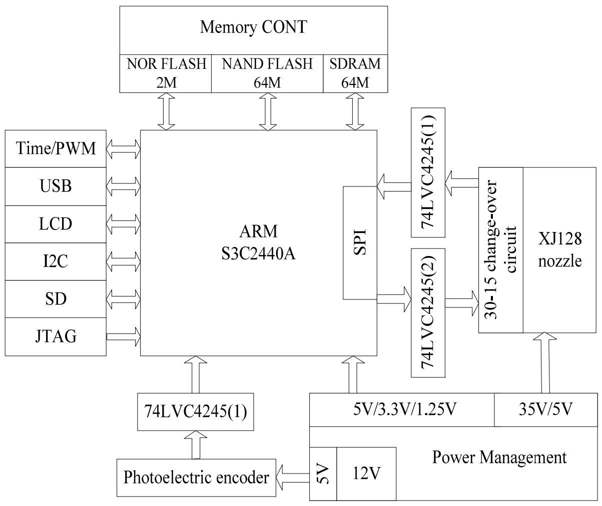


FIGURE I. HARDWARE STRUCTURE OF THE SYSTEM

B. Working Principle of Inkjet Printer

Taking an object on the spray line which is to be printed as an example. The object moves through the transmission device, and the photoelectric encoder which is coaxial of the motor of the device monitors the direction and speed of the spray line. Then the monitoring result will be feedback to the control system, which constitutes a closed-loop control system to ensure the stable operation of the conveyor belt. When the object is to be sprayed, laser sensor triggers spray printing and records the quantity of the printing products, and nozzle exchanges the printing data with control system through the SPI bus and achieves the printing function with the system clock.

C. Nozzle Interface Circuit

This paper has chosen XJ128 which is an on demand inkjet nozzle produced by British Purcell Company (XAAR). High resolution inkjet printer will transmit the data flow which the application has already handled to the nozzle through the serial peripheral interface SPI, and finally the nozzle prints out the corresponding information [2].

The hardware connection diagram of S3C2440 and XJ128 is shown in Figure 2.

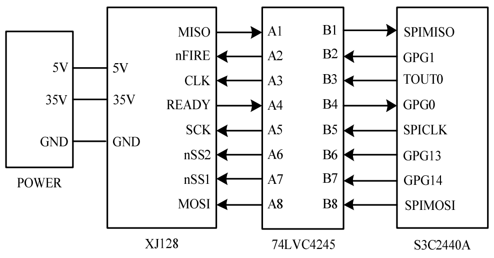


FIGURE II. HARDWARE CONNECTION DIAGRAM OF S3C2440 AND XJ128

III. BSP DEVELOPMENT

BSP, also known as board support package, is a software code that is used as an abstract interface between the operating system and the target development board. BSP mainly contains four parts: BootLoader, OAL, driver and configuration files. The development of BSP is a complex process, The specific work in this paper is introduced as follows.

A. *Transplantation of BootLoader*

BootLoader is a small program running before the kernel of the operating system. It can initialize the hardware device, establish mapping graph of memory space, in order to bring the hardware and software environment of system to a suitable state. Then BootLoader loads the image of operating system to memory so that the operating system can implement independently [3]. The function call sequence diagram of BootLoader is shown in Figure 3.

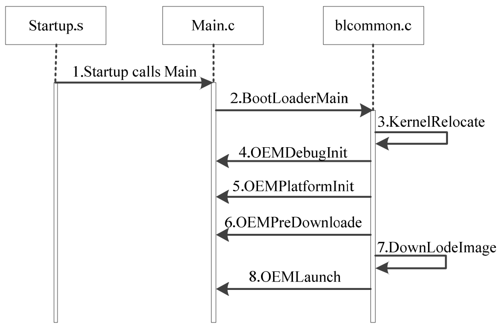


FIGURE III. FUNCTION CALL SEQUENCE DIAGRAM OF BOOTLOADER

The analysis of BootLoader and the specific code to implement are as follows.

(1) Startup()(Startup.s): This is an entry function of BootLoader, written in assembly language. This function completes the most basic initialization of the platform and copies the code to the RAM. Then BootLoader jumps to the main() function and system transfers into the C language environment. The following is part of the key code of Startup() function:

```
ResetHandler
......
; Set INTMSK, INTSUBMSK, disable all interrupt.
; Fin=12MHz, Configure MPLL=400MHz,
UPLL=48MHz.
……
; Copy boot loader to memory
b main
```

(2) BootloaderMain()(Blcommon.c): This function is the main control function and it controls the whole implementation process of Bootloader.

(3) KernelRelocate()(BLcommon.c): This function repositions the global variable of Bootloader in RAM.

(4) OEMDebugInit()(main.c): This function is used as an initialization of debug port. According to the characteristics of S3C2440 processor chip [4] and the hardware pin diagram of inkjet printer, UART0 is selected as the hardware port for debugging output and GPH2 port, GPH3 port are selected as TxD pin and RxD pin. The specific initialization and configuration are implemented in OEMInitDebugSerial(). Code as follows:

```
// GPH2 and GHP3 are UART0 TxD and RxD, respectively.
pIOPortReg->GPHCON&=~((3<<4)|(3<<6));
pIOPortReg->GPHCON|=((2<<4)|(2<<6));
pIOPortReg->GPHUP|=((1<<2)|(1<<3));
// Configure the UART.
ULCON = 0x03;
UCON = 0x05;
UFCON = 0x00;
UMCON = 0x00;
UBRDIV=(int)(S3C2440A_PCLK/(115200*16) -1);
```

(5) OEMPlatformInit()(main.c): This function is used to initialize the equipment on target board, including display screen, Flash, network card, real-time clock and so on.

(6) OEMPreDownload()(main.c): Before d-ownloading t he image, this function is calledto complete the preparatio n work.

(7) DownloadImage()(main.c): This function downloads the image of operating system to the hardware platform.

(8) OEMLaunch()(main.c): This function completes the operation that makes the start-up process jump to the operating system image to perform.

B. *Implementation of OAL*

The OAL code in WinCE5.0 is mainly divided into four parts: the system structure code, SoC chip code, board code and hardware independent code [5]. The hardware platform of inkjet printer mainly extends the peripheral hardware of development board. So the main job is to complete the modification of the board level OAL code. This paper uses the code reuse idea to reuse a lot of code in BootLoader to reduce the burden of OAL development. The relationship of OAL function call as shown in Figure 4:

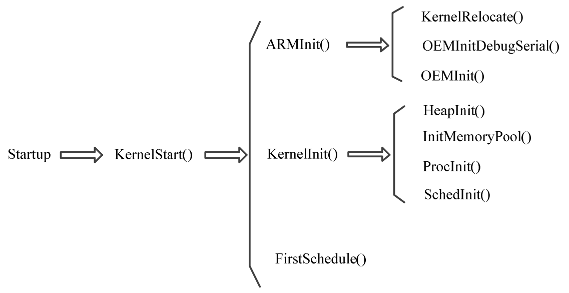


FIGURE IV. THE RELATIONSHIP OF OAL FUNCTION CALL

(1) Startup() is the entrance function of OAL, written in assembly language. This code has completed a lot of initialization work in BootLoader, then it begins to perform after BootLoader guides. Therefore, the Startup() in OAL does not need to repeated. The program just jump to the main control function KernelStart() to start.The code which will be implemented is as follows:

```
add r0, pc, #g_oalAddressTable-(.+8)
bl    KernelStart
```

(2) The primary task of ARMInit() function is to initialize the hardware platform of the inkjet printer, including Cache, interrupt, clock and kernel transmission layer etc.. In this process, KernelRelocate(), OEMInitDebugSerial() and OEMInit() will be mainly called. And the re-position function code and the initialization serial port function code are shared with BootLoader. The main hardware initialization work is done in the OEMInit() function.

(3) The OEMInit() function calls OALIntrInit() functio n to initialize the interruption of the system, calls the Co nfigureGPIO() functi-on to configure the universal port, c alls the OALTimerInit() function to initialize the clo-ck, a nd calls the OALKitlStart() function to initialize the KIT L connection. Part of I/O port initialization codes are sh own as follows:

```
s2440IOP->GPBDAT = 0x62;
s2440IOP->GPBUP          = 0x7FF;
s2440IOP->GPBCON= 0x2996A4;
…………
s2440IOP->GPHUP          = 0x7FF;
s2440IOP->GPHCON= 0x14A0AA;
```

The specific modifications refer to I/O part in S3C2440 chip manual so that the inkjet printer can configure the specific function through the port. The configuration of all I/O ports are implemented in init.c.

(4) KernelInit(): This function is used to initialize the operating system itself. This process is carried out in the kernel of the system, and OEM does not need to modify.

(5) This section of code is to allow the operating system to re-schedule, So that the first thread in the ready state can be selected to execute.

C. Driver Development

In the inkjet printer control system, the drive of nozzle abstract the hardware into a document so that managing hardware will be transferred the read and write control of corresponding files [6]. The application programs visit the corresponding standard flow interface function of the flow driver through API files which the operating system provides. The flow interface function includes XXX_Init(), XXX_Write(), XXX_Read(), XXX_ IOControl() and so on [7].

Written drive programs under WINCE environment, the first thing is to open up the address space of the register which can be done in the PRT_Init() function. The specific implement- ation of the spray printing work of nozzle will be completed in PRT_IOControl() function. The development of driver should be strictly in accordance with the nozzle control timing. XJ128 work sequence is shown in Figure 5:

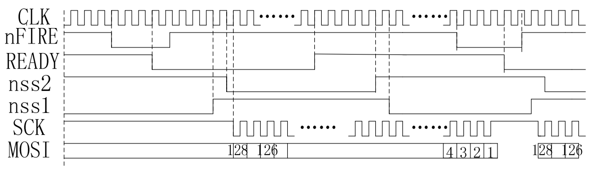


FIGURE V. XJ128 NOZZLE CONTROL TIMING

According to Figure 5, the code of nozzle driver can be completed in PRT_IOControl() function. The specific implementation is shown in Figure 6:

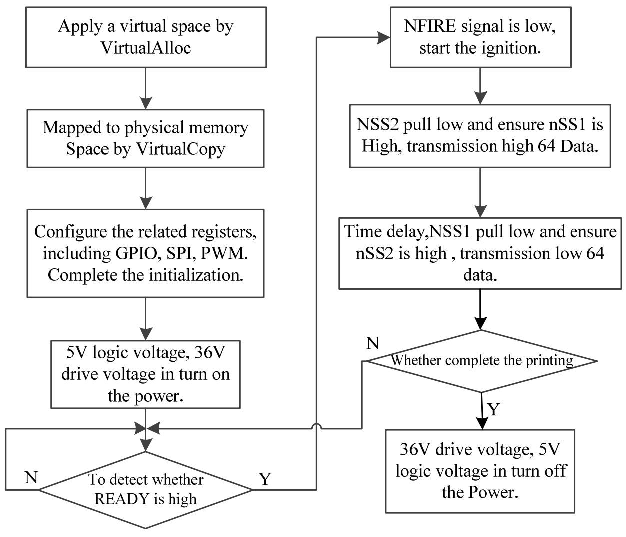

Apply a virtual space by VirtualAlloc
Mapped to physical memory Space by VirtualCopy
Configure the related registers, including GPIO, SPI, PWM. Complete the initialization.
5V logic voltage, 36V drive voltage in turn on the power.
N
To detect whether READY is high
Y
NFIRE signal is low, start the ignition.
NSS2 pull low and ensure nSS1 is High, transmission high 64 Data.
Time delay,NSS1 pull low and ensure nSS2 is high , transmission low 64 data.
N
Whether complete the printing
Y
36V drive voltage, 5V logic voltage in turn off the Power.

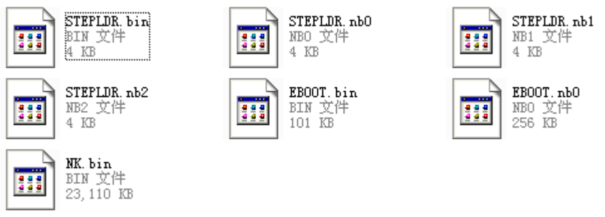
FIGURE VI. FLOW OF XJ128 NOZZLE DRIVE

D. Modification of the Configuration File

In WinCE system, the configuration files play a crucial role in the development of BSP, which is equivalent to build a system as well as a commander behind the project development. Configuration files not only specify the compilation rules of the source code, but also guide the operating system how to create a runtime CE image. And through the modification of the configuration files, the system can be extended and cut [8]. We can also add the nozzle driver and printer application programs through modifying the configuration files which will not to introduce here.

IV. CONSTRUCTION OF IMAGE FILE

Customizing operating system is a compli-cated and cumbersome process. Above all, whe-ther the BSP development is right or wrong is the prerequisite of the customization. And in the BSP development, the modification of the configu-ration file is the most important, which guides the construction of the operating system. The entire process of customizing operating system is comp-leted in Platform Builder [9]. After the Platform Builder integrates development environment to build the operating system, the runtime image of the operating system can be got and stored under the directory Release. The key system image file under the files are shown in Figure 7:


STEPLDR.bin
BIN 文件
4 KB

STEPLDR.nb0
NB0 文件
4 KB

STEPLDR.nb1
NB1 文件
4 KB

STEPLDR.nb2
NB2 文件
4 KB

EBOOT.bin
BIN 文件
101 KB

EBOOT.nb0
NB0 文件
256 KB

NK.bin
BIN 文件
23,110 KB

FIGURE VII. OPERATING SYSTEM IMAGE FILES

After customizing operating system, some runtime image files, like STEPLDR.nb1, EBOOT
.nb0 and NK.bin can be got and will be down- loaded to the hardware platform of the printer for repeated debugging. After the data copy into the development board is completed the first time, the development board will be restarted, and the start interface is shown in Figure 8.

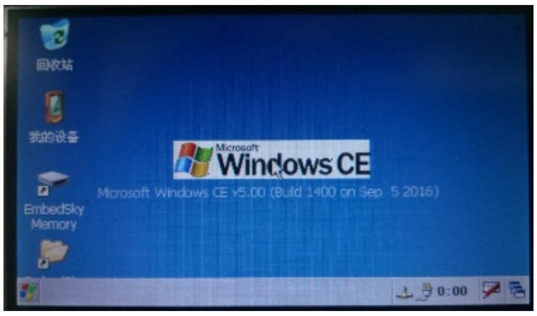


FIGURE VIII. START INTERFACE OF INKJET PRINTER PLATFORM

V.EXPERIMENTAL RESULTS

The printing work can be completed after running the application in the inkjet printer hardware experiment platform and loading the required printing graphics. As Figure 9 show, the printed images are vivid.

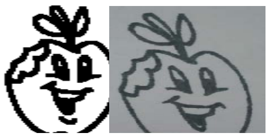


FIGURE IX. EFFECT OF THE FRONT JET PRINTING

VI. CONCLUSION

This paper puts forward a design of high resolution inkjet printer of embedded system ARM+WinCE platform in terms of hardware and software. The focus on the development of BSP provides a little reference to the study of the BSP development of other hardware platform under the same operating system. At the same time, this paper has carried out the development of the nozzle driver in the specification ways. The experimental results show that the system of inkjet printer is stable and the effect of jet printing is good, which means the high resolution inkjet printer Designed in this paper will have a good market value.

REFERENCES

[1] Zhuang Xiaoqi, Zhang Lijun, Fang, et al. Design and implementation of embedded inkjet printer control system based on ARM9[J]. Computer measurement and control,2010,18(8):1799-1801,1804.

[2] Neil Clymer,Shigeru Asaba. A new appro-ach for understanding dominant design: T-he case of the inkjet printer[J]. Journal ofEngineering and Technology Management, 2008,25(3):137-156.

[3] Tang Jun, Gong Xuecheng. Design and implement- tation of Bootloader based on WinCE system and S3C2440[J].Coal technology,,2010,29(5):164-166.

[4] Fan Shurui, Zhao Yanfei, Gao Tiecheng. ARM processor and C language development and Application[M]. Beijing University of Aeronautics and Astronautics Press, 2013.

[5] He Zongjian. Windows CE embedded system[M]. Beijing:Beijing University of Aeronautics and Astronautics Press,2006.

[6] Zhang Yi, Wang Haitao. Implementation of touch screen driver based on WinCE5.0 and S3C2410A[J].Journal of Chongqing University of Posts and Telecommunications (natural science edition),2008,20(6):742-745.

[7] Chen Danqiang, Wang Jinghua, Wang Chuang. Electric bomb controller detection research based on ARM-WinCE [J]. Measurement & Control Technology, 2015,34(01): 99-102.

[8] Zhou Jianshe. Windows CE device driver and BSP Development Guide[M].Xi'an:China Electric Power Press, 2008

[9] Kawamata N, Koga Y, Sugiyama W, et al.Inkjet printer: U.S. Patent Application 14/060,428[P]. 2013-10-22.