Linux Porting onto a Digital Camera

Shigeki Ouchi

RICOH Company Ltd., Software R&D Group
shigeki@src.ricoh.co.jp

Alain Volmat

RICOH Company Ltd., Software R&D Group
avolmat@src.ricoh.co.jp

ABSTRACT

The RDC-i700 is one of the digital camera shipped by Ricoh, which has 3.2M pixel CCD, a PCMCIA and a CF slot, 3.5 inch touch screen LCD. It originally works on VxWorks, but we ported Linux onto it to make it "programmable camera". The architecture of digital camera is totally different from PC, so it needs some modification, new design decision and new implementation to make Linux kernel to fit this new device. We'll introduce these technical work with also non-technical work to make it popular in this paper.

1 Why Porting Linux

RDC-i700, shown in Fig.1 is a commercial product which was already shipped. It works fine on VxWorks, one of the famous RTOS, and has system extensibility like PCMCIA or CF slot. You might think it isn't needed to change it to another OS, Linux. The reason we ported Linux onto this existing digital camera is to make it "programmable camera." Once it becomes a programmable device, many VARs (Value Added Reseller)or individual programmers may write a lot of useful software for it. Then it will be a good platform for business imaging use. In other words, our goal is to make a digital camera an application platform with the help of Linux power.

2 The Hardware

The RDC-i700 is one of the digital camera shipped by Ricoh. It has not only camera-related peripherals, but also has two PCMCIA slots, which allow users to send and receive images via e-mail, and to browse the Internet. So this is called as the world's first Internet-ready digital camera.

To give an detail explanation, let's divide all peripherals into two parts, PC-like peripherals and camera-related peripherals.

2.1 PC-like Peripherals

PC-like peripherals include CPU, expansion slot(PCMCIA and CF), input device(touch panel), output device(LCD) and so on.

RDC-i700 uses Hitachi SH7709A as a main CPU, which is one of the SH3 architecture 32bit RISC CPU including MMU and several internal peripherals such as serial controllers, I/O ports, AD/DA converters. Even though SH3 works in both big endian and little endian, only big endian is available on this architecture because the selecting pin of endian is hard-wired.

A PCMCIA controller (MRSHPC-02) is attached to SH3 CPU, allowing users to use a 16bit PCMCIA card and a CF card, as shown in Figure.2.

Input devices are a touch panel and several keys and buttons. Even though keys and buttons are very few and different from keyboard of PC, touch panel is almost same with the one of PC, and attached to SH3 via serial interface.

LCD is the only internal Output device, the

Figure 1: RDC-i700

controller of which is attached to SH3 via IPP(Image Processing Peripheral). A part of IPP functions plays a role similar to VGA chip in PC, but its interface is completely different.

Network connectivity using PCMCIA feature and 3.5 inch big touch screen make it possible to run a various range of applications. These can include camera specific application, such as capturing stills, sounds or videos, but also include PC-like application, such as sharing them wirelessly with anyone at anytime, sending and receiving an e-mail with attachments, sending a fax, surfing the Internet. These PC-like devices are necessary for these kind of applications.

2.2 Camera-related Peripherals

IPP is the most important chip which makes this device a digital camera, and other camera-related peripherals , CCD Controller, front-end chip of CCD, Image SD-RAM are located around it.

2.3 CCD and its Controller

CCD is just a sensor which captures an image, and captured image is stored into Imaging SD-RAM via IPP.

2.4 Mechanical Parts

Mechanical Parts include zoom, focus, iris and mechanical shutter. Because these parts needs high precision movement, stepping motors are used. The CPU has to set ports of the motor at a quite fast frequency in order to make the motor move rapidly. In that case the timer period is only several milliseconds.

2.5 Electric Flash

In case of Electric Flash(strobe), the difficulty is not controlling something at very high speed, but making perfect synchronization between the moment the strobe is going to flash and the moment the CCD will capture the image. In that case also, we will need precision of only very few milliseconds.

2.6 Auto Exposure and White Balance

In fact, Auto Exposure and White Balance are not the hardware but the algorithms. But since these two are indispensable functions to make this device a digital camera, we'll illustrate them here.

Figure 2: Block diagram of RDC-i700

Auto Exposure is the algorithm in charge of adjusting CCD's shutter opening time in order to have good image, in both dark and bright conditions. White balance algorithm adjusts Image Processing Peripherals settings to have good color matching between the image generated by the Image Processing Peripheral and the reality.

Several implementations of those 2 algorithms can exist (some needs very heavy calculations and other can be very simple), but the main issue is that those algorithms have to be performed very often (in the worst case, every frame of the CCD).

This second part introduced particularity of digital cameras. The next part will discuss about how those functions have been implemented into the Linux RDC-i700.

3 Existing Resources

Because Linux is an open source project, we can use a lot of existing resources for this new hardware. SH-Linux kernel can be obtained from http://sourceforge.net/projects/linuxsh and has been merged into original kernel source tree, and cross development tools are also on the web; http://www.sh-linux.org/. Of course, even though it supports SH3 CPU, it doesn't work on our digital camera without modifications. But we need much less labor than in the case of porting to new CPU.

SH3 CPU has internal serial interfaces, and SH-Linux supports serial console using this port. Since this interface is CPU internal, of course it doesn't depend on board architecture. So at the early stage of porting, we could use serial console for efficient debugging.

Addition to CPU internal peripherals, our PCMCIA controller is same series of the one used for Hitachi Solution Engine, which is an evaluation board for SH3. Since this board is supported by SH-Linux and one of the community programmer wrote a driver for this PCMCIA controller, we could support our controller with a very few modification.

Since this camera is not PC but embedded hardware, there's no PC-like BIOS. It means we have to make an boot loader by ourselves. Fortunately, SH-Linux project are also developing IPL called sh-ipl+g and boot loader called sh-lilo. At the first stage of our development, since we used

Compact Flash(CF) as a boot disk, these two software could be used. For our environment, a little fix was needed because our target works in big endian, but it wasn't a big labor. Recently we added a small code to the IPL which read compressed kernel image and jumped there, and made our code romized. This means now we need no another loader other than modified IPL.

Of course, a lot of user-land programs, including a small window system for embedded system, are also available with almost no modification.

4 Things Should be Newly Decided

As we mentioned in the previous section, we can use a lot of software resources with no or little modifications, but since there's no another digital camera which uses Linux for its operating system, we faced several designing issues, window system, file system, making it small and so on.

4.1 window system

There are several window system or GUI environment specialized for embedded systems. Qtopia, Microwindows and Shikigami[7] are good example. Since Shikigami was made by Japanese Company, AXE Inc., it has good ability to treat Japanese, for example handwriting character recognition for Japanese. This is really advantageous for us, because we are Japanese company, but it is a proprietary product and not open source.

On the other hand, Qtopia and Microwindows are available to the public including source codes. Qtopia is based on Qt/Embedded tool kit developed by TrollTech, and seems to be more powerful. Actually it is used by a famous PDA, Sharp Linux Zaurus. But it needs more resources, especially memory. Since our RDC-i700 has only 16MB RAM, making it work with attractive application software, say Web browser or Java seems to be really difficult.

Microwindows is a really small window system and seems to work fine on several architecture. Because RDC-i700 uses SH3 in bigendian mode, and we sometimes have to fix endian bugs when porting application software, so portability is one of the most important issue. Looking into source codes of Microwindows proved that there's only a few part which needs modification. And also Programing API is very similar to the one of X11, this appeals to us, Unix programmers.

That's why we decided to port microwindows to RDC-i700. After a short work, it works fine on RDC-i700.

4.2 file system

As for deciding the file system, we have to think two usages, one for Linux directory tree including user land programs, one for storage of pictures taken. We uses EXT2 file system for user land trees without a much thinking, because it should be finally romized and it is only located on CF when developing.

But we have to take care of the file system for storage. When CF is used as storage, it should be FAT format because of compatibility with other digital cameras. So issue is the case when internal 8MB NAND Flash is used for storage. As you know, Flash memory is already supported on Linux 2.4.x by JFFS and JFFS2. JFFS2 supports NAND Flash and it always compresses data. On the contrary JFFS doesn't compress data but doesn't support NAND Flash. Since picture data are usually stored on JPEG format which is already compressed, compressing feature of JFFS2 just causes performance down. At first we thought to try modifying JFFS2, but fortunately we found yet another file system for flash memory, YAFFS[6].

YAFFS not only saves us the time for modifying JFFS2, but also can save users almost 40% time for storing JPEG file.

4.3 making it small

Since RDC-i700 has only 8MB Flash ROM, everything should be kept small. These can include kernel, user-land programs and libraries, and so on. Currently kernel size is almost 1.5MB including TCP/IP stack and dozens of loadable modules. Obviously, some modules, say bluetooth drivers, are only used for special purpose. For the development use, this might not be so big problem, because we can use big CF card as its root file system. But for the practical use, this becomes a big issue.

Table 1: Comparing resources of window systems

	Qtopia	Micro windows	Shikigami
ROM	6-8M	1-2M	2-8M
RAM	4-8M	1-2M	4-8M

There are several known way to reduce footprint. one way is to omit unnecessary functions in glibc. Since glibc is really big, careful and laborsome work may reduce almost 1MB. Another way is to use "Crunched binaries." Busybox is one of the implementation on Linux, and it shrinks the user-land programs and libraries in the case of romized version very effectively. Since glibc is bigger than 1MB and there are a lot of small utility programs on /usr/bin, just replacing it with Busybox and uClibc reduces a lot of storage space.

5 Things Should be Newly Implemented

So far, we explained existing resources which needs no or little modifications and also explained what decision we have to make from the design point of view.

On the other hand, our digital camera is the worlds' first Linux embedded one, drivers for camera-related peripherals are missing. Indeed, this camera is an existing product and the product version of RDC-i700, which uses VxWorks, controls camera-related peripherals well. But nevertheless a lot of effort is needed for porting these drivers onto Linux, because drivers should be written in completely different manner on these two OS.

We show our design of device drivers in Fig.3.

5.1 CCD

CCD is the most important part which make this gadget a digital camera. Actually, CCD is just a sensor, it is attached to SH3 CPU via F/E (Front End) chip and IPP (Image Processing Peripheral), and is controlled by SH3 via these two chips. Since IPP driver is implemented as if it was a library inside the kernel, it can be called from another drivers and offers following functions.

- JPEG Compress/Decompress
- YUV - RGB Conversion
- Video Output (for both LCD and TV)
- Image Scaling

5.2 Electric Flash

Electric Flash, or sometimes called Strobe, enables to charge or flash the strobe. You may guess how to use this driver as following;

1. open /dev/strobe
2. send CHARGE or FLASH request via ioctl
3. close /dev/strobe

Of course, we can implement strobe driver like these if you just use it independently. But this doesn't make sense. You need to precisely synchronize the moment when CCD takes image and the moment strobe emits light. For that reason, we treated strobe as attribute of CCD, which means strobe-on or strobe-off, in the user-land, and the functions of this driver are called from CCD driver inside the kernel.

5.3 Focus Sensor

Focus sensor allows to measure the distance between the camera and the target. This unit calculate it internally and just output the distance. So if it was attached to CPU via serial chip, it would be really easy to use this. But in fact, unfortunately, it is attached to the IO port of SH3 directly, so it needs its own synchronization. Currently we didn't implement this driver, so focus is always adjusted by calculating how it is adjusted in full range and moving the best position.

Figure 3: Layers of Device Drivers

5.4 Mechanical Parts

Mechanical Parts include zoom and focus. As I mentioned above, these parts need high precision movement. Stepping motors used needs 1.8 milliseconds to become stable for each stepping. This means Linux default 10msec timer makes the movement really slow, because at each stepping, we need to wait 10 or sometimes 20msec instead of 1.8 msec. We solve this issue by changing kernel macro HZ from 100 to 1000, which permits us to use 1msec timer.

In this section, we've discussed what should be newly implemented.

6 Future Work

By our past work, Linux on the RDC-i700 works stably, and not only basic function as a digital camera but also application using network feature works fine. In other words, our implementing work is almost finished, but we have still several issues.

6.1 Technical Issue

6.1.1 Power Management

Biggest issue is power management. Currently we don't have any power management in the kernel, so the battery life is only 20-30 minutes when using Wireless-LAN card via PCMCIA slot. Obviously, we need big enhance in this field. The first work to do is to add power management function into each device driver. This seems to be not so difficult, just we need to implement sleep mode according to each specification of devices. The key point is how we integrate each device, i.e., when and which device we should put into sleep mode. Recently, as embedded Linux is getting booming, several project teams made some suggestions. We'll implement it based on their works.

6.1.2 Packaging

As I mentioned previously, current CF usage is too big to romize. Since each user wants to use digital camera in each own way, all-in-one package is convenient in the case we use CF as root

partition. But this doesn't hold true in case of romized version. Application specific camera is supposed to be used for a custom order, and we have only 8MB ROM in the camera, so we need to select only required kernel loadable module, libraries and application for each demand. This is not so difficult for skilled Linux developer, but it seems to be labor for application programmers. We have to create the system which automatically select required things from all-in-one package.

6.2 Non-technical Issue

Our goal is not just making programmable camera but making benefit from it. Hobby users may want to play with it and they are really important to boost a market for the initial period of time.

6.2.1 Distribution

In order to boost our project, We are now preparing for distributing the source codes. Although we still needs some more coordination with our legal section, we hopefully think it will be available soon.

6.2.2 Making Community

After successfully distributing our source codes, we'll have to make effort to make supporting communities. Even though source codes becomes opened, Big market is not created automatically. We have to support initial hobby users or small communities who are willing to play with it, and have to promote it. If such communities are getting bigger, third vendors may also get interested and start thinking of going into this field.

6.2.3 Search of Killer Application

Even though initial hobby users are really important for the success of this kind of development, there seems to be bigger market in business use.

For business use, platform itself doesn't make money so much. We have to get a lot of business from "solution system." Since it's impossible for only us to make this kind of system for various possible customer, obviously we need help of SIers or third vendors. In order to do that, we need to prove to SIers or third parties that making application software on this platform is profitable for them.

Then how do we prove it? Making killer application is probably one of the best way. To say honestly, our team is not so big to make application software by ourselves, so we've asked several teams to make their own application on top of this camera. And also we've accepted internship students who also implement their own application idea on it. These tasks are still in progress and we are still searching killer application.

6.3 Feedback to Hardware Design Division

In our project, we've been porting Linux onto an existing digital camera hardware. This means that we didn't design and decide the specification of the camera so that Linux fit well. If it would be possible to design camera hardware with view of Linux as an OS, it should be easier to implement camera-related peripherals drivers and to port a window system. Even though we are working not in commercializing division but in R&D division, we'll make several suggestions about new hardware design to them as follows.

Linear Access to Imaging SD-RAM linear and direct access to Imaging SD-RAM from SH3 CPU make it much easier to implement framebuffer and primitive functions for embedded window system.

Programmable IPP Using FPGA or other programmable device as an IPP makes this camera more flexible and practical "programmable camera." Of course it cost high, but it's still worthy for business imaging use.

Serial issue Usage of serial line in this camera is really complicated because of hardware cost. We know cutting the hardware cost is really important, but size of embedded software, and consequently software design and implementation cost, is getting bigger and bigger. So we think now is the time to change the decision.

Sub CPU issue In this camera, sub CPU, which is attached to SH3 via serial line, is used to control several devices like keys, buttons and RTC. This makes it complicated to

control devices which is connected to the sub CPU. We'd like to suggest to change more controllable architecture.

7 Conclusion

We ported Linux onto RICOH's existing digital camera, RDC-i700. Thanks to the Linux and other free software communities, it took shorter time to port kernel core part. We faced some issues including writing drivers for camera-related peripherals because this is the world's first digital camera powered by Linux, it works almost fine and becomes a "programmable camera". We hope we can distribute it soon.

8 Acknowledgment

First of all, we have to thank all the communities which are supporting Linux and other free software. Without their work, our project would have never been achieved.

And also Camera-Product division in our company was willing to support us. We are not the professional of the camera, so they were really big help for us.

Here, I have one request to readers and audiences. Our company, RICOH has currently NO commercial plan for making this project, porting Linux onto RDC-i700, into the market. So please do NOT contact Camera-Product division directly. Instead, feel free to contact each of two authors, Alain and me(Shigeki).

References

[1] SH-Linux kernel; http://sourceforge.net/projects/linuxsh

[2] SH-Linux Cross development tools; http://www.sh-linux.org/

[3] SH-Linux Boot Sequence; http://linuxsh.sourceforge.net/docs/abe/doc/linux-sh-kernel-bootup_E.php3

[4] Linux Device Driver (Japanese Edition); ISBN 4900900737

[5] Linux Kernel 2.4 (Japanese Edition); ISBN 4873111331

[6] YAFFS; http://www.linuxdevices.com/articles/AT9680239525.htm

[7] Shikigami; http://www.sikigami.com/english/