

FRAMEWORK TO PORT LINUX KERNEL ON POWERPC BASED EMBEDDED SYSTEM USED FOR TELECOM APPLICATION – IPBTS

Rajendra Prasad.M¹, S. Ramasubba Reddy², V.Sridhar³

¹Associate Professor,

ECE Department, Vidya Jyothi Institute of Technology, Hyderabad, India

rajendraresearch@gmail.com

²Assistant professor,

ECE Department, Chaitanya Bharathi Institute of Technology, Proddatur, India

ramasubbareddy.s@gmail.com

³Assistant Professor,

ECE Department, Vidya Jyothi Institute of Technology, Hyderabad, India

varadasri@gmail.com

ABSTRACT

The tremendous growth of telecom applications running on specific customized hardware known as an embedded system which necessitates analysing and designing a frame work to port linux kernel on PowerPC based board for telecom application. In view of economical benefit of transmission line cost, IP-BTS (IP based Base Transceiver Station) is developed which is a compact indoor base station used for high- density multi-band base stations that accommodates IP transmission lines. This paper proposes the procedure for porting linux kernel on PowerPC based custom boards, which means making the operating system work on unfamiliar hardware to run IPBTS application software. This paper elaborates the methods and progress of transplanting linux kernel on customized board which includes setting up the development environment on host PC, obtaining linux kernel source, configuring the linux kernel, cross compiling linux kernel, loading and running linux kernel on PowerPC based custom boards. The resulting embedded systems are used for various wireless telecom applications running on custom boards in telecom domain.

KEYWORDS

Embedded systems, porting, MPC 85xx processor, linux kernel, IPBTS, customized hardware.

1. INTRODUCTION

As the embedded system technology advances in research and time-to-time market pressures, an embedded system has to be developed in order to handle complex real time embedded telecom applications running on network processor based custom boards. While developing embedded systems [19] porting of an operating system to the custom boards is an important stage. Because of the open source, rich code resource and supporting many kinds of CPU architecture, linux is making steady progress in the embedded arena. This project comprises of software development for IPBTS which includes transplanting linux operating system to a custom board which is based on MPC8548E processor. Porting linux to a new custom board requires several activities:

- Determining the processor custom board architecture.
- Analysis and identifying source code files to change based on the platform architecture of the custom processor board.
- Upgrading the changes to the files which include writing the code.

- Configuring linux kernel source is as per the on the custom board as per the hardware specifications.
- Building test code and analysing the board to ensure that the embedded linux is running properly on the custom board.

The linux kernel source tree [1] [2] is consists of various files and subdirectories shown in figure 1. The important sub directory is the arch. In the source code of linux-2.6.10/arch, different architectures are represented such as arm, x86, sun, and ppc. This paper describes on ppc, the PowerPC processor based custom board for telecom applications. The kernel source code has three major components, the processor architecture independent code, the architecture dependent code, and the configurations. The PowerPC processor architecture related code is available under the linux-2.6.10/arch/ppc directory.

The configuration details are in the linux-2.6.10/arch/ppc/configs subdirectory. The architecture independent code is at the top level source code, linux-2.6.10/kernel, mm, net, fs, and others. Usually, no changes are required in the architecture independent code. However, the linux-2.6.10/drivers directory may need changes as per the hardware specifications of the custom board. The source code files referenced above are taken from open source repositories. There will be changes in these components as per the requirement of the project specification and hardware specifications. [9].This sub directory contains driver code for peripheral devices on custom board as IDE, PCI, video and audio etc, figure 1 shows graphically all top level directories.

1. linux-2.6.10/arch consists of the processor architecture dependent directories.
2. linux-2.6.10/drivers consists of the driver code directories.
3. linux-2.6.10/include consists of all the header files for all the architectures, this paper elaborates and describes only specific asm-ppc includes files related to our custom board as shown in the figure 2 and figure 3 and its hardware specifications are described in Section 3.1

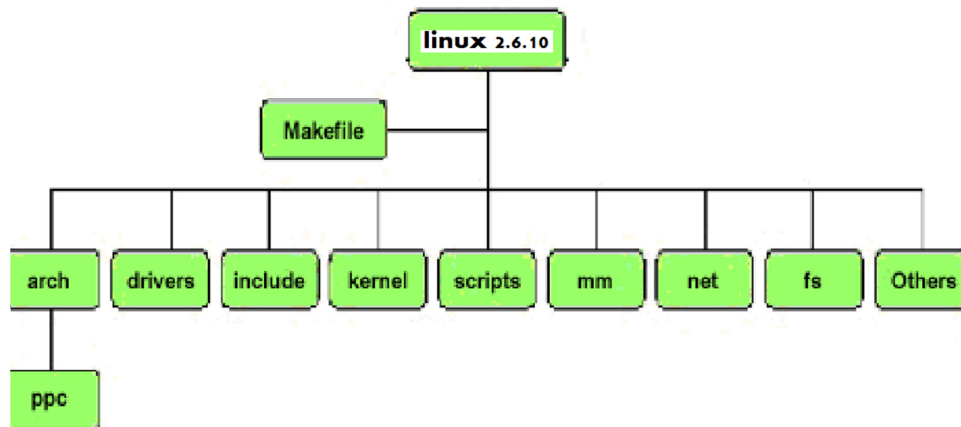


Figure 1. linux kernel source tree

4. linux-2.6.10/kernel includes the architecture independent code for the kernel like fork, spinlock.
5. linux-2.6.10/scripts consists of the scripting files used by the Makefile.

6. linux-2.6.10/mm contains the memory management code.
7. linux-2.6.10/net contains the networking code like Ethernet code.
8. linux-2.6.10/fs contains the code relating to the file systems.

1.1. The subdirectories in kernel source

In this section we described required subdirectories in the linux kernel source.

linux-2.6.10/arch/ppc/configs – In this subdirectory all standard configuration details will be available. If a PPC based custom boards hardware specifications are defined here, then a generic configuration for that PPC based custom board can be built [15].For the Sandpoint board we can use “make sandpoint_defconfig” command to build generic or standard configuration.

linux-2.6.10/arch/ppc – This is the core, standard and the starting sub directory for all the architecture dependent code for the PowerPC.

linux-2.6.10/arch/ppc/boot – This sub directory contain code for Linux boot loader. The important file head.S is the first program called by the BIOS while loading Linux Operating system.

linux-2.6.10/arch/ppc/boot/images – After building linux kernel image, this sub directory contains the kernel images like zImage or zImage.initrd.

linux-2.6.10/arch/platforms – This directory contains the low level source code for initializing the board. Usually, there is a source (.c) and a header (.h) file, which contains the low level code to continue to bring up the linux kernel

linux-2.6.10/include – In this directory header files for all the supported architectures are available. In the configuring phase (make menuconfig phase) a soft link, asm points to the exact architecture header files. In the case PowerPC, asm -> asm-ppc and these include files are specific to the kernel.

linux-2.6.10/arch/ppc/kernel – In this subdirectory the architecture dependent code of the kernel for PowerPC processor, such as head.S, the CPU specific start up code, cputable.c etc is available. This paper describes and elaborates the methods and progress of transplanting linux kernel on customized PPC based board used as an embedded system for IPBTS telecom application [10][16].

2. RELATED WORKS

Recently, several works of researchers have been focusing on adapting a real time kernel to micro processors or micro controllers based customized processor boards used for different applications in the telecom domain. ZHOU Qingguo [6] describes the procedure to port embedded linux to the XUP Virtex-II Pro development system and using serials of development tool kits. The Virtex-II Pro serials development system produced by Xilinx company provides an advanced hardware platform that consists of a high performance Virtex serials platform FPGA surrounded by a comprehensive collection of peripheral components that can be used to create a complex system and meet different special embedded application areas.

Sun Yanpeng [8] describes the methods and progress of transplanting the embedded linux to the target board based on the S3C2440 processor which includes the establishment of cross-compiler environment, the reduction and compilation of start-up code (boot loader) and linux kernel and the construction of the root file system with the point focused on the structure and function of boot loader.

Hu Jie [7] presented a novel transplanting method for embedded linux kernel is as well as its cut, compile and porting process under ARM platform. Chun-yue Bi [19] focused on the analysis of

International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011
Linux kernel, on the basis of research analysis, explored the key technologies of embedded Linux such as linux porting methods on ARM, improvement of linux real-time performance.

The MPC8548E architecture strongly integrated with a PowerPC™ processor core and system logic required for networking, IP based telecom and wireless infrastructure applications. The

MPC8548E is a member of the PowerQUICC III™ family of devices [14] that combine system-level support for industry-standard interfaces with processors that implement the PowerPC architecture [11]. Because of the above features in networking and telecom domain this paper describes and elaborates the methods and progress of transplanting linux kernel on customized PPC based board used as an embedded system in telecom domain.

3. BUILDING LINUX KERNEL IMAGE FOR POWERPC

3.1. PPC based target board hardware specifications

- The PPC based target board is based on Free scale MPC8548E PowerPC processor. The Processor board is designed as a flexible board to accommodate MPC8548E and MPC8543E processors.
- The PPC based target board has a DDR-2 SODIM slot which can accommodate standard DDR-SODIMM memory modules fro multiple vendors. The processor board is mounted with a 256MB memory.
- The PPC based Target board has two 32 MB flash (S29GL256N) mounted on board.
- The board has 2 serial eproms as Boot sequencer EEPROM and Board ID EEPROM. The serial EPROMS are 8Kb and are connected to the processor I2C bus.
- The processor board has two 10/100 Base-T Ethernet ports for communication.
- The top and bottom side views are depicted in figure 2 and figure 3.

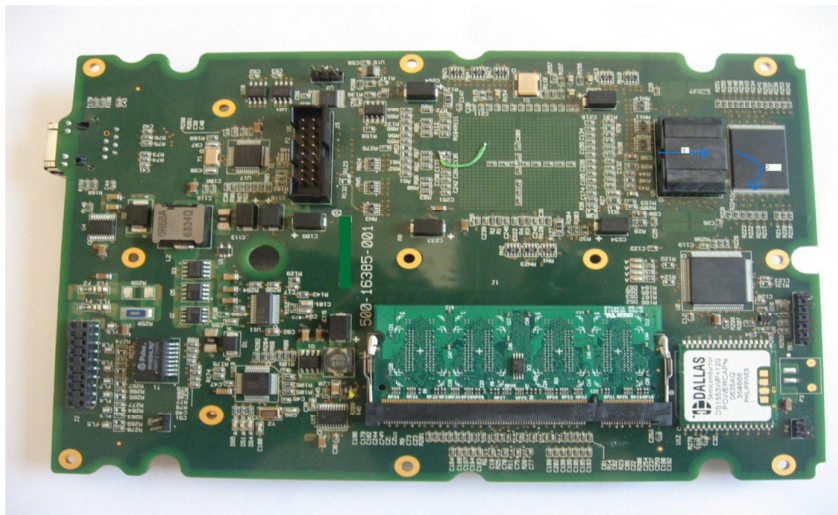


Figure. 2 Top side view of custom board

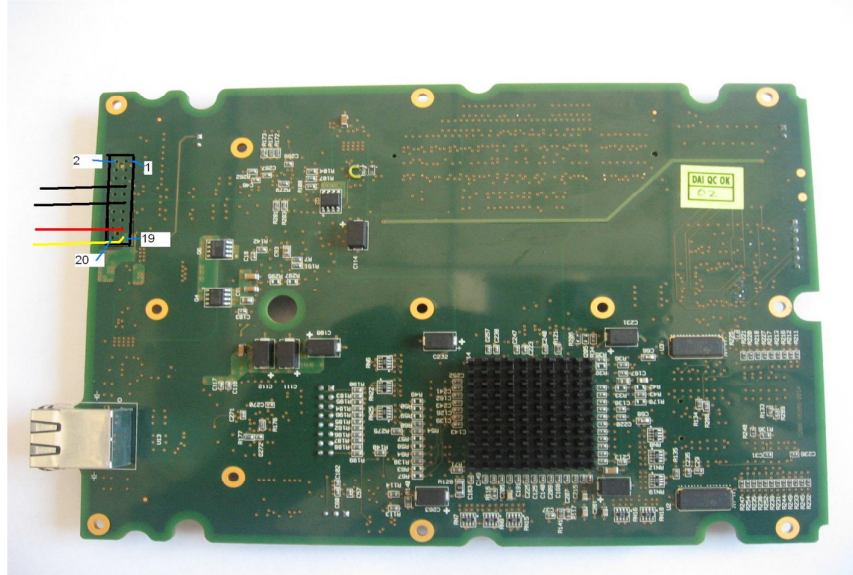


Figure.3 Bottom side view of custom board.

3.2. Linux kernel Development Environment

1. Download rpm for cross compiling “tc-mtwk-lnx-e500-sp-3.4.3-1.i686.rpm”.Thanks to support team of public LTIB project for providing free rpm [3] [13].
2. Configure and install “tc-mtwk-lnx-e500-sp-3.4.3-1.i686.rpm” as a root user using following command.

```
# rpm -i tc-mtwk-lnx-e500-sp-3.4.3-1.i686.rpm
```
3. The new cross compiler for PowerPC is now available as “/opt/mtwk/usr/local/gcc-3.4.3-glibc-2.3.3-spe/powerpc-linux-gnuspe/bin/powerpc-linux-gnuspe-gcc”
4. Add the PATH as the last line of .bashrc [5]

```
#cd /root  
# vi .bashrc
```
5. A specimen snapshot of .bashrc file is provided below. bashrc file contents

```
# .bashrc  
  
# User specific aliases and functions  
  
alias rm='rm i'  
alias cp='cp i'  
alias mv='mv i'  
  
# Source global definitions  
if [ f /etc/bashrc ]; then  
./etc/bashrc  
fi  
PATH=/opt/mtwk/usr/local/powerpc-linux-gnuspe/gcc3.4.3e500glibc2.3.3spe/bin:$PATH
```

6. Use “bash” command as a root user.

3.3. Getting the kernel source Tree

1. Download kernel source (2.6.10) from kernel.org.
2. Edit the top level Makefile in kernel source for PPC based custom target board option as
“CROSS_COMPILE”=”powerpc-linux-gnuspe-gcc”
”ARCH”=”ppc”.

3.4. Building linux kernel Image for PPC

1. To build a linux kernel image for PPC based custom board configure linux kernel source for PPC based custom board as per the hardware specifications. Starting in the top level of the source tree. Each and every level of kernel source has a Makefile. Each top level Makefile calls a lower level Makefile [4].
2. Use “make menuconfig ARCH=ppc” command to configure kernel source for PPC based custom board.
3. To generate complete kernel images use “make” command.
4. After successful cross compilation kernel “vmlinux.gz”is available in arch/ppc/boot/images location.
5. Using U-boot source utility “mkimage” generate bootable image for PPC based custom board.

4. DEPLOYING KERNEL IMAGE ON PPC BASED CUSTOM BOARD

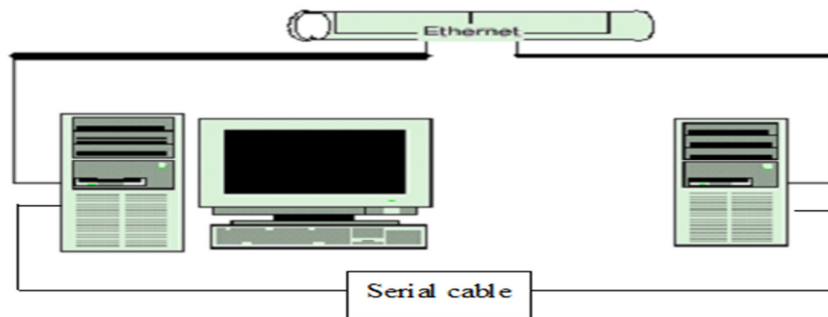


Figure 4 Linux host and PPC based target board.

1. Connect RS-232 C interface from the PPC based boards serial port to the Desktop linux host systems serial port of as shown in the figure 4. An Ethernet connection can also be established between the host and the target systems.

2. The kernel image generated from the procedure described in section 3 can be deployed to PPC based target board using tftp utility.
3. Use “ping” command to test the network connectivity between host and target systems from boot prompt and make sure that tftp server must be configured properly and working correctly on host system.
4. On u-boot prompt, set u-boot environment parameters as shown below [5] for network connectivity between target and host systems.

```
=>setenv ipaddr board_ipaddress
=>setenv serverip tftp_serverip
=>setenv gatewayip system_gatewayip
=>saveenv
=>setenv ipaddr board_ipaddress
=>setenv serverip tftp_serverip
=>setenv gatewayip system_gatewayip
=>saveenv
```

5. The PPC based custom boards bootable image is copied from arch/ppc/boot/images directory on the host system to “tftpboot” directory on the host system. The linux kernel image can be flashed to the PPC based custom board by following the steps below at boot prompt.

```
=> tftp 2000000 kernelimage
=> erase <kernelimage location> <kernelimagelocation +1ffff>
=> cp.b 2000000 <kernelimage location> <kernelimagelocation +1ffff>
=> protect on <kernelimage location> <kernelimagelocation +1ffff>
```

6. To boot linux kernel image on the custom PPC based board use “boot” command on U-boot prompt.

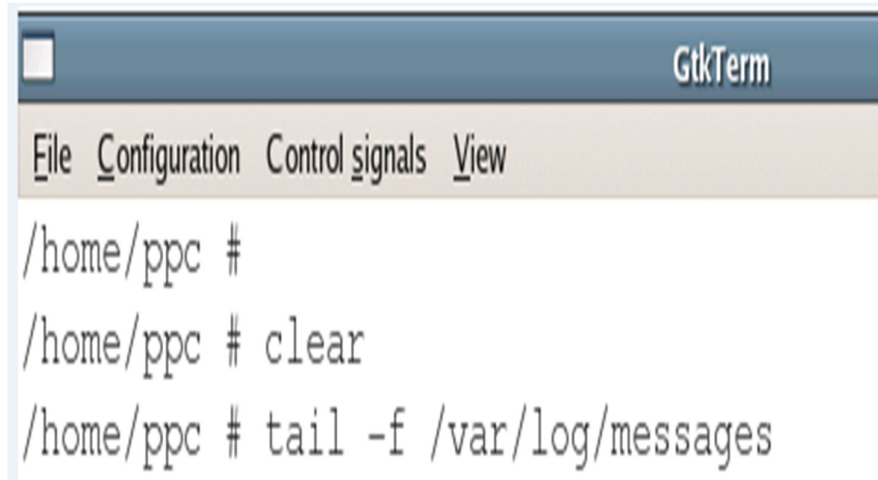
5. TESTING LINUX KERNEL IMAGE ON CUSTOMIZED BOARD

Configure serial communication software like gkterm/ minicom and set the com port for host and target board and boot the linux kernel. After booting linux kernel, figure 5 can be viewed on flash memory and ensuring linux kernel image is booted and working properly.

```
Image at FF800000:
  Image Name:   Linux for PowerPC
  Image Type:   PowerPC Linux Kernel Image (gzip compressed)

  Data Size:    991620 Bytes = 968.4 kB
  Load Address: 00000000
  Entry Point:  00000000
  Verifying Checksum ... OK
  ■
```

Figure.5 Testing of linux kernel image on flash on PPC based target board



```
GtkTerm
File Configuration Control signals View
/home/ppc #
/home/ppc # clear
/home/ppc # tail -f /var/log/messages
```

Figure. 6 Command prompt of PPC based target board

Figure 6 and 7 represents command prompt of linux operating systems on customized PowerPC based board and “tail” command is executed to view logging messages for linux kernel and IPBTS application [10] [16]. The log messages for linux booting process and conformation of log messages for IPBTS software application booted automatically with the help of shell script on linux operating system on customizedPowerPCbasedboard.


```
Jan 1 00:00:05 syslogd started: BusyBox v1.1.3
Jan 1 00:00:05 kernel: klogd started: BusyBox v1.1.3 (2007.10.11-11:35
+0000)
Jan 1 00:00:05 kernel: Memory CAM mapping: CAM0=256Mb, CAM1=0Mb, CAM2=
0Mb residual: 0Mb
Jan 1 00:00:05 kernel: Linux version 2.6.10
(root@localhost.localdomain) (gcc version 3.4.3 20041021 (prerelease))
#1 Thu Oct 11 16:55:32 IST 2007|
Jan 1 00:00:05 kernel: On node 0 totalpages: 65536
Jan 1 00:00:05 kernel: DMA zone: 65536 pages, LIFO batch:16
Jan 1 00:00:05 kernel: Normal zone: 0 pages, LIFO batch:1
Jan 1 00:00:05 kernel: HighMem zone: 0 pages, LIFO batch:1
Jan 1 00:00:05 kernel: Built 1 zonelists
Jan 1 00:00:05 kernel: Kernel command line: root=/dev/nfs rw nfsroot=
10.112.81.66:/nfsroot ip=
10.112.81.251:10.112.81.66:10.112.81.1:255.255.255.0:unknown:eth0:off
console=ttyS0,115200
Jan 1 00:00:05 kernel: OpenPIC Version 1.2 (1 CPUs and 44 IRQ sources)
at fcfbb000
Jan 1 00:00:05 kernel: PID hash table entries: 2048 (order: 11, 32768
bytes)
Jan 1 00:00:05 kernel: Dentry cache hash table entries: 65536 (order:
6, 262144 bytes)
Jan 1 00:00:05 kernel: Inode-cache hash table entries: 32768 (order:
5, 131072 bytes)
Jan 1 00:00:05 kernel: Memory: 252544k available (1652k kernel code,
412k data, 100k init, 0k highmem)
Jan 1 00:00:05 kernel: Calibrating delay loop... 997.37 BogoMIPS (lpj=
498688)
Jan 1 00:00:05 kernel: Mount-cache hash table entries: 512 (order: 0,
4096 bytes)
Jan 1 00:00:05 kernel: checking if image is initramfs...it isn't (no
cpio magic); looks like an initrd
Jan 1 00:00:05 kernel: Freeing initrd memory: 4673k freed
Jan 1 00:00:05 kernel: NET: Registered protocol family 16
Jan 1 00:00:05 kernel: PCI: Probing PCI hardware
Jan 1 00:00:05 kernel: Registering openpic with sysfs...
Jan 1 00:00:05 kernel: Generic RTC Driver v1.07
Jan 1 00:00:05 kernel: Serial: 8250/16550 driver $Revision: 1.90 $ 6
ports, IRQ sharing disabled
Jan 1 00:00:05 kernel: ttyS0 at MMIO 0xe0004500 (irq = 90) is a 16550A
Jan 1 00:00:05 kernel: ttyS1 at MMIO 0xe0004600 (irq = 90) is a 16550A
```

```
Jan 1 00:00:05 kernel: io scheduler noop registered
Jan 1 00:00:05 kernel: io scheduler anticipatory registered
Jan 1 00:00:05 kernel: io scheduler deadline registered
Jan 1 00:00:05 kernel: io scheduler cfq registered
Jan 1 00:00:05 kernel: RAMDISK driver initialized: 16 RAM disks of
131072K size 1024 blocksize
Jan 1 00:00:05 kernel: loop: loaded (max 8 devices)
Jan 1 00:00:05 kernel: eth0: Gianfar Ethernet Controller Version 1.1,
00:e0:0c:00:00:fd
Jan 1 00:00:05 kernel: eth0: Running with NAPI enabled
Jan 1 00:00:05 kernel: eth0: 256/256 RX/TX BD ring size
Jan 1 00:00:05 kernel: eth1: Gianfar Ethernet Controller Version 1.1,
00:e0:0c:00:01:fd
Jan 1 00:00:05 kernel: eth1: Running with NAPI enabled
Jan 1 00:00:05 kernel: eth1: 256/256 RX/TX BD ring size
Jan 1 00:00:05 kernel: eth2: Gianfar Ethernet Controller Version 1.1,
00:e0:0c:00:02:fd
Jan 1 00:00:05 kernel: eth2: Running with NAPI enabled
Jan 1 00:00:05 kernel: eth2: 256/256 RX/TX BD ring size
Jan 1 00:00:05 kernel: ns83820.c: National Semiconductor DP83820
10/100/1000 driver.
Jan 1 00:00:05 kernel: Universal TUN/TAP device driver 1.5 (C)1999-
2002 Maxim Krasnyansky
Jan 1 00:00:05 kernel: Uniform Multi-Platform E-IDE driver Revision:
7.00alpha2
Jan 1 00:00:05 kernel: ide: Assuming 50MHz system bus speed for PIO
modes; override with idebus=xx
Jan 1 00:00:05 kernel: NET: Registered protocol family 2
Jan 1 00:00:05 kernel: IP: routing cache hash table of 2048 buckets,
16Kbytes
Jan 1 00:00:05 kernel: TCP: Hash tables configured (established 16384
bind 32768)
Jan 1 00:00:05 kernel: NET: Registered protocol family 1
Jan 1 00:00:05 kernel: NET: Registered protocol family 17
Jan 1 00:00:05 kernel: indigo: speed = 100
Jan 1 00:00:05 kernel: eth0: PHY is Generic MII (20005c90)
Jan 1 00:00:05 kernel: interrupt transmit = 77
Jan 1 00:00:05 kernel: IP-Config: Complete:
Jan 1 00:00:05 kernel:      device=eth0, addr=10.112.81.251, mask=
255.255.255.0, gw=10.112.81.1,
Jan 1 00:00:05 kernel:      host=unknown, domain=, nis-domain=(none),
```

Figure.7 linux kernel booting log messages

IP BTS Controller application running on customized PPC board will support logging and tracing for all the protocols and modules using the linux logging facilities. There are many advantages of using linux logging facilities such as configuration of log files for different levels of logging, log rotate, remote logging. This application will read the logger related configuration parameters, for example severity level for logging, from the configuration file and send this to all the layers and modules during start up and initialization phase as shown in the figure 8 .

```

Jan  1 00:00:05 kernel: RAMDISK: Compressed image found at block 0
Jan  1 00:00:05 kernel: VFS: Mounted root (ext2 filesystem).
Jan  1 00:00:09 kernel: GPIODriver: no version for "struct_module"
found: kernel tainted.
Jan  1 00:00:13 kernel: indigo: speed = 100
Jan  1 00:00:13 kernel: eth2: PHY is Generic MII (20005c90)
Jan  1 00:00:13 kernel: interrupt transmit = 79
Jan  1 00:00:13 btsctrl: Starting IPBTS...
Jan  1 00:00:13 btsctrl: LogLevel from conf file =6
Jan  1 00:00:13 btsctrl: IpAddressLapdif1 from conf file =10.112.81.113
Jan  1 00:00:13 btsctrl: Port no of Lapdif from conf file =9055
Jan  1 00:00:13 btsctrl: Port no RTPSignallingPortNo2 from conf file =
9066
Jan  1 00:00:13 btsctrl: Duration of iDurationLapdif from conf file =10
Jan  1 00:00:13 btsctrl: AUTO Discovery from conf file =4
Jan  1 00:00:13 btsctrl: sIpAddressRTP from conf file =123.56.234.24
Jan  1 00:00:13 btsctrl: Port no iPortNoRTP from conf file =5002
Jan  1 00:00:13 btsctrl: iAbisTS from conf file =1
Jan  1 00:00:13 btsctrl: Wait To Kill Process from conf file =1
Jan  1 00:00:13 btsctrl: No Of Wait Cycle from conf file =1
Jan  1 00:00:13 btsctrl: IndigoBts.conf file read successfully
Jan  1 00:00:13 btsctrl: InitializeBTSCTRL: entering initialization
function
Jan  1 00:00:13 btsctrl: g_BtsCtrlInfo.iBtsCtrlPid= 749
Jan  1 00:00:13 btsctrl: GetProcessStatus: entering GetProcessStatus
function
Jan  1 00:00:13 btsctrl: ReadProcessStatus: entering ReadProcessStatus
function
Jan  1 00:00:13 btsctrl: ReadProcessStatus: leaving ReadProcessStatus
function
Jan  1 00:00:13 btsctrl: GetProcessStatus: leaving GetProcessStatus
function
Jan  1 00:00:13 btsctrl: ResetValues: entering ResetValues function
Jan  1 00:00:13 btsctrl: ResetValues: leaving ResetValues function
Jan  1 00:00:13 btsctrl: CleanMsgQs: entering CleanMsgQs function
Jan  1 00:00:13 btsctrl: All Msg Qs deleted
Jan  1 00:00:13 btsctrl: CleanMsgQs: leaving CleanMsgQs function
Jan  1 00:00:13 btsctrl: InitMsgQ: entering InitMsgQ function
Jan  1 00:00:13 btsctrl: InitMsgQ: leaving InitMsgQ function

```

Figure.8 IP BTS telecom application logging

6. CONCLUSIONS

This paper describes the process of building linux kernel image and booting linux image from u-boot loader on the target board based on PowerPC. This paper also guides and helps you to familiar with the whole development flow of embedded system for telecom applications. Porting linux operating system to a new hardware board works like an embedded system doesn't need to be hard but the whole development process is some sort of complicated, so it costs much more time to master the source code of kernel for different architectures and whole development flow. In the future this embedded system performance is evaluated [20], analysed and can be tuned to design a customized embedded system for any other telecom application.

ACKNOWLEDGEMENTS

Thanks to Wireless & Embedded System Group, HOD & faculty of ECE Department, VJIT for their suggestions and encouragement.

REFERENCES

- [1] Robert Love, (2005) "Kernel Development", Pearson Education, USA, pp 11-21.
- [2] Lennon, A., (2001) "Embedding Linux", *IEE Review*, Vol. 47, No. 3, pp 33 - 37.
- [3] <http://www.bitshrine.org/gpp/>
- [4] Karim Yaghmour, (2003) "Building Embedded Linux Systems", O'Reilly & Associates, pp100-302.
- [5] DENX Embedded Linux Development Kit available on <http://www.denx.de/wiki/bin/view/DULG/ELDK>
- [6] ZHOU Qingguo, YAO Qi, LI Chanjuan & Hu Bin, (2009) "Port Embedded Linux to XUP Virtex-II Pro Development Board", IT in Medicine & Education, IEEE International Symposium (ITIME), Vol. 1, pp 165 – 169.
- [7] Hu Jie & Zhang Gen-bao, (2010) "Research Transplanting Method of Embedded Linux Kernel Based on ARM Platform", *Information Science and Management Engineering, International Conference*, Vol. 2, pp 35-38.
- [8] Sun Yanpeng Peng Peng Zhang Yuan, (2009) "Linux Transplantation Based on The Processor S3C2440", *Electronic Measurement & Instruments, ICEMI '09. 9th International Conference*, Vol. 2, pp 306-309.
- [9] Ahmed MF. & Gokhale SS, (2009) "Reliable Operating Systems: Overview and Techniques", *IETE Tech Rev*, Vol. 26, pp 461-469.
- [10] Song Kai & Yan Liping, (2009) "Improvement of Real-Time Performance of Linux 2.6 Kernel for Embedded Application", *Computer Science-Technology and Applications, IFCSTA* , Vol.2, pp 71-74.
- [11] "The PowerPC Architecture: A Specification for a New Family of RISC Processors", Second Edition, by International Business Machines, Inc.
- [12] Craig Hollabaugh, (2002) "Embedded Linux: Hardware, Software, and Interfacing", Addison Wesley Professional, pp100-432.
- [13] <http://www.kegel.com/crosstool/>
- [14] MPC8548E PowerQUICC III™ Integrated Host Processor Family Reference Manual.
- [15] Prevotet J.C, Benkhelifa A, Granado B, Huck E, Miramond B, Verdier F, Chillet D, & Pillement S, (2008) " A Framework for the Exploration of RTOS Dedicated to the Management of Hardware Reconfigurable Resources", *Reconfigurable Computing and FPGAs*, pp 61-66.
- [16] Hessel F, da Rosa V.M, Reis I.M, Planner R, Marcon C.A.M & Susin, (2004) "Abstract RTOS modelling for embedded systems", *Rapid System Prototyping, 15th IEEE International Workshop* Vol. 10, pp 210-216.

International Journal of Software Engineering & Applications (IJSEA), Vol.2, No.4, October 2011

- [17] John Oliver, Ravishankar Rao, Diana Franklin, Frederic T. Chong & Venkatesh Akella, (2006) "Synchrosalar: Evaluation of an embedded, multi-core architecture for media applications", *Journal of Embedded Computing*, Vol. 2, pp157-166.
- [18] JOHN A. STANKOVIC, (1996) "Real-Time and Embedded Systems", *ACM Computing Surveys*, Vol. 28.
- [19] Chun-yue Bi, Yun-peng Liu, Ren-fang Wang, (2010) "Research of key technologies for embedded Linux based on ARM", *Computer Application and System Modeling (ICCASM), 2010 International Conference*, Vol. 8, pp 373 – 378.
- [20] Apostolos N. Meliones, Stergios D. Spanos, (2006) "Performance Analysis of Embedded Linux ATM for MPC8260 and Derivatives", *Computers and Communications, in: Proceedings 11th IEEE Symposium*, pp 101 – 108

Authors

M.Rajendra Prasad obtained his B.E and M.E Electronics and Communication Engineering and Digital systems from Osmania University, Hyderabad. He has 11 years of experience in embedded and telecom research and development. He is currently working as Associate Professor, **ECE Department, VJIT, Hyderabad**. His main research interests are embedded systems, wireless protocols and RTOS. Email:rajendraresearch@gmail.com



S.Ramasubba Reddy is working as Asst.professor in Department of Electronics and communication engineering for CBIT, Proddatur. He completed M.Tech (Communications and Signal Processing) from S.K University, Ananthapur. He completed B.Tech Degree in E.C.E from J.N.T.U Hyderabad. His area of research includes Digital Signal Processing, Digital Image Processing and Embedded Systems. Email: ramasubbareddy.s@gmail.com



V. Sridhar completed his M.Tech in wireless and mobile communications from JNT University, Hyderabad. Presently he is working as a Assistant Professor in ECE Dept, VJIT Hyderabad. His research interests are Wireless and Mobile communications, Digital signal and image processing, Email: varadasri@gmail.com

