Bringing openPOWERLINK to MIPS (Loongson 2F)

Minqiang Yang, Chanjuan Li, Qingguo Zhou, Nicholas Mc Guire
*Distributed & Embedded System Lab, SISE, Lanzhou University, China
Tianshui South Road 222,Lanzhou,P.R.China
mqyoung@gmail.com

Abstract

Ethernet is not exactly known as a fieldbus - but it is a reliable, fast and inexpensive technology and openPOWERLINK has a strong potential to turn Ethernet into a full featured field bus for the automation industry. At the same time, due to Loongson 2F's low power-consumption, fanless operation, availability of RT-PREEMPT based on GNU/Linux RTOS and integration in form-factors suitable for industrial usage, we have been working on porting openPOWERLINK to the Loongson 2F. In this paper, we will state the motivation for the port to Loongson 2F and concentrate on key technical issues of porting Ethernet driver for openPOWERLINK, which hopefully will aid others in similar tasks. The problems we faced, some possible problems to be aware of and some tips how to tackle the porting task should provide some initial guidance for a openPOWERLINK porting job. Further we will present some benchmark results based on RT-PTREEMPT comparing the result on different platforms(Loongson and VIA C3) and a short outlook on future work in this area.

Keywords: openPOWERLINK, Loongson, Real Time, Ethernet fieldbus, RTL8169

1 Introduction

The POWERLINK homepage [1] states that *"POWERLINK is CANopen over Ethernet"* but actually it is more than that. The statement fails to cover the real-time features of POWERLINK. Furthermore the International Electrotechnical Commission (IEC) has accepted Ethernet POWERLINK (EPL)[2] as a Publicly Available Specification (PAS).

OpenPOWERLINK is an open source Industrial Ethernet solution for EPL. EPL, which operates with standard Ethernet controllers, gives users Ethernet's flexibility and its characteristic benefits while achieving cycle times as small as 0.5 milliseconds in this Open Source implementation while ensuring high synchronicity. Supported by co-processors, POWERLINK even ensures cycle times of 0.1 ms.

The Loongson CPU which is designed by the Institute of Computing Technology (ICT), at the Chinese Academy of Science is a family of 64-bit, superscalar, low power MIPS processors designed to address all applications requiring high level of performance and reduced power consumption.

The Fuloong(2F) Mini PC made by Lemote are based on the 2nd generation of Loongson CPUs. The reason why we see the Loogson 2F's as the perfect platform for openPOWERLINK are it's low power-consumption, fanless operation, the availability of RT-PREEMPT based GNU/Linux RTOS as well as the integration in form-factors suitable for industrial usage. Furthermore, due to the collaboration between STMicroelectronics and Loongson, more and more consumers can be expected to chose Loongson boards for their applications. Therefore we have been working on porting openPOWERLINK to the Loongson 2F to extend its application in the field of real-time field-bus.

In the following sections, we will state the motivation for the port to Loongson 2F and concentrate on key technical issues of our work and problems we

*Research supported by National Natural Science Foundation of China under Grant No.60973137; Gansu International Sci.&Tech. Cooperation under Grant No.090WCGA891; the National HighTechnology Research and Development Program("863" Program) of China under grant No.2009AA01A138

faced during our work. Furthermore we will address possible future problems, show off some benchmarks we conducted and finally give a short outlook on future work.

2 Motivation

Due to the features of EPL and Loongson, we decided that it is to bring openPOWERLINK to Loongson. Our work will generate a profile of the performance of openPOWERLINK which will provide some valuable information for people who want to use openPOWERLINK on MIPS-based embedded platforms. Furthermore we intend to demonstrate the suitability of Loongson based platforms for automation application.

2.1 Ethernet-based fieldbus - EPL

Bringing together Ethernet, CANopen, and a newly developed stack for real-time data communication, POWERLINK integrates features and abilities from three different worlds. In contrast to a number of competing products, POWERLINK keeps very close to the Ethernet standard, retaining original Ethernet features, and thus reducing the cost of industrial deployment. It expands Ethernet with a mixed Polling and Time slicing mechanism named SCNM (Slot Communication Network Management, refer to Figure 1).

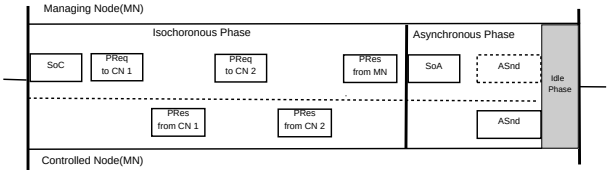

Fig. 1: *EPL Cycle*

The major advantages gained from using POWERLINK are:[3]

- Guaranteed transfer of time-critical data in very short isochronic cycles with a configurable response time
- Time-synchronisation of all nodes in the network with very high precision of submicroseconds
- Transmission of less time critical data in a reserved asynchronous channel

The protocol's second major advantage is the integration of the CANopen technology, a robust and proven protocol widely used throughout the automation world, which greatly simplifies setting up networks because of its extensive standardization. CANopen is one of the most popular higher layer protocols for CAN-based networks. Therefore there are a number of device and application profiles under development or already available which are used in example in building related applications like door control or lifts, for ships, trains, municipal vehicles or railways as well as for medical applications. Besides these standardized profiles, another big advantage of CANopen is it's use in a wide range of proprietary systems and applications.

The EPL object dictionary is identical to the structure of CANopen an object dictionary. Therefore, all CANopen application and device profiles can be directly used with EPL. The only exception is the index range between 1000h and 1FFFh which does not contain CANopen related data but EPL related data.[4] The third advantage of POWERLINK's success rests on technologies developed by the EPSG, and primarily on the POWERLINK stack, which adds real-time capabilities to the protocol.

The above listed properties of openPOWERLINK are the reason why we think that it has a strong potential to turn Ethernet into a full featured field bus for industrial automation and promote and push Ethernet as such.

2.2 Why Loongson is ready for industrial automation.

For those readers who are not familiar with the Loongson 2F CPU, here is a short list of main features:

- Scalable CPU frequency, up to 1GHz clock frequency
- 64-bit superscalar architecture
- Best-in-class power consumption: 4 W @ 1 GHz TDP (thermal design power)
- On-chip DDR2 memory controller
- Based on MIPS, a 64-bit RISC architecture
- availability of RT-PREEMPT based GNU/Linux RTOS

Loongson processors maximize the performance per dollar and per watt, and are therefore the perfect choice for price- and power-sensitive computers, thin clients, home media center devices and green industry. Currently, the Fuloong mini-PC's are mainly used in education and rural-informatization.

The properties of Loongson CPU's described above is extended by the versatility of the Linux OS

which is a reliable and secure OSS that greatly reduces system cost, supports by a rich set of software development tools and also can be configured as a real-time operating system using the RT-PREEMPT patch. Using RT_PREEMPT on Loongson 2F you are provided with a performant real-time operating system. Performance tests proving this point can be found in [8]. Relative benchmark between Loongson 2F and VIA C3 have shown that the Loongson 2F has less jitter, and latencies are smaller.

Another very nice feature of Lemote's Loongson 2F is it's integration in a form-factor suitable for industrial usage. This makes the platform ready for it's usage in the industry.

3 EPL Stack

The Standard Ethernet Data Link Layer of Ethernet POWERLINK is extended by an additional bus scheduling mechanism which makes sure, that at a time only one node is accessing the network. The schedule is divided into an isochronous phase and an asynchronous phase. During the isochronous phase, time-critical data is transfered, while the asynchronous phase provides bandwidth for the transmission of non time-critical data. The Managing Node (MN for short) grants access to the physical medium via dedicated poll request messages. As a result, only one single node (Controlled Node, CN for short) has access to the network at a time, which avoids collisions, usually present on Standard Ethernet. The CSMA/CD mechanism of standard Ethernet, which causes non-deterministic Ethernet behavior, is deactivated by the collision avoidance mechanism of the Ethernet POWERLINK scheduling mechanism.

EPL provides a virtual Ethernet driver interface to Linux which will send its packets during SoA[1] phase, as well as configure the interface with standard Linux commands.

The abstract model of EPL is illustrated as the following figure 2 [15].

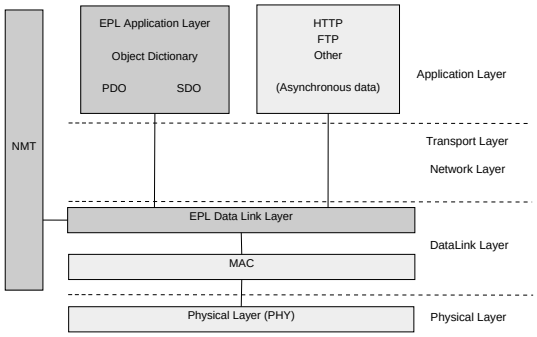

Fig. 2: *Abstract representation of the EPL OSI model*

As shown in figure 2, EPL is a protocol residing on top of the standard IEEE 802.3 MAC layer, which has its own Data Link Layer using SCNM instead of CSMA/CD. The virtual Ethernet driver provides a TCP/IP stack which can be used to process non-time-critical data, while time-critical-data only can be handled by the EPL stack directly.

4 Implementation

The EPL protocol stack core components are target independent, porting to a new platform requires you to implement the Ethernet driver for EPL, modify target dependent files for architecture(e.g. timer), configure your own object dictionary and adapt the API. For further information about the EPL stack please refer to Ethernet POWERLINK Protocol Stack[17].

Since the generic parts of the stack did not require any changes, we will mainly describe our work on writing an Ethernet driver for EPL. The following functions needed to be modified to adapt to our new NIC(network interface card).

Hardware-dependent functions of EPL driver:

```
/* Hardware initialization */
EdrvInitOne()
/* Resource giveup */
EdrvRemoveOne()
/* Packet transmit*/
EdrvSendTxMsg()
/* Request tx buffer */
EdrvAllocTxMsgBuffer()
/* Release tx buffer */
EdrvReleaseTxMsgBuffer()
/* Interrupt handler */
TgtEthIsr()
```

4.1 Driver for RTL8169

The Realtek RTL8110SC(L) Ethernet adapter built into the Fuloong 2f, is a highly integrated, high performance PCI Gigabit Ethernet Media Access Controller. In this paper, it is referred to as RTL8169 or NIC provides the following features:

- Fully implements the 33/66MHz, 32/64-bit PCI v2.2 bus interface
- Compliant with the IEEE802.3 specification for 10/100 Mbps Ethernet and IEEE 802.3z for 1000Mbps
- Compliant to Microsoft NDIS5 (IP, TCP, UDP) Checksum and segmentation Task-offload features, and supports IEEE802.1Q VLAN

[1]EPL frame types, SoA: Start of Asynchronous, SoC: Start of Cyclic, PReq: Poll Request

- Pad any packets less than 64 bytes automatically

The RTL8110SC(L)/RTL8169SC(L) supports a new descriptor-based buffer management, utilizing 3 descriptor rings. One is high priority transmission descriptor ring, the second is a normal priority transmission descriptor ring and the last one is the receive descriptor ring. Each descriptor ring may consist of up to 1024, 4-double-word consecutive descriptors. By writing a '1' to a specific bit of Transmit Polling Register, driver notifies the NIC that there are packet(s) waiting to be transmitted, the NIC will automatically transmit the packets in the buffer. The NIC clears the Transmit Polling Register bit after all the packets in the buffer have been transmitted, and generates an TOK(Transmit OK) interrupt after each successfully transmitted packet.

Upon receiving a packet over the wire, the NIC copies it to the Rx Buffer Manager, and notifies the driver via a ROK(Receive OK) interrupt.

In the following we listed some of the most important flags used to handle buffer management, in order to clarify this descriptor-based buffer management mechanism. They are split up into fields/flags used to transmit/receive data:

4.1.1 Transmit Descriptor

- OWN - This bit - when set - indicates that the data relative to this descriptor is ready to be transmitted. The NIC clears this bit when the relative buffer data is transmitted.
- EOR - This bit indicates whether this is the last descriptor in descriptor ring.
- FS,LS - These two bits indicate this is the first/last segment of the packet.
- IPCS, UDPCS, TCPCS - The driver sets this bits to ask the NIC to offload the checksum.
- Frame_Length - Transmit frame length.
- TxBuffL - Low 32-bit address of transmit buffer.
- TxBiffH - High 32-bit address of transmit buffer.

4.1.2 Receive Descriptor

- OWN - This bit - when set - indicates that the descriptor is ready to receive a packet. It will be cleared by the driver when the relative data has been read.
- EOR - This bit indicates whether this is the last descriptor in descriptor ring.
- Buffer_Size - This field indicate the receive buffer in bytes.
- RxBuffL - Low 32-bit address of receive buffer.
- RxBuffH - High 32-bit address of receive buffer.

4.2 NIC Driver

The EPL driver for RTL8169SC(L) mainly includes initialization, allocating transmit and receive buffer rings and mapping, interrupt handling, and other related functions (allocate buffer/send/error handle etc). All those parts will be described in detail in the following.

4.2.1 Initialization

Like other drivers in Linux, the EPL driver needs to register a PCI driver, enable it and install an interrupt handler for it. Furthermore the registers of the NIC have to be initialized.

4.2.2 Allocate DMA Ring Buffer

As described before, the RTL8169 supports a descriptor-based transmit and receive, we need to allocate buffer rings for transmitting and receiving and write the start address of descriptor array to Descriptor Registers. The start address of each descriptor group should be 256-byte alignment. Descriptors can be chained to form a packet in both Tx and Rx.

The Tx Buffer Manager DMAs packet data from system memory and places it in the transmit FIFO of the NIC, and pulls data from the FIFO to send to the Tx MAC. The Rx Buffer Manager uses the same buffer management scheme as used for transmits.

We must use physically contiguous memory which will be accessed by the DMA device on a physically addressed bus, so be sure you allocate the buffer by *kmalloc()* with flag GFP_DMA rather than *vmalloc()*.

4.2.3 Transmit

RTL8169 will poll the transmit descriptor ring automatically after the specific bit of the TPPolling Register has been set. Take note of the caching effect of descriptor ring, while you could prevent from it by using *pci_alloc_consistent()* which allocates a region of consistent memory, and you could freely read or write the region without having to worry about caching effects. Pay special attention to the fact that the RTL8169SC sequentially polls the descriptor from the first item to the last, but EPL needs

to reserve several buffers for frequently transmitting packets(SoA, SoC, PReq, etc.). The first packet EPL sends may not be in the first buffer. So you can not associate the descriptors and the buffers in sequential order when initializing, but you will have to assign a buffer to the descriptor while sending.

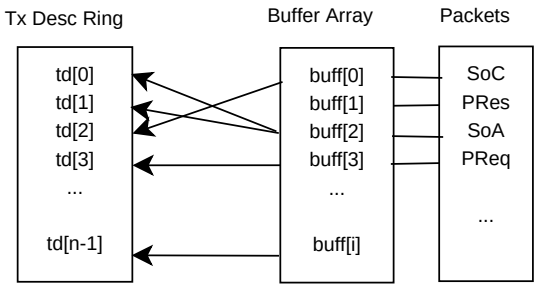

Fig. 3: *DMA Transmitting*

The figure shows that how our driver handles the descriptors and DMA buffers to send packets with the following sequence:

SoA - SoA - SoC - PReq

4.2.4 Interrupt handler and receive events

The EPL driver is not able to work in NAPI[12] mode. Therefore its interrupt handler must handle all the interrupts that are part of the driver, such as transmit interrupts, receive interrupts etc.

The driver will call the handlers of the data link layer when both ROK and TOK interrupt occur. In the interrupt handler, you may also need to consider error handling. When a packet is received successfully, it will be handed to the upper layer of the stack.

4.3 Tools

- GDB tracepoint[13]
  As a powerful debug tool, GDB tracepoint could observe the behavior of your real-time program whose correctness depends on its real-time behavior, delays introduced by a debugger might cause the program to change its behavior drastically, or perhaps fail, even when the code itself is correct. Please refer to GTP[14].

- Wireshark
  For diagnostics using Wireshark, the computer must be connected to the POWERLINK network. You may need an additional NIC which must not send packet. Be careful with the timestamps in Wireshark. Timestamping is done by network drivers[2]. The accuracy of the timestamp is depend on the working mechanism (e.g. if it works in NAPI mode) of your network driver and some delays (There's a delay between the arrival of that last bit and the interrupt for the packet and a delay between the start of interrupt handling and the point in the code path where the timestamp is attached to the packet).

- ftrace
  Ftrace is an internal tracer designed to help out developers and designers of systems to find what is going on inside the kernel. It can be used for debugging or analyzing latencies and performance issues of your program.[10]

5 Benchmark

The jitter[3] of SoC makes a great impact on the real-time performance of EPL. To benchmark the performance of EPL on Loongson2F, the timestamps of SoC packets are recorded(utilizing a typical multi-node setup). As a comparison, we also do the benchmark on VIA3C. The result shows that the jitter on VIA3C and Loongson2F under no-load condition are almost the same. While under overload condition, EPL on Loongson2F has less jitter but bigger average deviation.

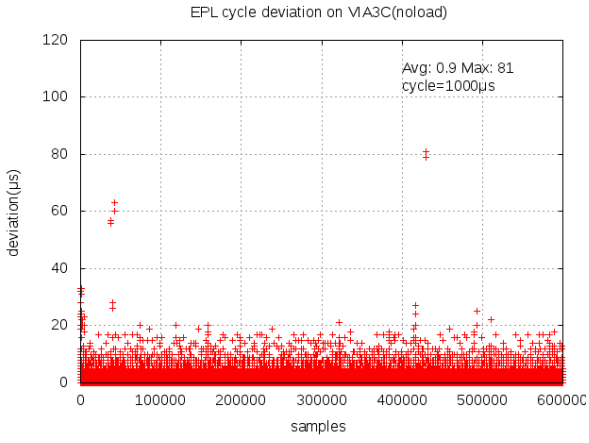

Fig. 4: *Jitter on VIA3C(no-load)*

[2]How the timestamp works is OS dependent. In some UNIXes that code is in the network drivers; it's higher up in the networking code path in other UNIXes. In Windows, with WinPcap, timestamping is done by the WinPcap driver. Here we are referring to UNIX like OSes.

[3]The variation of the latency is called jitter. But here the jitter refers to the variation of the deviation of communication cycle.

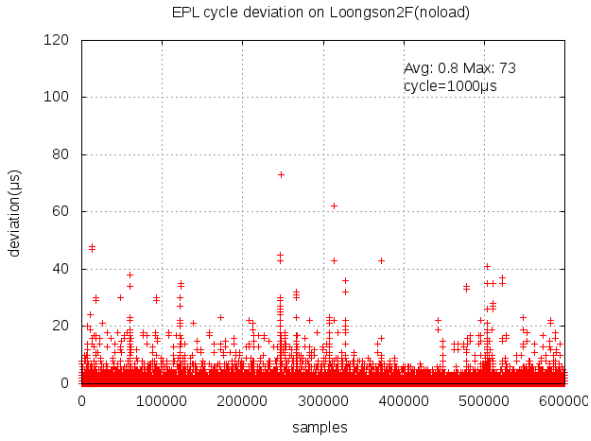

Fig. 5: *Jitter on Loongson2F(no-load)*

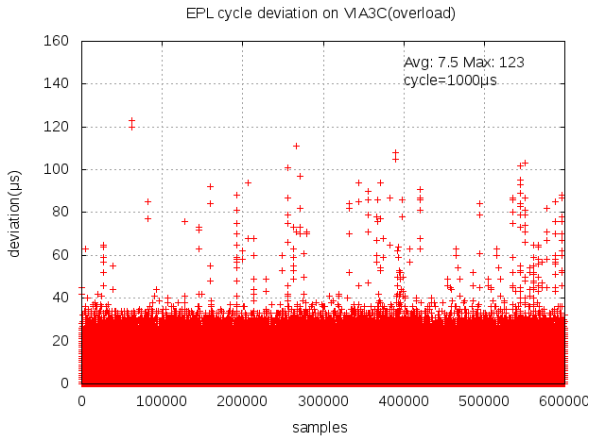

Fig. 6: *Jitter on VIA3C(overload)*

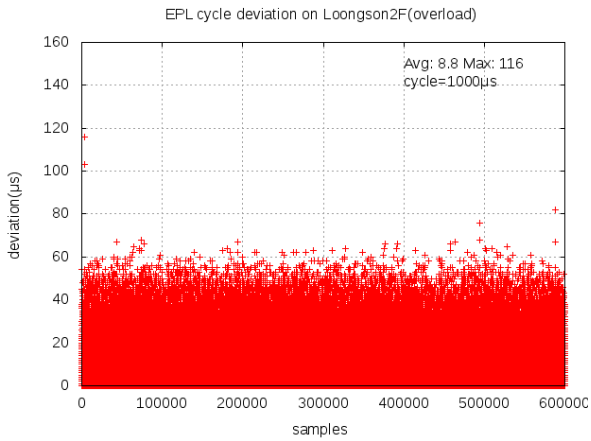

Fig. 7: *Jitter on Loongson2F(overload)*

Another point worthy of our attention is the time-drift on Loongson2F is notable smaller than it on VIA3C. This provides more space to optimization to EPL on Loongson.

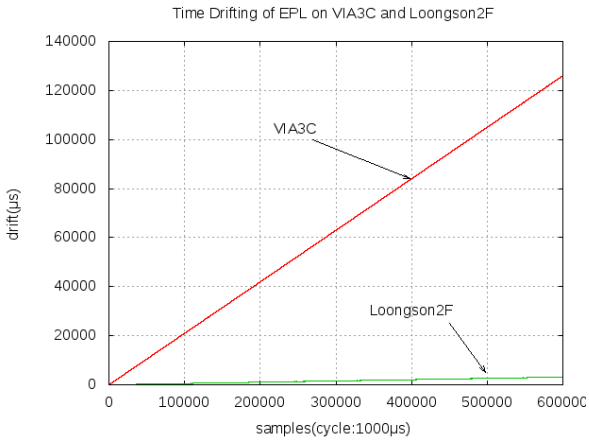

Fig. 8: *Time drift*

6 Conclusion

In this paper, we showed a case study for porting operPOWERLINK to Loongson2F, in which the most important work was porting EPL driver to the new hardware. The benchmark results show that the new drivers performance is comparable to the original 8139 driver.

To conclude the paper, we are now going to summarize the problems we faced, some possible problems that might arise in the next steps to be taken and give some hints on how to tackle the porting task to provide some initial guidance for a openPOWERLINK porting job.

- Ethernet adapter
  Be careful of the descriptor based transmission, as described above RTL8169 always polls the data from the first transmit buffer after it is driven and then the next one, but EPL never observes this order. Regarding the speciality of EPL, to put the NIC into promiscuous mode.

- Memory issues
  Make sure that Tx/Rx DMA buffers are allocated correctly, and refer to the datasheet of the NIC and follow its alignment requirement. If necessary you might use memory barriers, but be careful since incorrect use might introduce excrescent latency.

- Kernel driver
  Nowadays NIC drivers in kernel use the NAPI mode and many data structure EPL can not use, nevertheless we could reuse the code by removing their corresponding code. You could

even neglect the configuration for PHY registers in the beginning of your porting.

- Others tips
  Follow the guides of EPL and RT-PREEMPT to configure your kernel, and make sure your rt-kernel works before you get down to work on EPL. A hub rather than a switch should be used to connect EPL nodes[5]. Be sure you separate EPL nodes and your other working station.

At the moment the RT-PREEMPT patch for the network protocol is being optimized. After this task has been completed, we will be able to further improve the performance of EPL over RT-PREEMPT on Loongson to further improve the performance and all for even stricter real-time requirements.

References

[1] http://www.ethernet-powerlink.org/

[2] http://openpowerlink.sourceforge.net/

[3] Ethernet POWERLINK, http://en.wikipedia.org/wiki/Ethernet_Powerlink

[4] Ch.Schlegel, Extending CANopen Applications with ETHERNET Powerlink, 2007-09-24

[5] EPSG, Ethernet POWERLINK V2.0 Communication Profile Specification, Version 1.0.0

[6] RealTek, RTL8169SC/RTL8111SC(L) REGISTERS DATASHEET, 2005, Rev. 0.9

[7] RT-preempt, http://rt.wiki.kernel.org/

[8] Porting RT-PREEMPT to loongson 2F, Zhangjin Wu, Nicholas Mc Guire, 2009

[9] http://dev.lemote.com/code/rt4ls

[10] ftrace, Documentation/trace/ftrace.txt

[11] http://rt.wiki.kernel.org/index.php/Cyclictest

[12] NAPI,http://www.linuxfoundation.org/ collaborate/workgroups/networking/napi

[13] GDB tracepoint, http://sourceware.org/gdb/current/onlinedocs/gdb/Tracepoints.html#Tracepoints

[14] GTP, http://dslab.lzu.edu.cn/modules/lifetype/index.php?op=ViewArticle&articleId=31&blogId=22

[15] Bo Hansen et al., Ethernet Powerlink in Formula Student racing car

[16] Understanding the LINUX KERNEL 3sd Edition, Daniel P.Bovet & Marco Cesati, O'REILLY, 2005

[17] openPOWERLINK: Ethernet POWERLINK Protocol Stack, 2010