An Open Implementation of Profibus DP

TRAN Duy Khanh, Pavel PISA, Petr SMOLIK
Czech Technical University in Prague, Faculty of Electrical Engineering
Department of Control Engineering
Karlovo namesti 13, 121 35, Praha 2
info@pbmaster.org
http://www.pbmaster.org

Abstract

This paper presents a project named PBMaster, which provides an open implementation of the **Profibus DP** (*Process Field Bus Decentralized Peripherals*). The project implements a **software implementation** of this very popular fieldbus used in factory automation. Most Profibus solutions, especially those implementing the master station, are based on ASICs, which require bespoke hardware to be built solely for the purpose of Profibus from the outset. Conversely, this software implementation can run on a wide range of hardware, where the UART and RS-485 standards are present.

1 Motivation

Profibus is a fieldbus that can be used both in production automation and process automation and which has become a global market leader. Worldwide, over **28 million Profibus devices** were installed by the end of 2008.

Although the Profibus was initially standardized in the late 1980s it is not easy to find materials to help design and system engineers develop new products. Despite many interesting features like deterministic media access or fast data exchange, Profibus still seems to be only in the domain of professional applications and commercial solutions. The main reason is probably due to the high price of all Profibus products, whether hardware or software solutions.

Commercial solutions commonly need special hardware, and the software is mostly proprietary - which also significantly increases the price of Profibus applications. In addition, the primary supported platform for these solutions is the operating system MS Windows. Support for Unix-like systems is very low.

2 PBMaster project

Into this marketplace, **PBMaster** [1] comes with its solution within the field of this popular industrial bus. The project is still under extensive development, but aims to offer a cheap solution that will make it possible to use Profibus not only in commercial applications, but also in universities, homes or semi-professional applications. The key to achieving these objectives is in using common **inexpensive hardware** and **open source software**.

Initially, the project aimed to offer a solution for connecting common personal computers running Linux to this industrial bus. Now, it runs on several operating systems (**Linux**, **FreeBSD** and **NetBSD**) and sys-less embedded hardware based on the **ARM** architecture. The objectives are to offer multi-platform drivers, libraries and applications capable of carrying out the master, slave and analyzer functions of the Profibus. In the future the project will try to offer a complex and inexpensive software based solution for applications using Profibus.

The following picture describes the software structure of the PBMaster project. We will take a closer look at each component later in this paper.

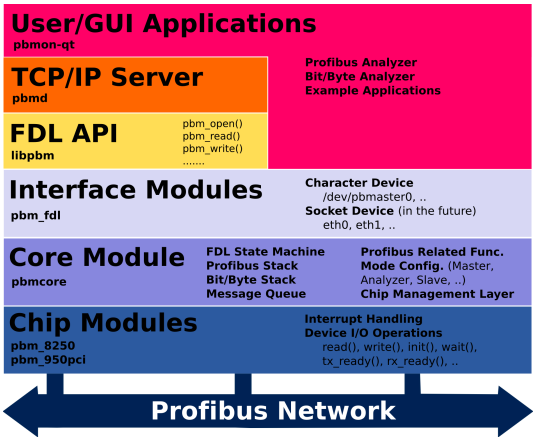

FIGURE 1: *Software structure*

3 Open source vs. Profibus patents

The project was established to offer **an open implementation** with all the components to be released under the GNU GPL v.2 or a later version. Earlier this year **distribution of the source code was been suspended** due to patent problems preventing the distribution of the implementation under an open-source license. The patents appeared this year even though they were applied in 1992. Two of the three patents related to the Profibus implementation affect the FDL master implementation within the PBMaster project.

Despite very positive feedback from the Profibus experts after the presentation of the project at the Profibus Conference in Krakow in July 2009, the PI (*Profibus International organization*) and patent holders declared they did not want to support the community and repeated that **Profibus is an "open" but not "open-source" standard**. The PI grants rights for using the patents to Profibus members. This means the **PI members can use** PBMaster's implementation. The situation unfortunately does not allow distribution of the master implementation, which covers a significant part of the project, in the community until the patents are valid.

4 Hardware System Structure

As mentioned before, implementing the standard by software makes it possible to run on a wide range of hardware including that which was not designed for the purpose of Profibus. It is also one of the means by which the solution can be made inexpensive. The software implementation can run on hardware integrating UART (*Universal Asynchronous Receiver/Transceiver*) circuits with RS-485 output, the physical standard used by Profibus DP. The current version supports three types of hardware. We will discuss them in the following sections.

4.1 RS-232/RS-485 Converter

The converter offers an economic solution in order to connect a PC to a Profibus network. The essential requirement is for the connected PC to have a serial port, which could be a problem as modern computers lack this communication port. The electric level of RS-232, generated by the common PC, is converted using this dongle to RS-485, used by Profibus DP. The dongle is powered directly from the serial port of the PC. It is the reason why the converter is very small with very few components integrated. The lack of external power supply limits use of the dongle in very extensive networks with long cabling; furthermore, it's not possible to use cable termination and the number of nodes on the network is limited to around 10 nodes.

A highest attainable Profibus speed of 19200 bit/s is another limit of this converter. The limitation is caused by differences in speeds supported by RS-232 of the personal computer and the speeds supported by Profibus DP. This converter can be run in mode Master and Analyzers. It is a very low cost solution. All schematic and assembly materials are available in the project's repository.

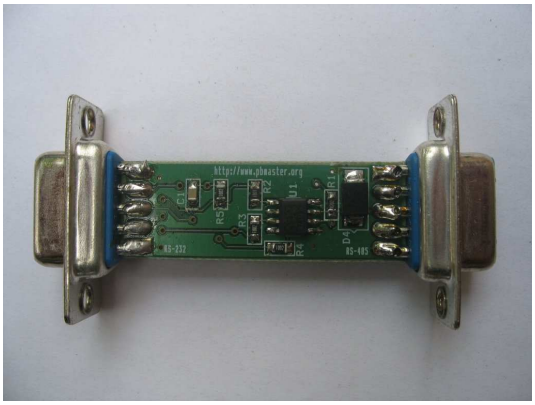

FIGURE 2: *RS-232/RS-485 Converter*

4.2 PCI based cards

In order to achieve higher speeds it is necessary to use some UART integrated extension cards. The current drivers support PCI based cards integrating the OX16C954 chip, probably the fastest UART integrated circuit with a PCI interface on the market. It is a high performance UART with 128 byte FIFOs by Oxford Semiconductor. The IC integrates a 16C550 compatible UART offering an ample FIFO size and many other interesting features. One of the most important features is the maximal bus speed of 15 Mbit/s in normal mode and 60 Mbit/s in external clock mode. Using this IC the drivers **achieve a communication speed of 12 Mbit/s**, the maximal speed of Profibus DP. The PCI card contains up to four ports. Each port can run in Master or Analyzers mode up to 12 Mbit/s independently of the others. The PCI card does not allow usage in the Bit analyzer mode at the moment as there is no built-in hardware support (could be added easily).

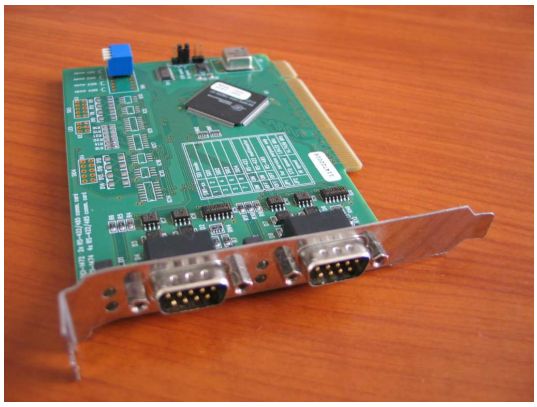

FIGURE 3: *UART Integrated PCI Card*

4.3 ARM based boards

Having the option to use the Profibus drivers on an embedded system is especially interesting. This option makes connecting to the Profibus network easy, cheap and removes the need for big hardware support. Reading sensors and controlling actuators would be very easy and available for common applications like home automation, robot controlling or even control of a production line. The embedded system offers a lot of options for creating a bridge between different standards, as the micro-controllers usually offer many types of interfaces.

The following figure shows a board integrating an LPC 2148 ARM with UART interfaces by NXP Semiconductors. The highest tested Profibus speed was 500 kbit/s (limited by the transceiver). It is expected to achieve a Profibus speed 1.5 Mbit/s or higher with a faster transceiver. The software runs on a target without operating system. It is a robust solution and can be used in Master or experimentally in Slave mode.

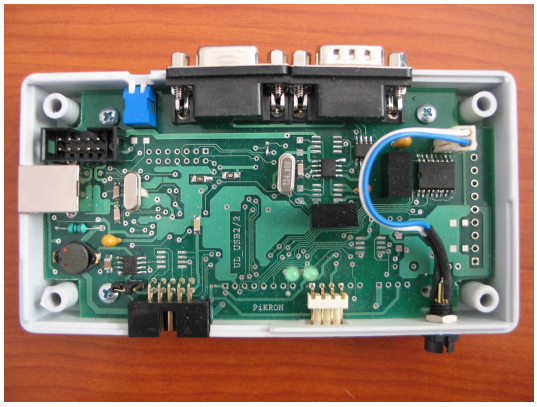

FIGURE 4: *Embedded Board Integrating an ARM LPC 2148*

This very tiny module integrating ARM7 by Atmel offers many interesting interfaces like Ethernet, USB 2.0, SPI, I2C, CAN transceiver, UART + RS-232/RS-485 transceivers etc. It runs FreeRTOS, a highly portable real-time operating system. Profibus support for this module is under development and is expected to achieve very high speeds thanks to the DMA channel and very high speed RS-485 transceiver.

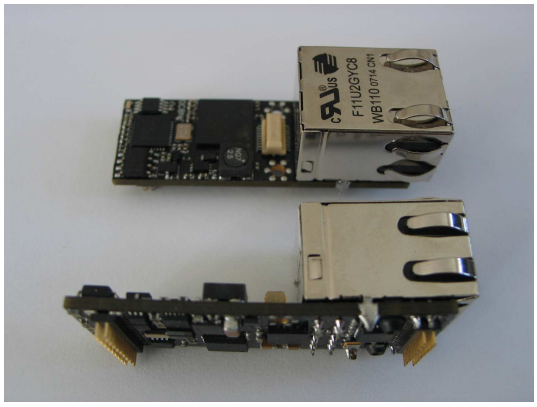

FIGURE 5: *ARM7 board running FreeRTOS*

5 Device Drivers

Almost all hardware needs software to support the functionality it was designed for. In our case, this

takes the form of a device driver implementing low-level support called FDL (*Fieldbus Data Link*). Most of the hardware supported solutions use ASICs to implement this FDL layer. In contrast, the PB-Master solution is implemented solely by software. Along with the FDL layer the drivers cover also the low-level part of the Profibus analyzer and UART Bit/Byte analyzers. The software structure was designed to be as modular as possible. The modularity allows porting to other platforms without extensive modification of the base part. The created framework offers a simple way to write driver support for new hardware.

The current driver design is divided into three groups of modules:

The core module – (*pbmcore*) – is the main driver implementing the basic device data structure, chip related defines, a bit/byte stack implementation and the most important part, the Profibus FDL stack. The implementation of Profibus covers *Finite State Machines* of an FDL Master station, FDL Slave station, low-level part of the Profibus Frame Analyzer and UART Bit/Byte Analyzer.

The module is platform and hardware independent and by itself does nothing but offer functions to other chip drivers. This allows use of several chip drivers simultaneously without any code redundancy. In addition it makes possible supporting new hardware without deep knowledge about the Profibus standard.

Chip drivers – (*pbm_950pci, pbm_8250, ..*) – implement **basic I/O operations** with real hardware like chip initialization, read, write etc. The chip module, upon loading, registers to the core module these basic operations and provides hardware and system dependent support for things such as resource allocation or interrupt registration. The remaining actions are controlled by the core module through the state machine.

User space interface module – (*pbm_fdl*) – the third group of modules implements an interface between user applications and the FDL layer in the kernel space. Currently, the module is a **character device** but in the future a **socket module** will be developed to allow communication using a socket-based technique. An application writes its request to, and reads responses from, the device. This data is passed to the core module and then is addressed correctly to an appropriate chip device. Regardless of whether the user chooses to communicate using a file or a socket, the API between an application and a kernel is unified, as well as the API between an interface module and the core module.

6 FDL API

Once the drivers are installed, communication with another station on the bus is provided by sending bytes in the **raw form** directly to the Profibus device. Even thought it is simple to realize communication in this way, the fact that a developer needs to understand frame structures, and even the Profibus specification, make it unsuitable and inconvenient for most application developers. Hence an API was developed to offer a unified and simple mechanism to access the Profibus network.

The API is provided in the form of a library. The library offers a unified system and architecture-independent programming interface handling the transfer between FDL applications and Profibus stations.

The project had a plan to fully support the FDL programming interface by Siemens. In the end, backwards support was not implemented due to the proprietary license of that programming interface. Nonetheless, the project's API partly supports backward compatibility with the Siemens's FDL programming interface. Everything related to Siemens's Request Block should be removed as the library does not work with that structure. Replace the Request Block with a buffer. This backward support is implemented as macros calling the proper API functions described previously. Mapping between the project's API and the API by Siemens is listed below:

pbm_open	**SCP_open**
pbm_close	**SCP_close**
pbm_write	**SCP_send**
pbm_read_poll	**SCP_receive**
pbm_errno	**SCP_get_errno**

7 Profibus DP

The Profibus DP (*Decentralized Peripherals*) layer is based on top of the FDL layer. There are three groups of specification for the DP layer. The basic version *DPV0* should be supported by all DP devices which involve support for Cyclic Data Exchange and Diagnostics. The *DPV1* extensions are an integral part of the Profibus PA specification providing Acyclic Data Exchange, Process Alarm Han-

dling etc. Extension *DPV2* introduces additional enhancements that are used in high-speed servos and drives and in functional safety systems. DPV0 will be supported by the project and its implementation is under development. This will be an important step for certification of the implementation.

8 TCP/IP Server

Until now, the software supported only Unix-like operating systems or embedded ARM based systems. With the server, any station supporting TCP/IP can connect to the Profibus network remotely.

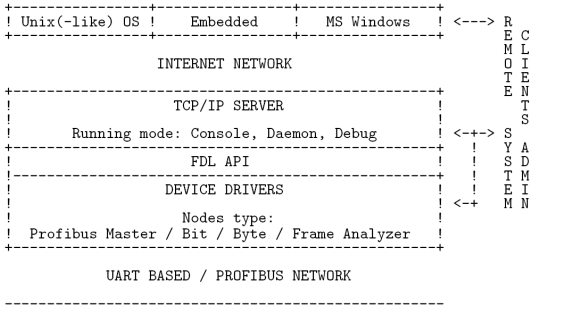

FIGURE 6: *Client-Server Architecture*

The figure illustrates a client-server model. The server is installed on the machine along with the device drivers. The device drivers implement installed nodes on the bus in several modes. The server provides access to devices created by the drivers over the Internet network. On one side it uses the FDL programming interface for accessing those devices. On the other side socket communications are established to serve the client demanding remote access to devices.

9 Bus Analyzer and Monitor

Applications in industry often demand for tracking problems on the fieldbus. The problems can be diagnosed and localized by listening to and analyzing communication on the bus. By analyzing the captured data, it is possible to detect problems like frames with errors (often caused by signal reflection), station inactivity, address collisions, too short Slot Times of the Master stations, even bad timing of stations.

This section introduces a graphical analyzer and monitoring program. The program was written in QT and as with other components of the project, was designed to be fast, reliable and modular. Thanks to its modularity, it will be more simple to integrate new features into the program so it could become a versatile industrial bus control and monitoring program. The current version supports Profibus Frame analyzer and UART Bit/Byte analyzer.

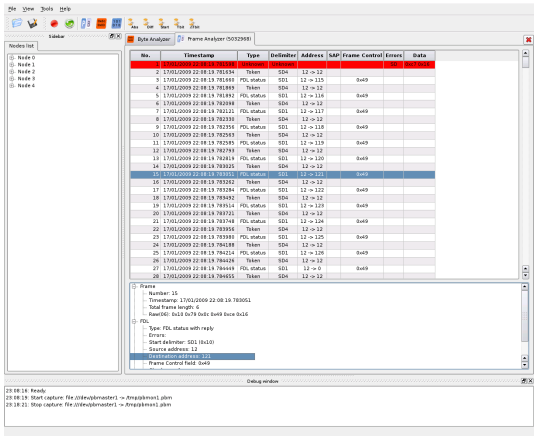

FIGURE 7: *Profibus FDL/DP Analyzer*

The main window contains *menus*, *tool bars*, *side bar*, *debug window* and *tabs* with specific functions. The global buttons like *Start Capture*, *Refresh View*, *Timestamp settings* affect only the visible tab. The debug window is collective for all tabs. The side bar shows information about the nodes. Several tabs of the same type can be present at the same time. The analyzer offers other interesting features like online view during capture, offline view of saved files and timestamp display in five formats. Timestamp can be in *absolute time*, *time difference between data*, *time from the start of capture*, *bit time difference between data* and *bit time from the start of capture*. Capture trigger, view filter and statistics of captured data will be implemented in the next version.

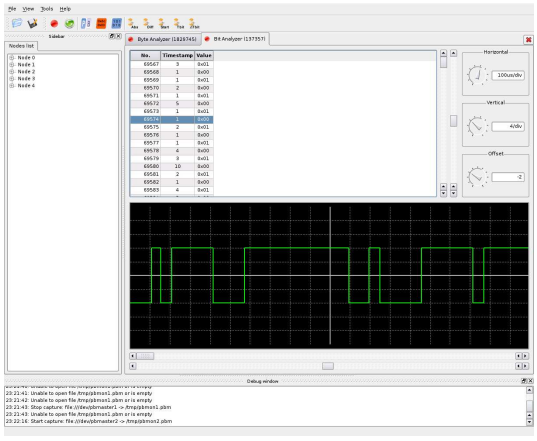

FIGURE 8: *Bit Analyzer*

All addresses are specified using the URL mechanism (Uniform Resource Locator). A URL identifier could be *pbm://server.pbmaster.org:11000/0*, *file:///dev/pbmaster0* or simply */dev/pbmaster0*. The first URL refers to a device number 0 on a remote server using port number 11000, the latter two refer to a local file.

10 Live Linux CD

The PBMaster project is an effort to develop and maintain an open-source implementation of Profibus DP. Running in open-source software has also disadvantages. Probably the biggest disadvantage is that almost everyone using computers is familiar with commercial operating systems like MS Windows or Mac OS, but not everyone has experience nor the ability to get on with Linux, BSDs, etc. Despite rapid improvements and the spreading use of open-source software, applications based on commercial software and MS Windows are still dominant and this situation applies especially for Profibus industry.

That is why the project has developed a **Live Linux CD** offering an easy way to use the Profibus solution without need for software installation. It is not necessary to install any operating system nor other program - just plug in the hardware, insert the Live CD and start working with Profibus.

The CD is based on the Debian Lenny [6] distribution. It contains utilities and useful tools like Xfce desktop environment and its programs, Firefox internet browser, Kate text editor, Openoffice.org office suite, OpenSSH client and server, GCC compiler collection and Make utilities, Midnight Commander, Vim editor, GIT, CVS, SVN, media player and many more programs including all necessary libraries to compile the project's components. The number of packages in total is more than 750.

FIGURE 9: *Xfce Desktop with Profibus Utilities*

Imagine having an environment supporting Profibus and other UART utilities in just a few minutes. There is **no need for installation**, no manual configuration required - just put the Live CD into your drive and reboot the computer.

11 An Overview of Applications

The figure shows a model production line running Profibus using PBMaster. The drivers implement a master station with an integrated PCI card, based on an OX16PCI954 chip, running at 1.5 Mbit/s. It is possible to run up to 12 Mbit/s without any problem. The speed limitation is caused by one slave station. Actuators and sensors are connected to slave stations by WAGO.

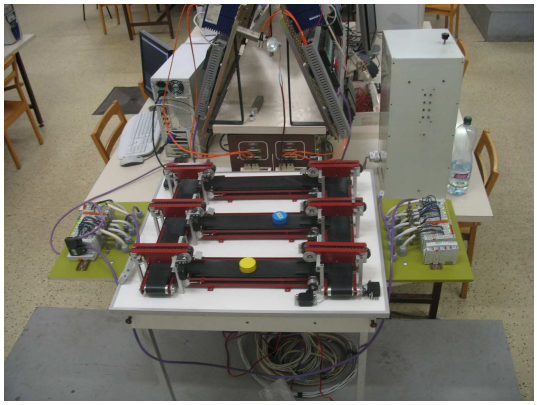

FIGURE 10: *A Model Production Line*

The figure shows several Profibus networks. One network consists of an ARM based board running in master mode, a slave station with digital inputs/outputs and an analyzer running in bit mode. The network communication speed is 500 kbit/s. The second network consists of two Profibus nodes communicating at 12 Mbit/s. One node simulates a master station, the second node runs in frame analyzer mode. The rest 4 unconnected ports are masters running at 187.5 kbit/s, 500 kbit/s, 1.5 Mbit/s and 6 Mbit/s. All nodes run simultaneously.

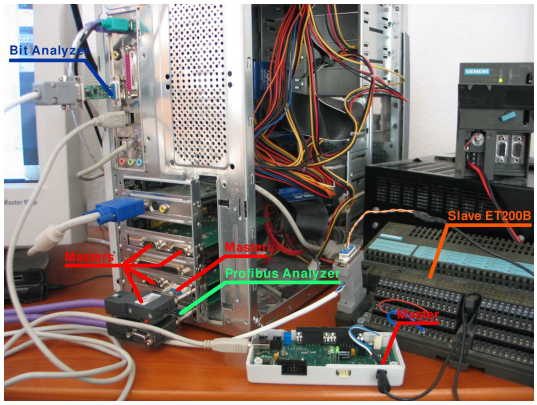

FIGURE 11: *Master-Slave Communication and Bus Monitoring*

The figure shows two Profibus networks. There is one master, one slave and one bit analyzer on the first network running at 19200 bit/s. The second network consists of one master and one frame analyzer station running at 12 Mbit/s. Data from the analyzer is sent over the Internet network to the analyzer and monitoring program running on a notebook. More examples can be found on the project's website [2].

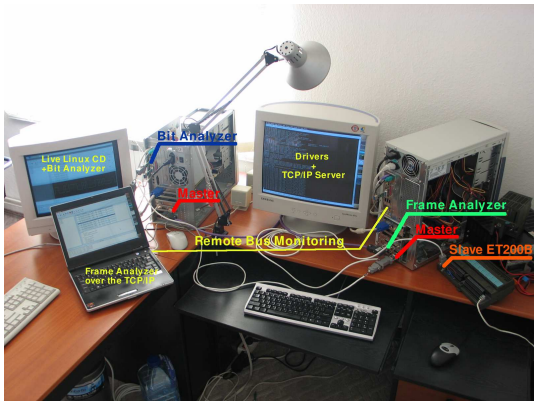

FIGURE 12: *Remote Bus Monitoring*

12 Friendly projects

I would like to mention two projects, **uLan** and **ProfiM**, which have similar objectives as PBMaster. Some solutions and ideas used in PBMaster have been taken from these projects.

The project uLan [4] provides a 9-bit multi-master message oriented communication protocol, which is transferred over the RS-485 link. Characters are transferred in the same way as for the RS-232 asynchronous transfer except for the parity bit, which is used to distinguish between data characters and protocol control information. The physical layer consists of one twisted pair of leads and RS-485 transceivers. The project is developed and maintained by Pavel Pisa and Petr Smolik. For more information please refer to the project homepage.

ProfiM [5] is a project implementing simulation of master stations using only UART integrated PCI cards or a simple converter of RS-232/RS-485. It leads to an inexpensive solution as there is no need of any expensive proprietary hardware nor software. In addition it offers an FDL programming interface compatible with the API by Siemens. The project supports only Windows OS. One of the disadvantages is that the project has been inactive since 2004. The driver was not designed to be modular and from my point of view it is inefficient and even the stability of the driver is not favorable. The project was developed by Pavel Trnka and maintained by Petr Smolik. For more information please refer to the project homepage.

13 Compiling with OMK

At present, the device drivers are built inside the source code directory using system's default make scheme. On the other hand, the project's applications are built out of the source tree using Make system called OMK (*Ocera Make System* [7]).

OMK [3] is a Make system developed and maintained by Pavel Pisa and Michal Sojka from the Department of Control Engineering at Czech Technical University in Prague under the *OCERA project*. The main objective of the OMK system is to simplify compilation of components on the host machine as well as cross compilation for the target. In addition the system provides for a better directory and file structure. The make system allows building out of source trees and storing results of the compilation in a separate directory structure to simplify testing and program installation.

A key solution is to have a central Makefile with compilation rules for most sub-components and components. This solution allows faster and smoother changes to the system, such as kernel updates. Having most rules in a central file, Makefiles in source directories can be very simple.

OMK was not designed to support the BSD systems. A number of changes have been made in order to use this make system on FreeBSD and NetBSD. Although not all OMK features are available yet under FreeBSD/NetBSD, it is possible to compile all

project's applications using this system. The OMK system is particularly useful for cross-compiling to the final architecture (e.g. embedded ARM).

14 Advantages and disadvantages

One of the advantages and a strong point of the project is based on the modular design of the software. It makes the implementation portable and multi-platform. The software achieves 12 Mbit/s with very good performance. Thanks to the software implementation, applications based on RS-485 could be switched to use the Profibus, therefore it is not necessary to invent a new protocol in the application based on this physical standard.

There are of course many disadvantages of the software implementation. Even though it works reliably in most cases, it is not possible to guarantee fully deterministic responses on common operating systems due to their design (e.g. when the CPU is occupied by handling a lot of interrupts from another devices). This could be solved by using Real-Time operating systems (tested on Linux with Real-Time Preempt patches). Another option to achieve fully deterministic responses is to use embedded boards without operating system.

This is a software implementation without hardware support. It means there is no oscilloscope mode nor speed detection by analyzing the signal. Another very important disadvantage is that this implementation is not yet certified (under development).

15 Conclusion and Future Work

An open implementation of Profibus has been presented throughout this paper. One of the stated goals for the project was the creation of multi-platform software capable of running on systems with operating systems as well as embedded boards without OS. The components were designed to facilitate porting to new platforms. The software is implemented to be fast, reliable and requiring minimal system resources.

Currently, the project offers an implementation of Profibus FDL master stations, FDL slave stations, Profibus FDL/DP frame analyzer, bit and byte analyzer for UART based bus, an FDL programming interface, Live Linux CD, a TCP/IP server for remote access, a set of examples and free Profibus documentation. The software runs on Linux, FreeBSD and NetBSD operating systems as well as ARM based embedded systems. A few open hardware solutions are also available.

There are many applications, where Profibus can help to make our life more comfortable. The Profibus can be use for example in automatic door systems, power saving systems, temperature regulation in buildings, etc. Application of this open solution may be appreciated by universities and companies, as it is currently necessary to have very expensive commercial hardware and software for any experimentation with Profibus.

Despite these achievements, there are still many problems to be solved to facilitate use in professional applications. The future objective is to implement the Profibus DPV0 layer and to create an unofficial certification of the implementation. Even though the slave abstraction layer is working, there is still a lot of work to be done to improve it. The project will extend support for new hardware and operating systems like FreeRTOS and MS Windows. Libraries, graphical analyzer and other applications will be improved.

The problems related to Profibus are very complex and much effort is required to overcome them. My hope is that this project will make more people familiar with this popular fieldbus solution, so the extent of use of the Profibus standard would not be limited only to commercial applications. The software implementation can **spread the use of the standard** to applications, where hardware supported solutions are inconvenient or not possible.

References

[1] *PBMaster Project*, Tran Duy Khanh, http://www.pbmaster.org

[2] *PBMaster - Wiki Pages*, Tran Duy Khanh, http://wiki.pbmaster.org

[3] *OMK Project*, Pavel Pisa, Michal Sojka, http://rtime.felk.cvut.cz/omk/, CTU in Prague

[4] *uLan Project*, Pavel Pisa, Petr Smolik, http://cmp.felk.cvut.cz/∼pisa/ulan/ul_drv.html, CTU in Prague

[5] *ProfiM - Profibus Master*, Pavel Trnka, Petr Smolik, http://profim.sourceforge.net, CTU in Prague

[6] *Debian Project*, http://wiki.debian.org/DebianLive

[7] *OCERA Project*, http://www.ocera.org