PROGRAM LIBRARY "LBS-LIB" FOR AGRICULTURAL BUS SYSTEM (LBS, DIN9684) – FIRST OPEN SOURCE PROJECT IN AGRICULTURE

A. SPANGLER, H. AUERNHAMMER, M. DEMMEL

Department of Bio Resources and Land Use Technology, Crop Production Engineering, Technical University Munich, Germany
E-mail : spangler@tec.agrar.tu-muenchen.de

ABSTRACT

Typical European agricultural machine combinations consist of devices of different brands. To realise electronic control of implements with standard terminals, automatic documentation and process optimisation, a capable standardised open communication protocol is needed. The existing standard for an agricultural BUS System DIN 9684 (LBS) needs a software tool, which enables an affordable, compatible and capable development of devices like tractor, implement and terminal. The Open Source program library "LBS-Lib" was designed to fulfil these needs. With the help of suitable interface functions, which serve standard algorithms for the interactions defined by the published standards, an agricultural equipment manufacturer can concentrate on creating the special application for his device. Adoption to different electronic control unit types and quick improvement of the software are supported by Open Source.

INTRODUCTION – REQUIREMENTS ON AN IMPLEMENTATION OF STANDARDISED OPEN COMMUNICATION PROTOCOL

Complexity of open communication protocol for agriculture

The open communication protocols DIN 9684 and ISO 11783 are designed to enable a manufacturer and construction independent documentation and process automation function. Despite closed networks like within a tractor, the configuration of the network (which devices) and each connected device (attributes) isn't known during development of single device. Corresponding to the strategies of automatic production planning systems (Siegert et al, 1996) the interactions should use an abstract view on all devices and their services. So an effective framework for the communication of measurement values or set-points of process information like speed, wheelslip and working state is important. Among the flexibility the process data interactions should be very effective in view of relation between BUS load and achieved information or control.
Regarding security in open communication networks, each device should be designed as separated intelligent system (FIG. 1), which:

- provides services (receive set-point and send information)
- uses services of other devices (send set-point and request information)
- decide which set-points to accept
- decide how accepted set-points are realised
- decide how sensor values are calculated to process data information

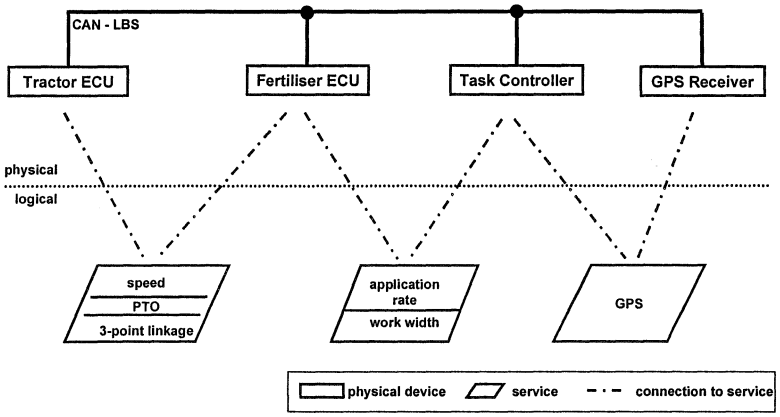

FIGURE 1: LBS – an open network of services

Need for high grade of compatibility

To reach the needed capability of the network and its components, all interactions must be implemented with a high grade of compatibility. Beside the formatting of single messages, the compatibility must be ensured for sequences of messages for different Use Cases like setting a set-point, its answering information (whether accepted or rejected) and automatic solving of conflicting set-points. Such situation dependent interaction sequences are hard to define in printed standard sufficiently, because of the size of documentation and arising of new Use Cases in real configurations.

PROBLEM –
REALISING COMPATIBLE COMMUNICATION SOFTWARE

The interactions based on process data enable a sophisticated object oriented communication. Each information instance like application rate has interface methods for exact, min and max set-points or retrieving of measurement values with programs. The process data algorithms should be flexible enough to handle parallel measuring programs independently and should manage received set-points dependent on sender. For the handling of complicated work constellations, each device should use compatible interaction sequences.
Keeping these requirements in mind, the development of a complete device is divided in realising complex base algorithms and algorithms specific for the application. The greatest part of compatibility and capability issues depends on the base infrastructure, so that a big need for a standard implementation arises. But at the moment every manufacturer created an individual and incompatible implementation of the standard for his specific ECU.

SOLUTION IDEA-
ENFORCING COMPATIBILITY AND QUALITY WITH THE HELP OF OPEN SOURCE

The main reasons for the lack of a commercial standard implementation for LBS are little profit expectations for software developers caused by the administration and support costs and the high development investment costs to reach commercial quality for such a product.
This is a typical case where Open Source projects can help. First, each user can adopt the software to the wanted platform on his own (can co-ordinate with others). Second, the more

the same software is tested from different people, the quicker and cheaper (in view of one using company) the needed level of quality can be reached.
If the same algorithms for all communication and interaction tasks can be accessed free of charge as Open Source from every interested manufacturer, a fundament for compatibility is created. Regarding financial aspects, such a standard implementation is also affordable for small projects. This is very important for complex interconnected networks, where the capability of the whole system is dependent from the implementation of the weakest part.
Driven by the needs of applications, an Open Source base tool can be extended rapidly to serve suitable interfaces for interactions to avoid incompatible messages and to allow easy access to complex interactions and network state interpretations. But such a quick evolution needn't shrink the quality, if all users perform co-ordinated tests. This interaction of software experts of different companies is possible, if everybody accepts the base interest of a well working compatible tool. A survey about the main reasons for embedded companies in using Linux revealed, that a "collaborative open source development produces superior software", "allows fully understanding what's going on inside the OS" and "eliminates dependence on a single OS vendor" were the main reasons to use Linux (Lehrbaum et al, 2001).

SOLUTION REALISATION-
OPEN SOURCE PROGRAM LIBRARY LBS-LIB

The partial project 2 of the research group IKB-Dürnast has to gather process data information with LBS. But as there are no flexible enough recording systems and "Implement Indicators" (IMI) available, they have to be developed. The research aim includes the recording of working state parameters like working depth, width or state, so that sensors must be connected to the IMI. The software of both the recording task controller and the IMI must be able to handle dependent on the configuration a flexible amount of process data values. Additionally the sensor signals must be interpreted to construction independent values. Last but not least the ECU of some implements offer only different subsets of the standardised interpreted information, so that the recording system must adapt the recording to the connected device.
The development of the software for the different computers can be eased by an appropriate program library, which isn't available (see above). So it was decided to develop a program library called LBS-Lib which enables primarily the wanted research and which can be used as the starting point for a reference implementation of DIN 9684 as an Open Source program library.

Design of LBS-Lib to work as Open Source project

The LBS-Lib is designed in three layers (FIG. 2), to realise in each of them the individual maximum code uniformity. The top layer, which contents all algorithms to define the communication, can be used without any changes on every platform. All hardware interactions like receiving or sending CAN messages are piped through the hardware abstraction layer. This serves an abstract platform independent interface to the communication layer and to the application itself. Each direct hardware access is handled by a hardware adaption layer, which uses mostly a simple name mapping from BIOS or OS functions and types. By restricting any hardware adoption changes to some simple header and source files, the adoption to different ECU types gets easier and most parts of the LBS-Lib can profit from optimisation and peer-review of the same code base.

As the adoption to a new platform is a change to the LBS-Lib, the "Lesser General Public License" (LGPL) (Free Software Foundation et al, 1999) enforces the publishing of the adopted files. This motivates interested people to co-ordinate the adoption, to speed the work.

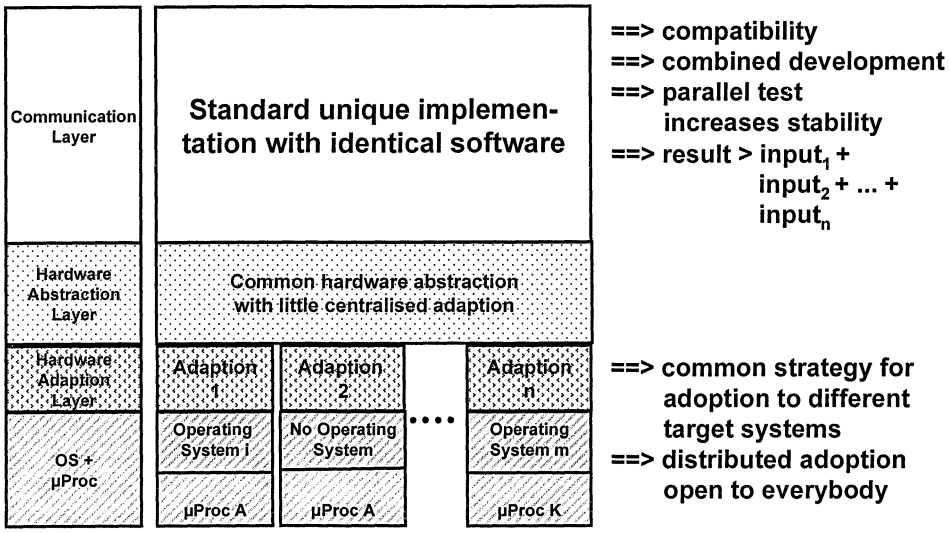

FIGURE 2: Layered structure of the LBS-Lib

An Open Source project needs a strict modularised design for the co-ordination of a distributed development. The LBS-Lib uses objects which are grouped to functional components. Therefore the evolution and peer-review validation can focus the levels object, functional group, LBS-Lib internal and interface.

The LBS-Lib communication layer consists of functional components for the different tasks (FIG. 3) of a device like system management and process data interaction. The hardware abstraction layer uses functional components for different hardware drivers.

The LBS-Lib was implemented using C++, because it supports an object oriented design and can be as fast, or faster than C and Fortran (Veldhuizen et al, 1998). The objects of the

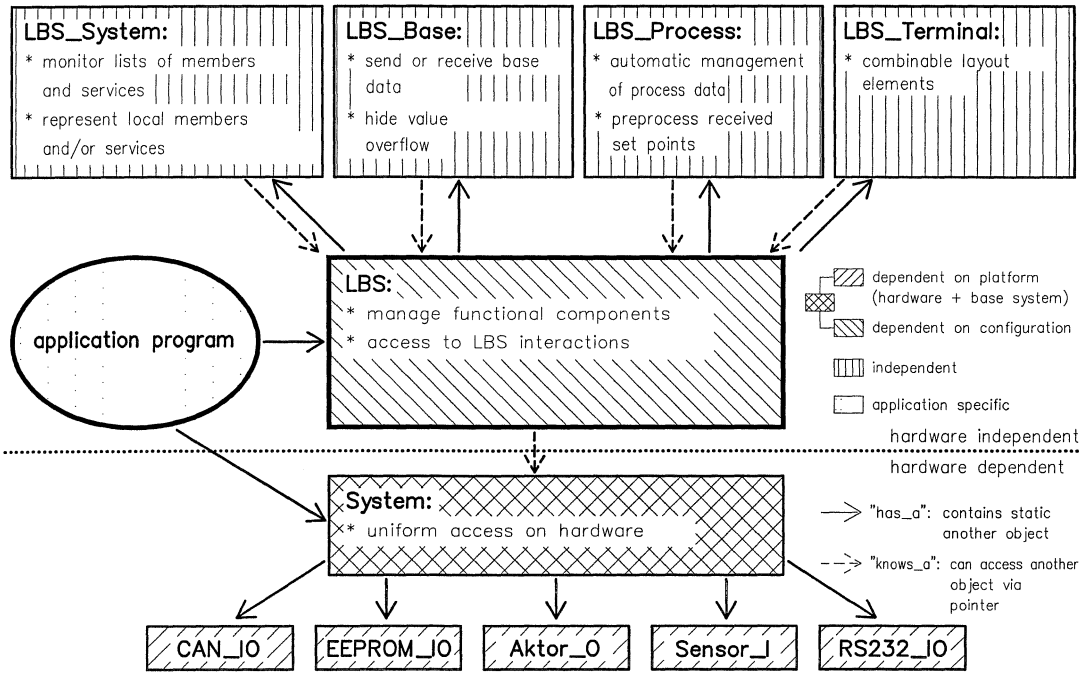

FIGURE 3: Modular structure of the LBS program library LBS-Lib

LBS-Lib are designed to perform standard actions autonomous in background, to facilitate the development of specific applications. Furthermore the LBS-Lib provides an interface definition for application development. A central configuration file allows the application specific exclusion of sub-functions like virtual terminal or some hardware drivers.
The modularised design of the LBS-Lib allowed a smooth integration of a DIN 9684 equivalent subset of the upcoming ISO 11783 standard, which consists of:

- system management
- periodic send or receive of real and wheel based speed and distance, front and back power take off (PTO) rounds per minute (RPM), front and back hitch position and engine RPM
- interactions based on process data

Process data messages, whose hosting device is identified by the key <device type, mounting position>, are routed automatically with the suitable protocol type dependent on the targeted device. So the application doesn't have to handle the protocol dependency, which is very important for a smooth combination of DIN 9684 and ISO 11783. To allow this, the DIN protocol has to use the standard ISO bitrate, as long as no device is using the DIN bitrate.

Open Source organisation of LBS-Lib

The LBS-Lib was presented within three national and international workshops in Freising and Sapporo (Japan), where an organisation of the project according to York (2000) was presented (FIG. 4). Typical for Open Source projects, the free available services need some kind of sponsoring. This money is needed for running the project and its infrastructure and for paying a central administrating group, which should stay independent from single manufacturers, because the decisions about rejecting or accepting changes or extensions must be commonly accepted as impartial. Lehrbaum (2000) and Staff (2000) describes some reasons, why it makes business sense for manufacturers who use an Open Source project to support it either with resources or with financial donations.
The basic idea of the open services is to share information and knowledge of a basic communication protocol which must be implemented compatible. This is not a quality feature on its own, because only the application which uses DIN or ISO offers real services.
Manufacturers who need special service like guaranteed response time limits, training of developers or project development assistance can get them based on commercial support contracts, if a company supplies commercial services for the LBS-Lib.

CONCLUSIONS AND OUTLOOK

With the integration of a subset of ISO 11783, the LBS-Lib is a capable tool for manufacturers, who want to develop one ECU for their machines, which can than be used with other already existing DIN and future ISO 11783 devices, so that the farmer needn't change all his equipment afterwards. But there must be also stated, that some developers have problems to adopt to the object oriented modelling of software.
It is very important for companies considering the use of this project that the LGPL guarantees a good quality in future in two ways which are stated by Moen (1999). First the license prevents the LBS-Lib from destructive forks, because the maintainer of the actual reference version can always take and integrate the changes and extensions of forked variants. Second the users are not dependent from specific people, because they can always appoint a new team of maintainers for a new forked reference version, if they are unsatisfied with the actual development of the project.

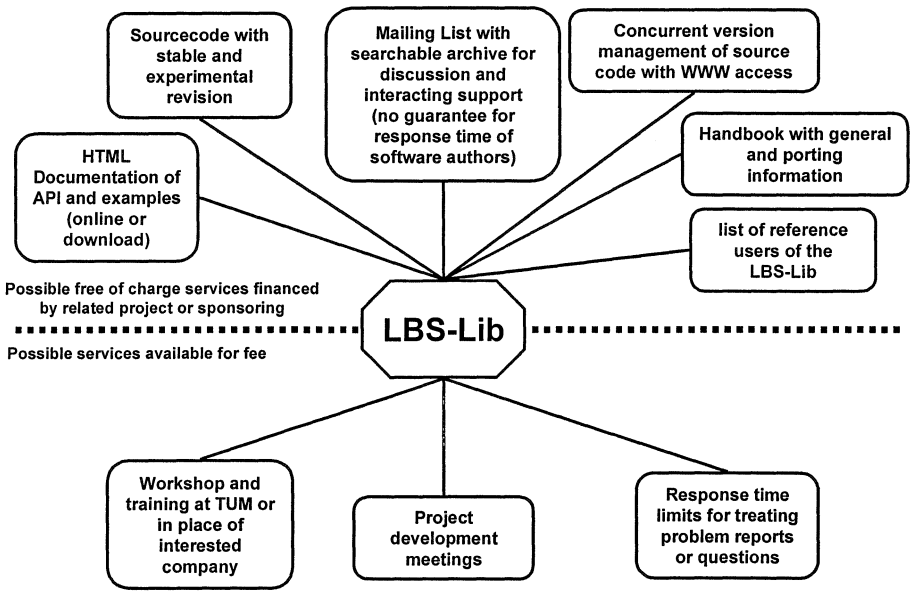

FIGURE 4: Possible future organization of the Open Source project LBS-Lib (derivated from York, 2000)

ACKNOWLEDGEMENTS

I give my thank to the *Deutsche Forschungsgemeinschaft* (DFG) for financing the research project and to the personnel of the experimental station Duernast.

REFERENCES

Auernhammer, H. (1993) Landwirtschaftliches BUS-System LBS, *KTBL-Arbeitspapier 196*, Germany

Deutsches Institut für Normung (1997) *DIN9684: Landwirtschaftliches BUS System*, Beuth Verlag GmbH, Berln, Wien, Zürich, Germany

http://www.gnu.org/copyleft/lesser.html, (11/27/2000)

http://www.linuxdevices.com/articles/AT8151978006.html, (01/23/2001)

http://www.linuxdevices.com/articles/AT8842791300.html, (11/06/2000)

http://www.linuxcare.com/viewpoints/article/11-17-99.epl, (11/17/1999)

Siegert, H. J. (1996) *Robotik. Programmierung intelligenter Roboter*, Springer Verlag, berlin, Heidelberg, Germany

http://linux.com/jobs/newsitem.phtml?sid=74&aid=7302, (02/19/2000)

Veldhuizen, T. L., Jernigan, M. E. (1998) Will C++ be faster than Fortran?, *Department of Systems Design Engineering*, University of Waterloo, USA

http://www.linux-mag.com/2000-02/trench_01.html, (02/01/2000)