Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Embedded software and hardware implementation system for a human machine interface based on ISOAgLib*

Enkhbaatar TUMENJARGAL†[1,2], Luubaatar BADARCH[1,2], Hyeokjae KWON[1], Woonchul HAM[1]

([1]*Division of Electronic Engineering, Chonbuk National University, Jeonbuk-do 561-756, Korea*)

([2]*School of Information and Communication Technology, Mongolian University of Science and Technology, Ulaanbaatar, Mongolia*)

†E-mail: enkhbaatar79@gmail.com

Received Sept. 19, 2012; Revision accepted Jan. 9, 2013; Crosschecked Feb. 25, 2013

Abstract: Modern agricultural machinery demands adoption of embedded electronic and remote sensing technology for precision agriculture. One of the electronic devices commonly used is the virtual terminal (VT) for tractors. A VT's functions and terminology are described in the ISO 11783 standard. This paper presents a control system design and implementation for a VT and some other electronic control units (ECUs) for agricultural vehicles based on that standard. Hardware and software development for the VT is implemented using the ISOAgLib open library, in the advanced embedded system. The main part of the system is an embedded board based on a Samsung S3C6410 ARM11 core microprocessor with a controller area network (CAN) module. Its working environment is Windows Embedded CE 6.0 (WinCE6.0). The ISOAgLib library provides abundant open sources consistent implementation of ISO 11783. It is written in C++ programming language using object-oriented technology. In this paper, we describe an ISO 11783-based tractor control system with a CAN and its implementation in the embedded system. This paper also explains the operation of a CAN-bus device driver in WinCE6.0 and some modifications of ISOAgLib for our target system. The target system consists of the VT, an ECU for the global positioning system (GPS), and an ECU for lighting for an agricultural tractor. The ECU for GPS and the ECU of a light controller are implemented using STM32F107F ARM Cortex M3-based development boards.

Key words: Controller area network (CAN)-bus, Virtual terminal (VT), Embedded system, ISO 11783
doi:10.1631/jzus.C1200270 **Document code:** A **CLC number:** TP39

1 Introduction

This paper describes the implementation of an agricultural machinery control system based on the ISO 11783 standard. The standard for a virtual terminal (VT), which is used to set up the operational parameters of tractors and implements, is described in the first edition of ISO 11783-6:2004. A VT is an operator interface device that displays information to operators and allows operators to input information.

A VT provides a common user interface for all devices that have a graphic interface. A VT contains a graphic display with a limited set of graphical objects, a few soft keys with an icon on display, and a means to navigate the display and manipulate the values. How the display is shown in a VT is stored in an 'object pool'. The object pool is a representation of a working set (WS), and consists of objects supported by the standard. The objects may be input numbers, output numbers, bar meters, needle meters, polygon graphics, or bitmap graphics. The objects have parameters like position, size, color, and value. The object pool defines object types, the relation between objects, and all the parameters for each object. A WS allows communications among units on a network where several control functions are acting as distributed processes providing a single application. These several control functions, each with a distinct NAME, can be in different electronic control units (ECUs) connected by different nodes. As soon as the WS is connected to the network and powered, the VT and

* Project supported by the Post BK21 Project of Chonbuk National University and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (No. 2010-0010531)

WS start to communicate. After initial handshaking and requests, the WS starts to upload its object pool to the VT, and the display appears on the VT screen. If the mobile system (tractor with implements) contains more than one WS, the active display can be changed on the VT (Stone *et al.* 1999).

The ISO 11783 standard was jointly developed by tractor and implements manufacturers including AGROCOM GmbH & Co. Agrarsystem KG (2009), AGCO Co. (2002), Deere & Co. (2012), DICKEY-John Co. (2012), and Müller Elektronik GmbH & Co. (2012). These manufacturers have also created a specification defining how this standard should be recognized. This specification is commonly known as ISOBUS. Several research teams work on the implementation of VTs and the ISO 11783 standard. In Korea, we are cooperating with the Korea Agriculture Company in research on and application of VTs. The purpose of our work is to implement hardware and software design of VTs based on the ISO 11783.

In this paper, we propose a generic ISO 11783-compatible implementation based on the ISOAgLib library. The ISOAgLib provides open source libraries to implement the ISO 11783 standard, and is written in an object-oriented C++ language. Also, we introduce a concept of the device drivers in Windows Embedded CE 6.0 (WinCE6.0). The design and implementation of reliable device drivers is notoriously difficult and constitutes the main source of system failures. We consider mainly the device drivers for the controller area network (CAN) 2.0B protocol and the high speed serial peripheral interface (SPI) bus. The S3C6410 microprocessor has no CAN-bus interface but two high speed SPI interfaces. The CAN-bus module consists of an MCP2515 CAN controller and a CAN-bus driver chip. The MCP2515 CAN controller communicates with the S3C6410 microprocessor via the SPI interface.

In recent years, academic and industrial institutes have devoted attention to developing ISO 11783 standard related ECUs, VT, the task controller (TC), and file server (FS). Stone *et al*. (1999) described how to use the standard for agricultural machinery and compared other standards DIN9684 and SAE J1939. Ohman *et al.* (2008), Pereira *et al.* (2009), and Oksanen *et al.* (2011) implemented VTs and TCs for agricultural machinery. Also, Kim *et al.* (2011) worked on a communication model for the ISO 11783 standard. Speckmann and Jahns (1999) and Craessaerts *et al.* (2005) investigated the possible benefits of CAN-bus development in Windows-based programming environments.

2 Background and benchmarking

2.1 Controller area network (CAN) protocol

In a CAN network, each message is preceded by an identifier that is unique to the transmitting controller and multiple controllers can communicate over a single two-wire bus. A CAN transmits data in frames containing a header and 0 to 8 bytes of data. There are three separate CAN standards: CAN 1.0, 2.0A (standard CAN), and 2.0B (extended CAN). The main difference between these standards is the length of the identifiers that precede each message. The original specifications (Versions 1.0 and 2.0A) specify an 11-bit message identifier. Version 2.0B extended frames contain a 29-bit identifier which allows over $2^{29}-1$ message identifiers (Bosch, 1991).

The 29-bit identifier is made up of an 11-bit identifier ('base ID') and an 18-bit extended identifier ('ID extension'). Fig. 1 shows the differences between the three CAN standards. This paper describes our research based on the CAN 2.0B version. For

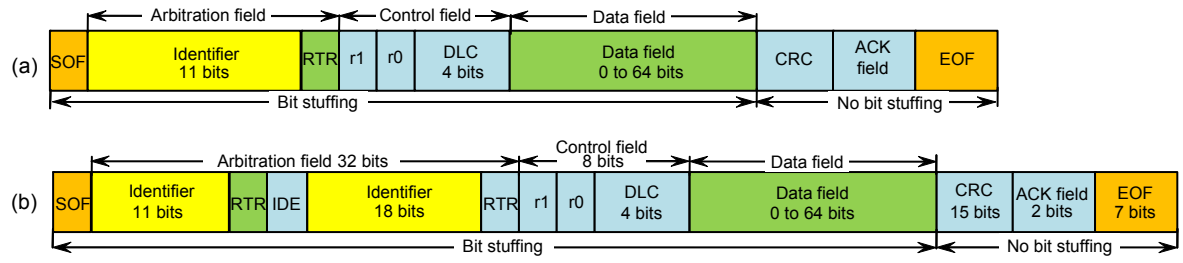

Fig. 1 Frame structures of the CAN-bus: (a) CAN 2.0A; (b) CAN 2.0B

CRC: cyclic redundancy check; DLC: data length code; EOF: end of frame; IDE: identifier extension bit; RTR: remote transmission request bit; SOF: start of frame

more detailed information regarding the CAN-bus, see Bosch (1991).

2.2 Overview of the ISO 11783 standard

The ISO 11783 standard contains specifications for a serial data network for control and communications in forestry or agricultural tractors and mounted, semi-mounted, towed or self-propelled implements. Its purpose is to standardize the method and format of transfer of data between sensors, actuators, control elements, tractor, etc. The ISO 11783 standard is sometimes called ISOBUS. It consists of several parts: a general standard for mobile data communication, a physical layer, a data link layer, a network layer, a network management, a VT, an implement messages applications layer, power train messages, a tractor ECU, a task controller and management information system data interchange, a mobile data element dictionary, a diagnostic, and a file server. Fig. 2 describes a typical ISO 11783 network topology (ISO 11783-4:2001).

In this part, we describe the data link layer and the use of CAN extended data frames over the network. The data link layer enables the reliable transfer of data across the physical link. This consists of sending the CAN data frame with the necessary synchronization, sequence control, error control, and flow control. Table 1 shows the arbitration and control fields of the 29-bit identifier for CAN and the 29-bit identifier for ISO 11783. For more information regarding the communication protocol, see standard parts ISO 11783-3:2007, ISO 11783-4:2001, and ISO 11783-5:2001.

Table 1 Mapping of ISO 11783 into a CAN arbitration and control field (ISO 11783-3:2007)

Bit number	CAN	ISO 11783
Bit-1	Start of frame (SOF)	SOF
Bit-2	ID28	Priority (P3)
Bit-3	ID27	P2
Bit-4	ID26	P1
Bit-5	ID25	EDP
Bit-6	ID24	DP
Bit-7–Bit-12	ID23–ID18	PF8–PF3
Bit-13	SRR	SRR
Bit-14	IDE	IDE
Bit-15–Bit-16	ID17–ID16	PF2–PF1
Bit-17–Bit-24	ID15–ID8	PS8–PS1
Bit-25–Bit-32	ID7–ID0	SA8–SA1
Bit-33	RTR	RTR
Bit-34–Bit-35	r1–r0	r1–r0
Bit-36–Bit-39	DLC4–DLC1	DLC4–DLC1

DLC#: data length code bit number; DP: data page; EDP: extended data page; ID#: identifier bit number; IDE: identifier extension bit; PF#: PDU format bit number; PS#: PDU specific bit number; RTR: remote transmission request bit; SA: source address; SRR: substitute remote request

ISO 11783 currently supports five types of messages: commands, requests, broadcast/responses, acknowledgements, and group functions. The command message type categorizes those parameter groups that command a specific or global destination from a source. The request message type, identified by the parameter group number (PGN), provides the ability to request information globally or from a specific destination. The broadcast/response message type can be either an unsolicited broadcast of information from a controller or the response to a command or request. Acknowledgement messages are available in two

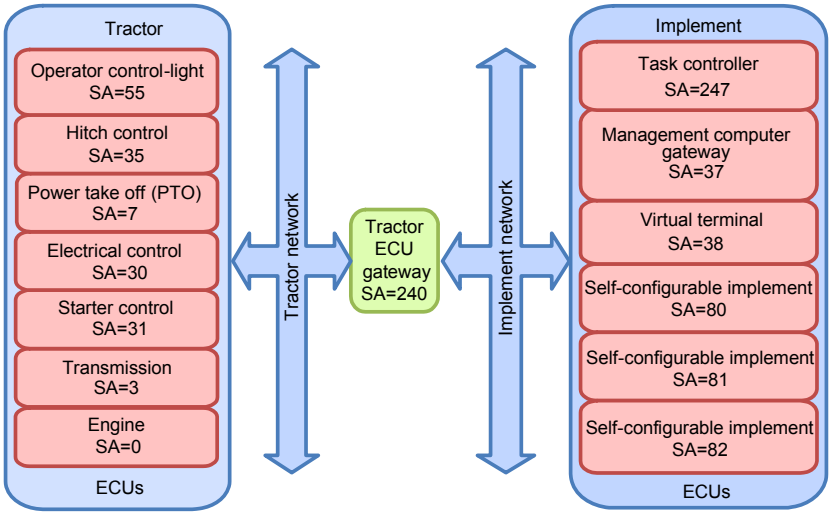

Fig. 2 A typical ISO 11783-4:2001 standard network topology
SA: source address

forms: the first is provided for by the CAN protocol and the second is a response of a 'normal broadcast', 'ACK', or 'NACK' to a specific command or request, provided for by an application layer. The group function message type is used for groups of special functions (proprietary functions, management functions, multi-packet transport functions, etc.).

Transport protocol (TP) functionality is subdivided into two major functions: message 'packetization' and reassembly, and connection management. These are described in the following subclauses, in which the term 'originating controller' corresponds to the controller that transmits the request-to-send (RTS) message, and the receiving controller corresponds to the controller that transmits the clear-to-send (CTS) message. Messages greater than 8 bytes in length are too large to fit into a single CAN data frame. Therefore, they are broken into several smaller packets, which are transmitted in separate CAN data frames.

The individual packets that comprise a large message have to be identified separately, so that they can be reassembled correctly. The first byte of the data field is defined as the sequence number of the packet. Individual message packets are assigned a sequence number of from 1 to 255. This yields a maximum message size of 255 packets×7 bytes/packet =1785 bytes. Fig. 3 illustrates message packetization and reassembly processes.

A connection management message is used to initiate and close connections, and to control flow. A TP provides the following messages: the connection mode request to send (TP.CM.RTS), the connection mode clear to send (TP.CM.CTS), the end of message acknowledgment (TP.EndOfMsgACK), the connection abort (TP.AbortMsg), and the broadcast announce message (TP.BAM). A data transfer (TP.DT) message is used to communicate the data associated with a PGN. Fig. 3 describes the multi-message packetization and sample transport protocol processing. ISO 11783 system messages greater than 8 bytes are transferred to and received from ECUs with a TP or an extended TP. The TPs are specifically used for the TC, FS, and VT. For more information about the VT, TC, and FS, see standard parts of ISO 11783-10:2009, ISO 11783-6:2004, and ISO 11783-13:2007.

2.3 Benchmarking the ISOAgLib library

In this section, we introduce the ISOAgLib open source programming library, developed by University of Munich and the OSB & IT Engineering Co. in Germany (Spangler and Wodok, 2010). It is suitable for embedded communication software in ECUs such as VT, TC, and FS. All functions relating to the ISO 11783 standards as well as the established machine interfaces are already implemented in the library. Fig. 4 describes the system architecture of the ISOAgLib programming library.

The ISOAgLib library consists of several parts: communication, scheduler, driver extension, hardware abstraction layer (HAL), and hardware/driver. The library allows the building of ISOBUS-compatible equipment without the protocol implementation contained in this standard. The ISOAgLib is now hosted and maintained by the OSB & IT Co.

The ISOAgLib was developed to be compatible with various systems, and these systems can be composed of microprocessors, memory, human machine interfaces (HMIs), and interfaces with CAN-bus. Because of this, the ISOAgLib is divided into two parts, the library itself and an HAL. The library provides the address claim (AC) negotiation of an ECU to the CAN-bus. It also includes the implementation of TPs that are used in the initialization of an ECU communication with a VT and TC. The HAL is responsible for communicating with the operating system or BIOS device that is running the application (Fig. 4). The ISOAgLib library-based embedded devices are implemented in a Linux and Windows operating system environment.

Many versions of the ISOAgLib programming library have been developed, as well as some tutorials and examples that are useful for studying ISOBUS. In our case, we used version 2.2-rc5 of ISOAgLib, since this version includes some tutorials and examples. In particular, we studied the CAN server application program, and the ECUs for global positioning system (GPS) sensor, display, DataSource, TractorBridge, etc. The previous ECUs tested in the Windows XP environment are shown in Fig. 5. By testing all ISOAgLib programs and executing in a console application program, you can see all the transmitted and received CAN messages. We used several development environments for modifying ISOAgLib source codes and implementing projects, including the DEV CPP free development tool, GNU, and Microsoft Visual Studio C++.

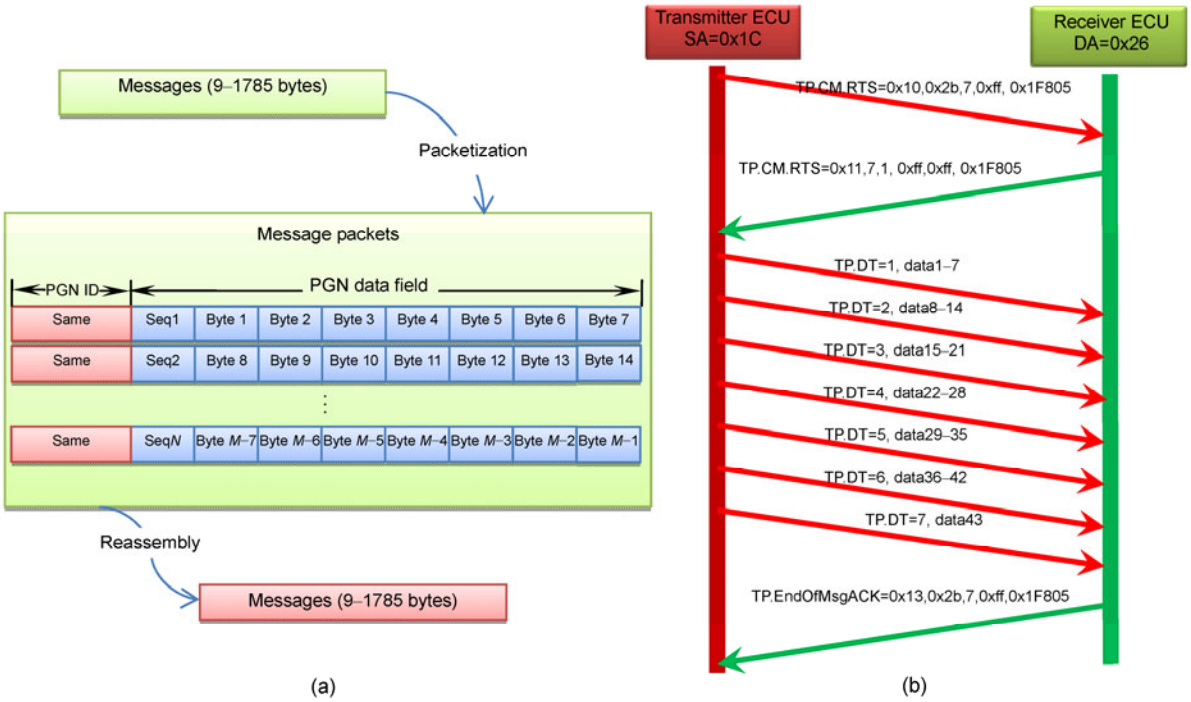

Fig. 3 Multi-message packetization process (a) and the sample for transport protocol processing (b) (ISO 11783-3: 2007) (DA: destination address; SA: source address)

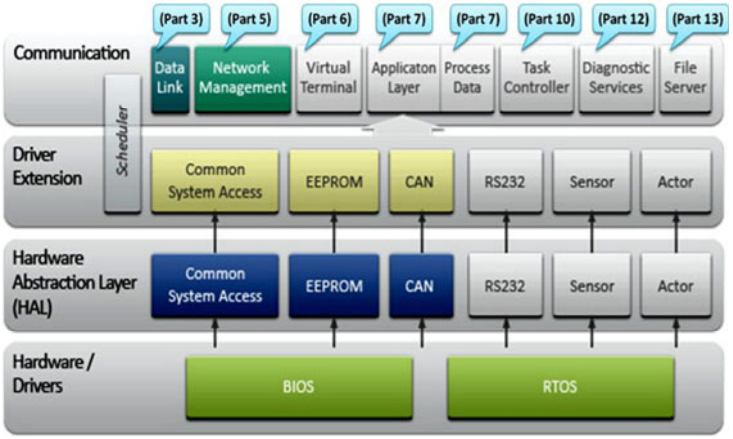

Fig. 4 System architecture of the ISOAgLib open library (Spangler and Wodok, 2010)

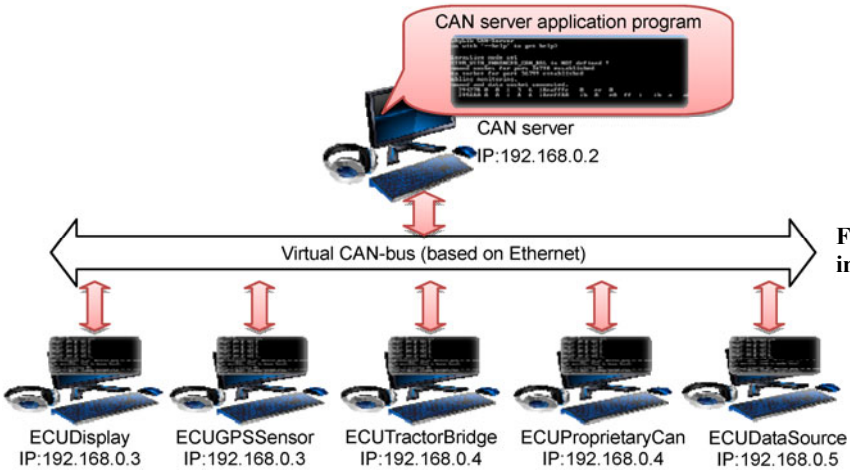

Fig. 5 Virtual ECUs working with a CAN server

Figs. 5 and 6 describe the simulation structure of the ISOAgLib open source system in Windows XP, which consists of several virtual ECUs working with a CAN server application program.

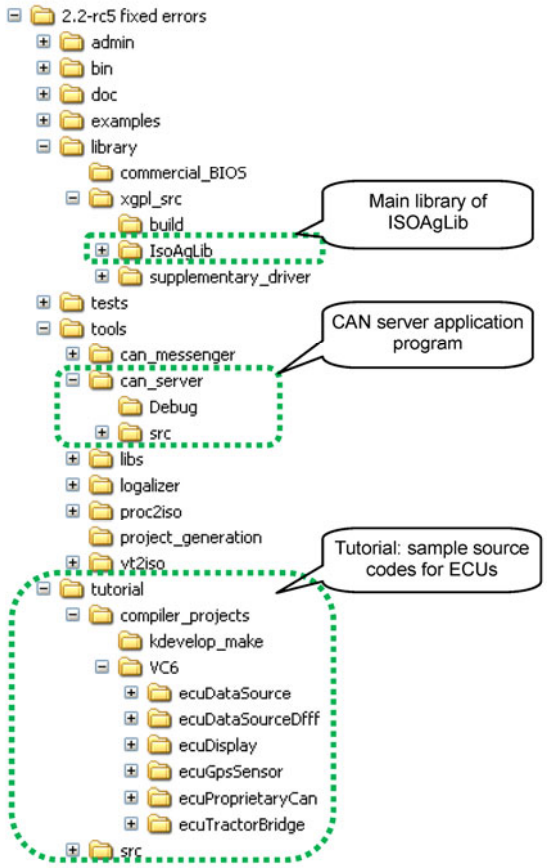

Fig. 6 Folder structure of the ISOAgLib version 2.2-rc5

3 Implementation of hardware

This section describes the hardware implementation of the VT and sample ECUs based on ISOAgLib and ISOBUS. The main part is implementation of CAN-bus in the embedded system. The hardware of the VT consists of the advanced embedded board and the CAN-bus module.

We have developed and implemented the whole hardware schematic and prototype circuit board (PCB). The Samsung S3C6410 32 bit ARM11 Core (667 MHz) microprocessor includes several useful interfaces: the high-speed SPI, 2D/3D graphics acceleration, USB2.0 OTG, Ethernet, camera interface, etc. Since the S3C6410 microprocessor does not have a CAN-bus interface, we use the MCP2515 CAN microcontroller in our system, which has an SPI communication interface. Fig. 7 describes the design of the CAN-bus module and implemented prototype.

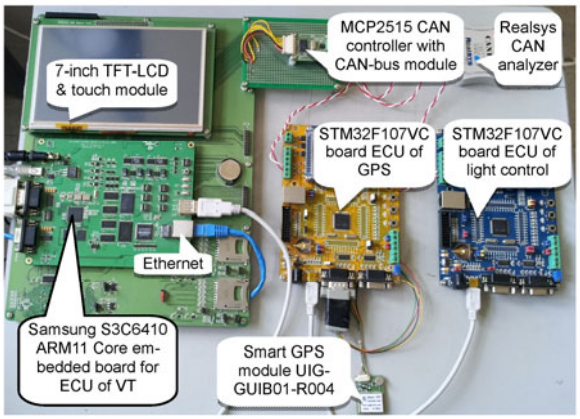

Fig. 7 The whole embedded system consisting of the VT and sample ECUs of GPS and lighting

The hardware of other ECUs is implemented in the 32-bit microcontroller STM32f107VC development board. Fig. 8 illustrates the characteristics of the ECU implementation board. The STM32F microcontroller's main central processing unit (CPU) architecture is an advanced RISC machine (ARM) V7 Cortex-M3, which is facilitated by several useful peripherals. The CAN controller plays an important role as it supports CAN 2.0A and 2.0B active and passive with data rates up to the maximum 1 Mb/s. The CAN controller also has extensions to support fully deterministic communication defined under the time-triggered CAN (TTCAN) protocol. When enabled, the TTCAN extensions support automatic message retransmission and will place a timestamp message in the last two data bytes of the CAN message packet. The next most important feature of the CAN controller is the filtering of received messages.

Since a CAN is a broadcast network, every message transmitted is received by every node on the network. In a CAN network of any reasonable complexity, there will be a large number of messages sent over the CAN-bus. In such a network, the CPU of a CAN node will spend all its runtime responding to CAN messages. To avoid this problem, all CAN controllers have some form of message filtering that blocks unwanted messages from reaching the receive buffers.

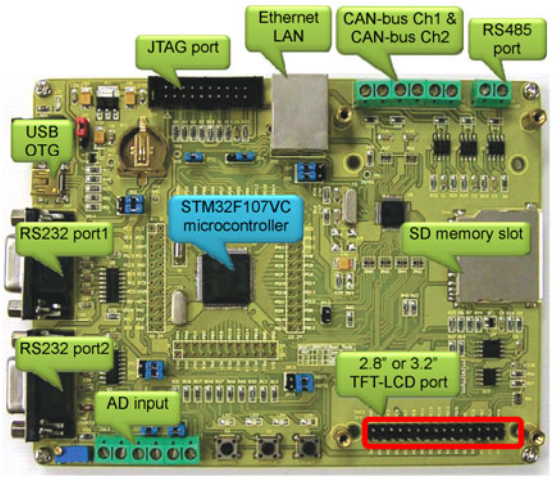

Fig. 8 Characteristics of the STM32F107VC microcontroller development board

JTAG: joint test action group; LAN: local area network; OTG: on-the-go; TFT-LCD: thin film transistor liquid crystal display

4 Implementation of software

This section describes the firmware and system level programming. The proposed implementation of the software environment is shown in Fig. 9, which depicts the application program of a VT that works with real ECUs via a CAN-bus. The application program of the VT runs in WinCE6.0 and accesses the CAN-bus using a CAN device driver. The virtual ECU application programs work in a Windows XP environment on a PC, which is very helpful for analyzing the software working processes.

4.1 Programming methodology at the firmware level

The firmware level programming CPU executes only one process; no operating system concept is considered. If the hardware system is not working with an operating system, this time programming code should be written at a firmware level.

Because most embedded system developers use standard C programming language, deploying code for a complicated system is very limited. Implemtation of the ISO 11783 standard is not a simple project for programmers. The ISOAgLib library's source-codes are written in an object-oriented programming language C++, and can easily be used to solve some complicated protocols. Since C++ programming is supported by the Standard Template Library (STD), methods such as template and namespace, list, queue, and vectors are often used in a programming. Another important concept for programmers is a software development tool for embedded systems. An embedded development tool should consist of a compiler and a linker devoted for a programming and

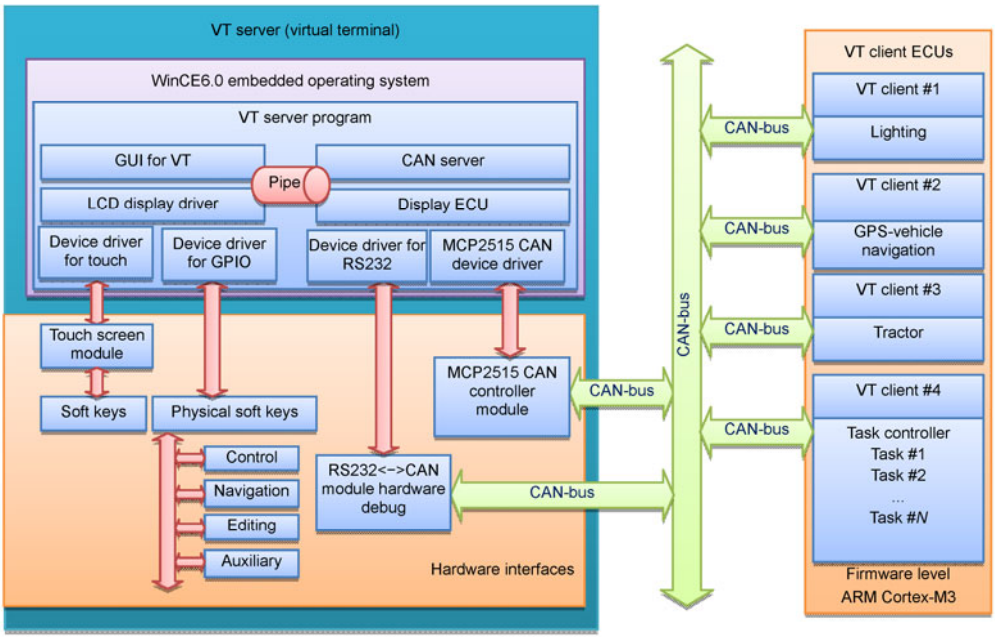

Fig. 9 Proposed development block diagram of VT and environment

GPIO: general purpose input/output; GUI: graphical user interface; LCD: liquid crystal display

assembler codes for certain microprocessors. One of the popular tools is IAR Embedded Workbench, which supports C/C++ and assembler for ARM7/ARM11 core processors. We successfully implement the codes for the ECUs of GPS and Sprayer based on the ISOAgLib library in the IAR Embedded Workbench environment. The IAR Embedded Workbench tool cooperates with a J-Link universal debugger device which is very useful for developing embedded system dedicated codes.

4.2 Programming methodology at the system level

System level programming means programming a code to be consistent with a certain operating system: in our case, WinCE6.0. In this section, we consider the device driver for CAN-bus and an application program for VT in WinCE6.0.

4.2.1 Programming for a device driver

Device drivers provide a bridge between a peripheral device and the upper layer of the operating system and the application software. In fact, device drivers are the largest part of the board support package (BSP) for an operating system design. Design and verification of device drivers is very complicated due to the need for a thorough knowledge of chips and boards, microprocessors, peripherals, operating systems, compilers, logic and timing requirements, each of which is considered to be difficult. Several different types of driver implementation architecture are available. The most common type in WinCE6.0 is a layered device driver structure. In this architecture, a driver consists of two parts, the model device driver (MDD) library and the physical device driver (PDD) library.

The operating system boot-up then configures device drivers for peripherals used in the system. This means that the CAN-bus device driver is already initialized as the system starts. The application program of the VT just calls the device driver DLL file (our case CAN.dll) from the kernel. The VT should access the CAN module through the device driver on the operating system. The DLL port functions realized by the CAN stream driver are: CAN_Init, CAN_Deinit, CAN_Open, CAN_Close, CAN_Read, CAN_Write, CAN_Seek, CAN_IOControl, CAN_PowerDown, and CAN_PowerUp. Fig. 10 describes the access of the application program to the CAN-bus device driver.

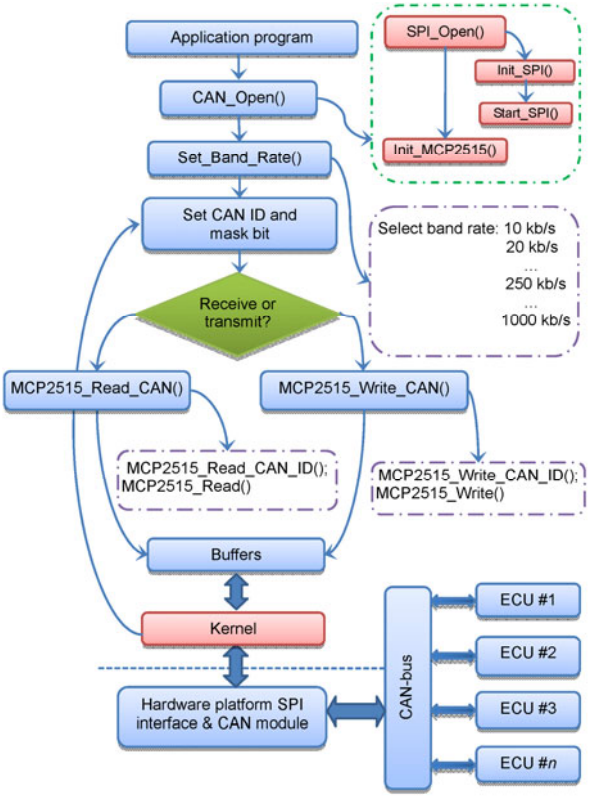

Fig. 10 Sample flow chart for the application program of a VT working with a CAN-bus device driver

4.2.2 Programming for a VT application

The application program for a VT is implemented based on the ISOAgLib open source library and codes. The VT working process and sample design are standardized in the ISO 11783 standard. VT-related codes are implemented in the ISOAgLib library, but nowadays the OSB & IT Engineering Co. takes care of the whole source codes and library. They use the VT-designer tool for designing the VT program. Because of the high price of this commercial programming tool (annual fee 1450 euros) we prefer to analyze and use the source codes for the VT2ISO tool. This converts the virtual terminal designer project file (VTP) and extensible markup language (XML) file to C++ and header files for VT application. The VT2ISO tool uses some open source library for the XML parser, which is Xerces-c_2_5_0D.dll file. This procedure is described in Fig. 11. We develop the Windows application program for a VT in MFC and C++ programming languages. The main working procedure of a VT server ECU and VT Client ECUs (or WS) is illustrated in Fig. 12. We present the

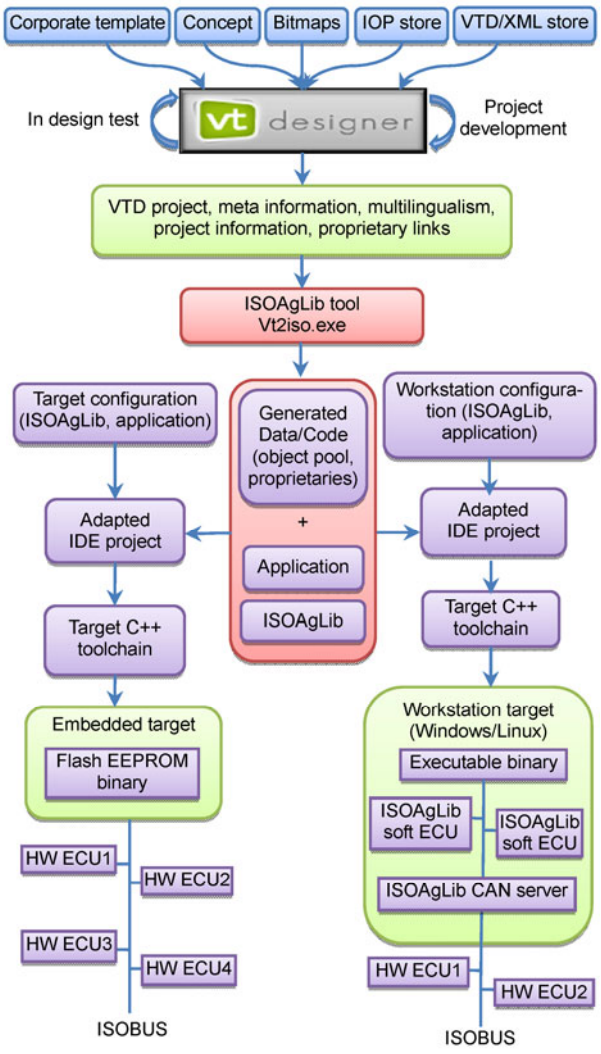

Fig. 11 Complete development structure of ISO 11783 applications with the ISOAgLib toolchain
IOP: ISOBUS object pool; VTD: virtual terminal designer

considerations with regard to the sample ECUs for GPS sensor and lighting control system architecture, as examples. In the next section, we describe our experimental results and give more detailed descriptions of these implementations.

5 Experimental results

The main propose of this work was to implement a VT application program and hardware design based on ISO 11783 and the ISOAgLib library. We developed sample systems for GPS and lighting with VT, and the implementations consist of hardware and software systems. The hardware system for VT is an advanced embedded board that is supported by the real-time operating system WinCE6.0. Currently, the application level program development is based on ISOAgLib which is developed only graphically for our system. The main communication parts are implemented using the ISOAgLib library according to the ISO 11783 standard. We considered two sample systems, the GPS sensor receiver and the light control, for a tractor. Figs. 13 and 14 show the implementation of sample lighting and GPS ECUs, respectively, working with a VT.

According to the ISO 11783 standard, the initialization messages between the implemented ECUs and the VT (Tables 2 and 3) include address claiming, PGN request, sending status message of VT and WS master, language information, memory information, softkey information, hardware version, and connection management messages.

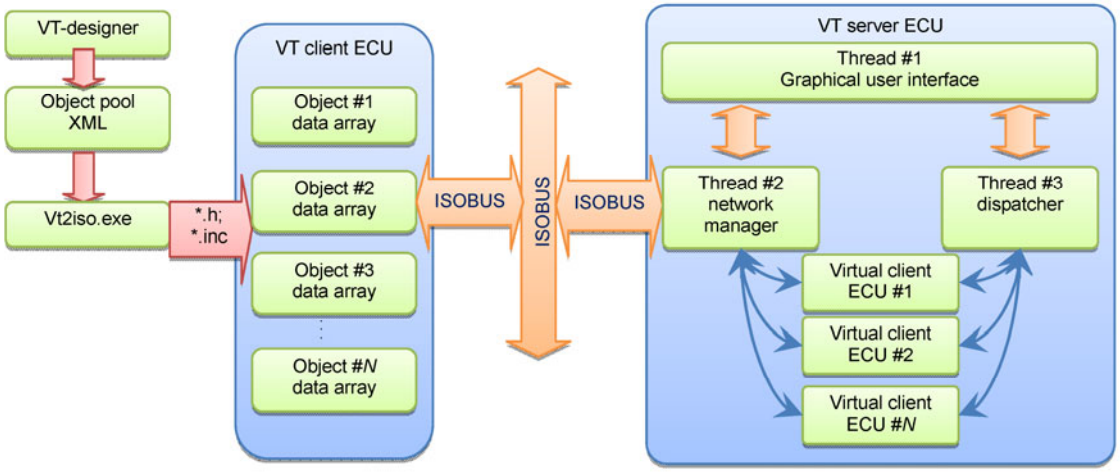

Fig. 12 Working procedure of a VT server and client ECU

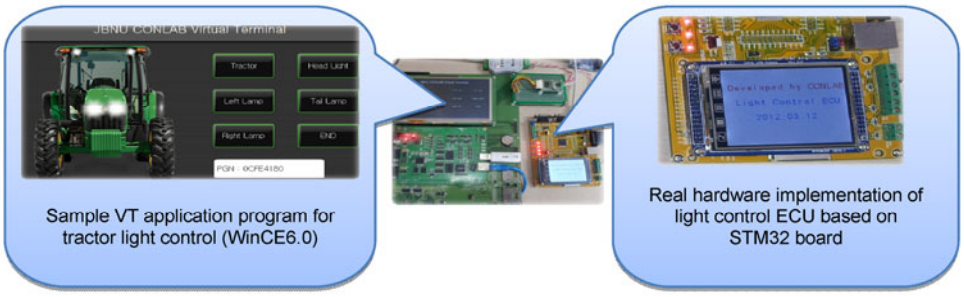

Fig. 13 Implementation of a lighting ECU and a lighting control application running on a VT

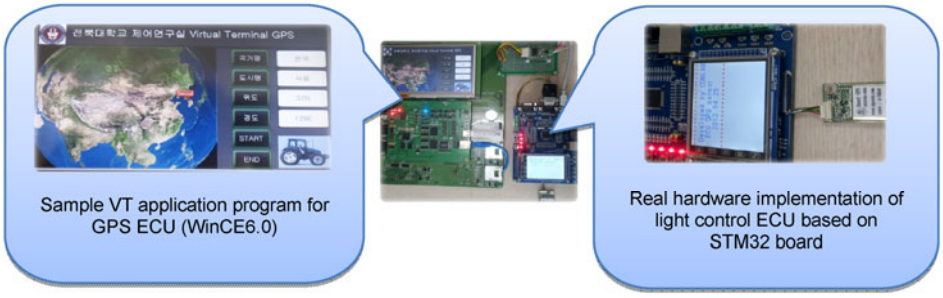

Fig. 14 Implementation of a GPS ECU and a GPS information monitoring application running on a VT

Table 2 Sample messages of ECU GPS and VT

Message	PGN			SA	DA	DLC	Data							
VT_Addr.Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
GPS_Addr.Claimed	00	EE	00	1C	All	8	1B	00	E0	FF	01	19	0E	A0
VT_Request	00	EA	00	26	All	3	00	EE	00					
VT_Addr.Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
GPS_Addr.Claimed	00	EE	00	1C	All	8	1B	00	E0	FF	01	19	0E	A0
VT_Language Code	00	FE	0F	26	–	8	64	65	40	00	00	00	FF	FF
GPS_WS master	00	FE	0D	1C	–	8	01	FF	FF	FF	FF	FF	FF	FF
GPS_Get memory	00	E7	00	1C	26	8	C0	FF	00	00	00	00	FF	FF
VT_Response Get memory	00	E6	00	26	1C	8	C0	04	00	FF	FF	FF	FF	FF
GPS_Get number of SoftKey	00	E7	00	1C	26	8	C2	FF	FF	FF	FF	FF	FF	FF
VT_Response Get number of SoftKey	00	E6	00	26	1C	8	C2	01	FF	FF	40	28	6	40
GPS_Get text font	00	E7	00	1C	26	8	C3	FF	FF	FF	FF	FF	FF	FF
VT_Response Get text font	00	E6	00	26	1C	8	C3	FF	FF	FF	FF	FF	FF	FF
GPS_Get hardware	00	E7	00	1C	26	8	C7	FF	FF	FF	FF	FF	FF	FF
VT_Response Get hardware	00	E6	00	26	1C	8	C7	FF	02	04	F0	00	F0	00
GPS_TP.CM_RTS	00	EC	00	1C	26	8	10	2B	00	07	FF	05	F8	01
VT_TP.CM_CTS	00	EC	00	26	1C	8	11	07	01	FF	FF	05	F8	01
GPS_TP.DT	00	EB	00	1C	26	8	01	FF	00	00	36	C3	DE	1D
GPS_TP.DT	00	EB	00	1C	26	8	02	00	34	8E	34	83	EF	AD
GPS_TP.DT	00	EB	00	1C	26	8	03	06	00	10	50	3D	64	58
GPS_TP.DT	00	EB	00	1C	26	8	04	9A	01	F0	D8	FF	FF	87
GPS_TP.DT	00	EB	00	1C	26	8	05	13	00	00	DD	FD	00	CD
GPS_TP.DT	00	EB	00	1C	26	8	06	CD	CD	CD	CD	CD	CD	CD
GPS_TP.DT	00	EB	00	1C	26	8	07	00	FF	FF	FF	FF	FF	FF
VT_TP.CM_EndOfAckMsg	00	EC	00	26	1C	8	13	2B	0	7	FF	05	F8	01

DA: destination address; DLC: data length code; PGN: parameter group number; SA: source address

Table 3 Sample messages of the lighting ECU and the virtual terminal

Message	PGN			SA	DA	DLC	Data							
VT_ Addr.Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
Light_Addr.Claimed	00	EE	00	80	All	8	25	B3	FF	E8	00	17	00	A0
VT_Request	00	EA	00	26	All	3	00	EE	00					
VT_ Addr.Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
Light _ Addr.Claimed	00	EE	00	80	All	8	25	B3	FF	E8	00	17	00	A0
VT_Language Code	00	FE	0F	26	–	8	64	65	40	00	00	00	FF	FF
Light _WS master	00	FE	0D	80	–	8	01	FF	FF	FF	FF	FF	FF	FF
Light _Get memory	00	E7	00	80	26	8	C0	FF	00	00	00	00	FF	FF
VT_Response Get memory	00	E6	00	26	80	8	C0	04	00	FF	FF	FF	FF	FF
Light _Get number of SoftKey	00	E7	00	80	26	8	C2	FF	FF	FF	FF	FF	FF	FF
VT_Response Get number of SoftKey	00	E6	00	26	80	8	C2	01	FF	FF	40	28	6	40
Light _Get text font	00	E7	00	80	26	8	C3	FF	FF	FF	FF	FF	FF	FF
VT_Response Get text font	00	E6	00	26	80	8	C3	FF	FF	FF	FF	FF	FF	FF
Light _Get hardware	00	E7	00	80	26	8	C7	FF	FF	FF	FF	FF	FF	FF
VT_Response Get hardware	00	E6	00	26	80	8	C7	FF	02	04	F0	00	F0	00
Light _TP.CM_RTS	00	EC	00	80	26	8	10	8D	06	F0	FF	41	FE	00
VT_TP.CM_CTS	00	EC	00	26	80	8	11	10	01	FF	FF	41	FE	00
Light _TP.DT	00	EB	00	80	26	8	01	11	01	00	00	08	01	64
…														
Light _TP.DT	00	EB	00	80	26	8	07	00	FF	FF	FF	FF	FF	FF
VT_TP.CM_EndOfAckMsg	00	EC	00	26	80	8	13	8D	06	F0	FF	41	FE	00

PGN: parameter group number; DA: destination address; DLC: data length code; SA: source address

6 Conclusions

Besides developing this system, we have checked the feasibility of using the ISOAgLib open source library to implement the real implementation of ECUs for agricultural machinery. In addition to the successful implementation of the HMI and several useful ECUs based on the ISOAgLib library, a real hardware system based on advanced embedded boards was developed. In this work, we used two kinds of embedded boards: the ARM11 core-based system for VT and the ARM Cortex-M3 core microcontroller development board for sample ECUs. The device driver for the high speed SPI interface and MCP2515 CAN controller was implemented in the WinCE6.0 operating system. In future work, we aim to improve further the application program for VTs according to the technical specifications of ISO 11783. Our developed product, consistent with the standardization of the communication between tractor and implement, is convenient for tractor drivers and farmers. In addition to developing the VT, we are aiming to develop an application program that can be used for automatic generation of a code for any proposed ECU for an agricultural tractor (e.g., ECUs for Sprayer, Data Source, Auxilirary Sensor, and Tractor Bridge, and for TCs, and FSs).

References

AGCO Co., 2002. FieldStar, the Science of Agriculture. Virtual Terminal User's Guide, Publication No. 79015206, Duluth, USA.

AGROCOM GmbH & Co. Agrarsystem KG, 2009. CEBIS MOBILE VA User Manual. Bielefeld, Germany.

Bosch, R., 1991. CAN Specification Version 2.0. Postfach, Germany.

Craessaerts, G., Maertens, K., de Baerdemaeker, J., 2005. A Windows-based design environment for combine automation via CANbus. *Comput. Electron. Agric*, **49**(2): 233-245. [doi:10.1016/j.compag.2005.04.007]

Deere & Co., 2012. GreenStar 3 Display 2630 Operator's Manual. Publication No. OMPFP12408, California, USA.

DICKEY-John Co., 2012. Auto Section Control System: Operator's Manual. Publication No. 11001-1561B-201207, Auburn, USA.

ISO 11783-1:2007. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 1: General Standard for Mobile Data Communication. International Organization for Standardization, Geneva.

ISO 11783-3:2007. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 3: Serial Control and Communications Data Network. International Organization for Standardization, Geneva.

ISO 11783-4:2001. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 4: Network Layer. International Organization for Standardization, Geneva.

ISO 11783-5:2001. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 5: Network Management. International Organization for Standardization, Geneva.

ISO 11783-6:2004. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 6: Virtual Terminal. International Organization for Standardization, Geneva.

ISO 11783-10:2009. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 10: Task Controller and Manegement Information System Data Interchange. International Organization for Standardization, Geneva.

ISO 11783-13:2007. Tractors and Machinery for Agriculture and Forestry-Serial Control and Communications Data Network-Part 13: File Server. International Organization for Standardization, Geneva.

Kim, S., Park, S., Kim, C., Kim, M., 2011. Implementation of the Communication Model for ISO 11783 Standards Based on AUTOSAR. ASABE Annual Int. Meeting, p.1-12.

Müller Elektronik GmbH & Co., 2012. ISOBUS-Terminals Flexible and Future-Proof Through APP & GO. Salzkotten, Germany.

Ohman, M., Kalmari, J., Visala, A., 2008. XML Based Graphical User Interface Editor and Runtime Parser for ISO 11783 Machine Automation Systems. Proc. 17th IFAC World Congress, p.1578-1583. [doi:10.3182/20080706-5-KR-1001.00269]

Oksanen, T., Kunnas, A., Visala, A., 2011. Development and Runtime Environment for Embedded Controller Supporting ISO 11783 Standard. Proc. 18th IFAC World Congress, p.2912-2918. [doi:10.3182/20110828-6-IT-1002.02203]

Pereira, R., Godoy, E., Sakai, R., Cavani, F., Sousa, R., Porto, A., Inamasu, R., 2009. ISO 11783 Standard: Procedures for Serial Data Communication Between the Implement ECU with the Task Controller. CIGR Proc. of Technology and Management to Increase the Efficiency in Sustainable Agricultural Systems, p.1-19.

Spangler, A., Wodok, M., 2010. Development of ISO 11783 Applications in an Object Oriented Way Using the Open Source Library ISOAgLib. Available from http://www.isoaglib.com [Accessed on Sept. 30, 2010].

Speckmann, H., Jahns, G., 1999. Development and application of an agricultural BUS for data transfer. *Comput. Electron. Agric.*, **23**(3):219-237. [doi:10.1016/S0168-1699(99)00042-3]

Stone, M.L., McKee, K.D., Formwalt, C.W., Benneweis, R.K., 1999. ISO 11783: an Electronic Communications Protocol for Agricultural Equipment. Agricultural Equipment Technology Conf., p.1-17.