DESIGN AND IMPLEMENTATION OF VIRTUAL TERMINAL BASED ON ISO11783 STANDARD FOR AGRICULTURAL TRACTORS

W.Ham, T.Enkhbaatar, B.Luubaatar and K.Hyeokjae

Division of Electronic Engineering
Chonbuk National University
Jeonju, Jeonbuk, South Korea

ABSTRACT

The modern agricultural machinery most common use of the embedded electronic and remote sensing technology demands adoption of the Precision Agriculture (PA). One of the common devices is the Virtual Terminal (VT) for tractor. The VT's functions and terminology are described in the ISO11783 standard. This work presents the control system design and implementation of the VT and some Electronic Control Units (ECU) for agricultural vehicles based on the ISO 11783 standard. The VT development of the hardware and software systems is implemented in the advanced embedded system and by using the IsoAgLib open library. The advanced embedded system is made by the Samsung S3C6410 ARM11 core microprocessor based embedded board with CAN module and working environment is Windows Embedded CE 6.0 (WinCE6.0). The IsoAglib library provides several open sources to implement ISO11783 protocols. It's written by an object oriented C++ programming language. In this paper, we describe ISO11783 based tractor control system via Controller Area Network (CAN) and how to implement the embedded system. The paper also describes CAN-Bus device driver in WinCE6.0 and some modifications of IsoAglib for our target system. The target system consists of the VT, ECU of GPS and ECU of Sprayer for agricultural tractor. The ECU of GPS and ECU of light controller are implemented by using STM32F107F ARM Cortex-M3 based development boards.

Keywords: CAN-bus, ECU, Virtual terminal, Embedded System, ISO11783

INTRODUCTION

This technical article describes how to implement agriculture machinery control system based on ISO11783 standard. The standard of VT, which is used for the operation parameters setup of tractor and implements, is described in the part 6 of ISO11783. VT is an operator interface device that provided to allow display of information to operators and to allow operators to provide input information.

VT provides a common user interface to all working sets on the bus. VT contains a graphic display with limited set of graphical objects, a few soft keys with an icon on display, means to navigate on display and manipulate the values. How the display is shown in virtual terminal is stored in “object pool”. The object pool is a representation of a working set, and consists of objects supported in 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 of objects and all the parameters for each object. As soon as the working set (WS) is connected to the network and powered, the VT and WS start to communicate. After initial handshaking and requests, the WS starts to upload its object pool to VT, and display appears on the VT screen. If the mobile system contains more than one WS, the active display can be changed on the VT. (Stone et al 1999)

The ISO11783 standard has been jointly developed by tractor and implement manufacturer including AGCO, AGROCOM, DICKEY-Jonh, John Deere, Siemens and Fendt. These manufacturers have also created a specification defining how this standard should be interpreted. This specification is commonly known as ISOBUS. Also several researcher teams work on the implementation of VT and ISO11783 standard. In domestic, we are cooperating with Korea Agriculture Company in the research and application of VT. The purpose of this technical article is to implement hardware design and software design of VT based on the ISO11783 standard.

In this paper, we propose to generic ISO11783 compatible implementation based on the IsoAgLib library. The IsoAgLib provides open source libraries to implement ISO11783 standard, and is written with Object oriented C/C++ language.

In this paper we introduce concept of device drivers in the WinCE6.0. The design and implementation of reliable device drivers is notoriously difficult and constitutes the main portion of system failures. We mainly consider device drivers for CAN 2.0B protocol and high speed Serial Peripheral Interface (SPI) bus. Because the S3C6410 microprocessor doesn’t have any CAN-bus interface, there have two high speed SPI interface. The CAN-bus module is consists of MCP2515 CAN controller and CAN-bus driver chip. The MCP2515 CAN controller communicates with the S3C6410 microprocessor by using SPI interface.

The remainder of this paper is organized as follows: Section 2 overview of ISO11783 standard, CAN protocol and IsoAglib open library; Section 3 describes the implementation of hardware design; Section 4 describes the implementation of software design; Section 5 describes the experimental and results; and finally Section 6 discusses future work and conclusions.

BACKGROUNDS AND METHODS

Control Area Network (CAN) protocol

CAN network is that each message is preceded with an identifier that is unique to the transmitting controller and that multiple controllers can communicate over a single two-wire bus. CAN transmitting data in frames containing a header and 0 to 8 bytes of data. There are three separate CAN standards: CAN version 1.0, Version 2.0A (Standard CAN), and version 2.0B (Extended CAN). The main difference in these standards is the length of the identifiers that precede each message. The original specifications (Version 1.0 and 2.0A) specify an 11 bit message identifier. The Version 2.0B Extended Frames contain a 29-bit identifier which allow over 2^{29}-1 message identifiers. The 29-bit identifier is made up of the 11-bit identifier ("Base ID") and the 18-bit Extended Identifier ("ID Extension"). The figures 1 shown in difference of three CAN standards. In our research work presented in this paper based on the CAN2.0B standard.

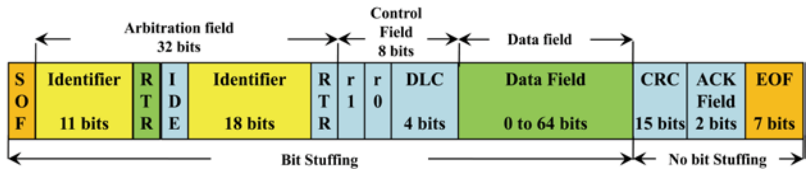


Figure 1. Frame structure of CAN 2.0B

ISO11783 overviews

The ISO11783 as a whole specifies a serial data network for control and communications on forestry or agricultural tractors and mounted, semi-mounted, towed or self-propelled implements. Its propose is to standardize the method and format of transfer of data between sensors, actuators, control elements, tractor or implement and etc. The ISO11783 standard is sometimes called as ISOBUS. It consists of several parts: general standard for mobile data communication, physical layer, data link layer, network layer, network management, virtual terminal, implement messages applications layer, power train messages, tractor ECU, task controller and management information system data interchange, mobile data element dictionary, diagnostic and file server. The figure 2 describes typical ISO11783 network topology.

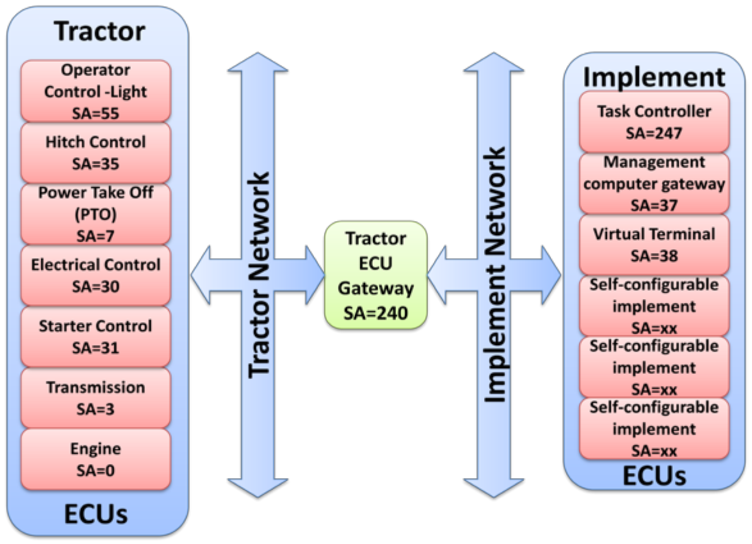

Tractor
Operator Control -Light SA=55
Hitch Control SA=35
Power Take Off (PTO) SA=7
Electrical Control SA=30
Starter Control SA=31
Transmission SA=3
Engine SA=0
ECUs
Tractor Network
Tractor ECU Gateway SA=240
Implement Network
Implement
Task Controller SA=247
Management computer gateway SA=37
Virtual Terminal SA=38
Self-configurable implement SA=xx
Self-configurable implement SA=xx
Self-configurable implement SA=xx
ECUs

Figure 2. The typical ISO11783 network topology. This figure describes the typical ISO11783 network topology.

The ISO11783 standard is not yet widely used in the Korean agricultural companies and researchers. The investments in the IT and Agriculture project is necessary to reach the international standard.

ISOAgLib open source library

In this section we introduce the ISOAgLib open source programming library, which is developed by Munich University and OSB & IT engineering company in Germany. It is suitable for embedded communication software in electronic control units (ECU) such as virtual terminal (VT) or task controller (TC) and File Server (FS). All functions according to the ISO11783 standards as well as the established machine interfaces are already implemented in the library. The figure 3 describes in the system architecture of ISOAgLib programming library.

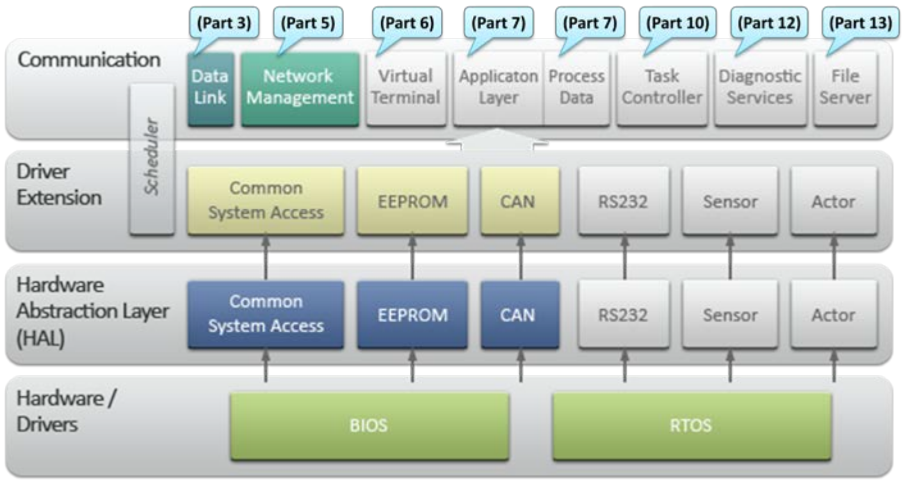


Figure 3. System architecture of ISOAgLib open library.

The IsoAglib library consists of several parts: Communication, Scheduler, Driver Extension, Hardware Abstraction Layer and Hardware/driver. This library allows building ISOBUS compatible equipment without the protocols implementation contained in this standard. Nowadays, hosting and maintaining IsoAglib are carried by OSB & IT Company.

The IsoAglib was developed to be compatible with various systems, and these systems can be composed of microprocessor, memory, Human Machine Interface and interface with CAN-bus. Because of this, the IsoAglib is divided into two parts: the library itself and Hardware Abstraction Layer (HAL). The library provides the Address claim (AC) negotiation of an ECU to the CAN-Bus. It also implements the transport protocols that are used in the initialization with VT and TC. The HAL is responsible for communicating with the operating system or BIOS device that is running the application, as can be seen in figure 3. The

IsoAglib library based embedded devices easily implement in the Linux and Windows operating system.

They develop so many versions of IsoAglib programming library, some tutorials and examples useful for studying ISOUS. In our case we are studying version 2.2-rc5 of IsoAgLib, because this version has some tutorials and examples. Especially we studied CAN server application program, ECU of GPS sensor, ECU of Display, ECU of DataSource, ECU of TractorBridge, and etc. The previous ECUs tested in Windows XP environment, it's shown in figure 4. The testing whole of IsoAglib programs execute in console application program, you can see whole transmitted and received CAN messages. The IsoAglib resources and projects made by several development environments, for example you can use DEV CPP free development tool, GNU and Microsoft Visual Studio C++.

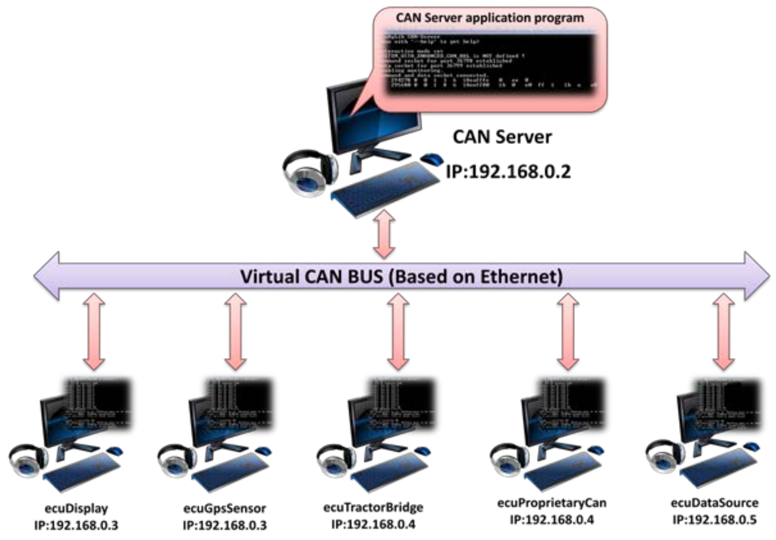
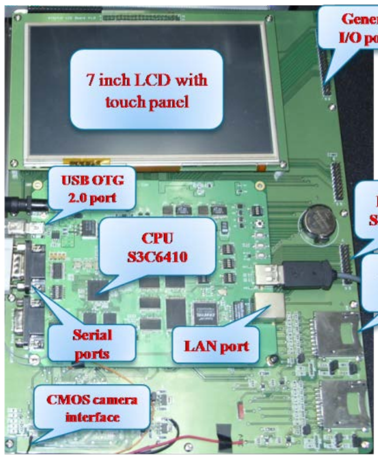


Figure 4. This figure describes in the simulation of ISOAgLib open sources in Windows XP, which consists of several virtual ECUs working with CAN server application program.

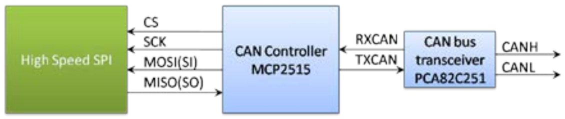
In our research team focusing on real hardware implementation in the advanced embedded systems, which are Windows CE and Firmware level programming for several ECUs. The next sections explain implementation of both environments.

IMPLEMENTATION OF HARDWARE

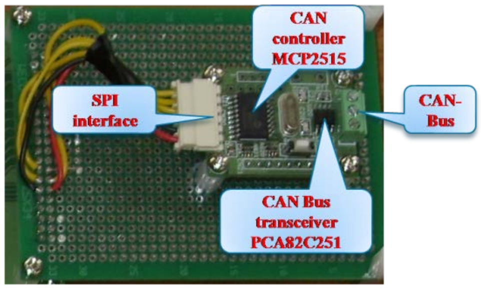
This section describes the hardware implementation of VT and sample ECUs based on IsoAglib and ISOBUS. The main part is implementation of CAN-bus in the embedded system. The hardware of VT is consists of the advanced embedded board and the CAN-Bus module.



(a)



(b)



(c)

Figure 5. This figure shown in the implementation of VT, which is consists of fig 5.a advanced embedded board, fig 5.b the block diagram of CAN-bus module, and fig 5.c the prototype board for CAN-Bus module.

The all hardware schematic and Prototype Circuit Board (PCB) developed our research team and implemented. The Samsung S3C6410 32 bit ARM11 Core (667Mhz) microprocessor is integrate several useful interfaces, which are the high-speed SPI, 2D/3D graphics acceleration, USB2.0 OTG, Ethernet, Camera interface, and etc. Since the S3C6410 microprocessor doesn't have CAN-Bus interface, we add the MCP2515 CAN microcontroller in our system. The MCP2515 CAN controller has SPI communication interface. The figure 5.b and 5.c describe the design of CAN-bus module and implemented prototype.

The hardware of other ECUs implemented on the 32-bit microcontroller STM32f107VC development board. The STM32F microcontroller's main Central Processing Unit (CPU) architecture is Advanced RISC Machine (ARM) V7 Cortex-M3, its support several useful peripherals. The one of useful peripheral is CAN controller, and it supports CAN 2.0A and 2.0B active and passive with data rates up to the maximum 1Mbit/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 message timestamp in the last two data bytes of the CAN message packet. The next most important feature of a CAN controller is it's receives message filtering. Because 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.

IMPLEMENTATION OF SOFTWARE

This section describes how to implement in the firmware level programming and the system level programming. The proposal implementation of software environment is shown in the figure 6, where the application program of VT-working with real ECUs by using CAN-bus and the virtual ECUs-using Ethernet network are depicted. The application program of virtual terminal runs on WinCE6.0 and access to CAN-bus with CAN device driver. The virtual ECUs application programs work on Windows XP environment on the personal computer (PC). It's very helpful for analysing the software working processes.

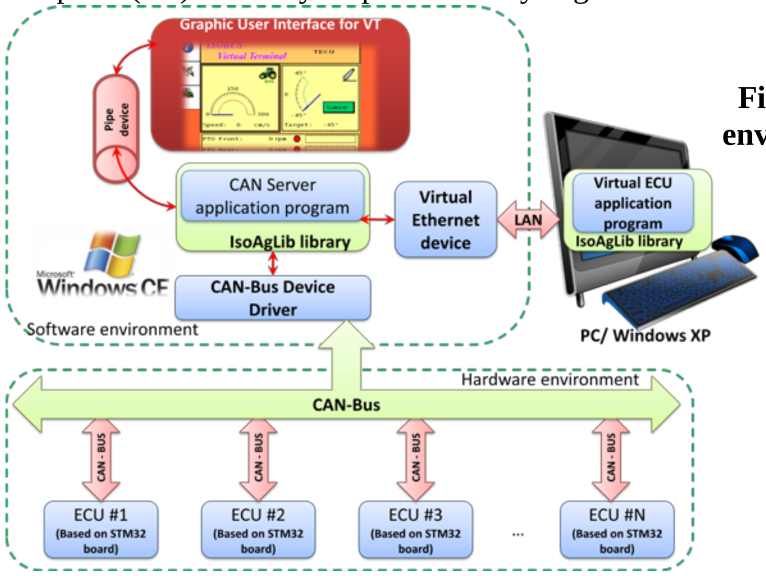
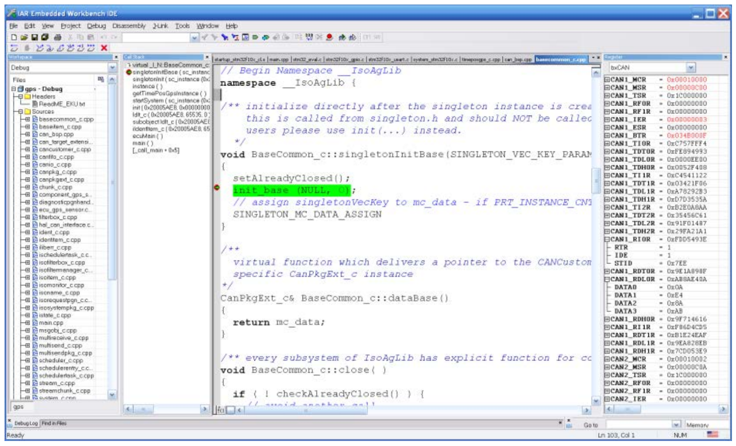

Graphic User Interface for VT
Virtual Terminal
Pipe device
CAN Server application program
IsoAgLib library
Virtual Ethernet device
LAN
Virtual ECU application program
IsoAgLib library
Windows CE
CAN-Bus Device Driver
Software environment
PC/ Windows XP
Hardware environment
CAN-Bus
CAN - BUS
ECU #1 (Based on STM32 board)
ECU #2 (Based on STM32 board)
ECU #3 (Based on STM32 board)
...
ECU #N (Based on STM32 board)

Figure 6. Develop environment of VT

Programming methodology for firmware level

In the firmware level programming CPU executes only one process; no operating system concept is there. If the hardware system not working with any operating system this time programming code should be write in the firmware level.

Due to the most embedded system developers use standard C programming language, some limitation exist in complicated system. ISO11783 standard implementation is not simple project for programmers. The IsoAgLib library's source codes written with the object oriented programming language C++, and it can easely solve some complicate protocols and etc. Because of the C++ programming can support Standard Template Library (STD), it can make templates and namespaces for lists, queue, vectors, and etc. Other important thing is development tools for embedded system. The widely used embedded development tools are a compiler and a linker proposed for C and assembler codes. The one of famous tools is IAR Embedded Workbench, which supports C/C++ and assembler for ARM7/ARM11 core processors. By using this tool, we successfully implement the ECU of GPS and ECU of Sprayer codes based on IsoAglib library in the IAR Embedded Workbench environment. IAR Embedded Workbench tool works with J-Link universal debugger device. The sample hardware debugging process is shown in the figure 7.



```
// Begin Namespace __IsoAgLib
namespace __IsoAgLib {

/** initialize directly after the singleton instance is crea
    this is called from singleton.h and should NOT be callec
    users please use init(...) instead.
  */
void BaseCommon_c::singletonInitBase(SINGLETON_VEC_KEY_PARAM
{
  setAlreadyClosed();
  init_base (NULL, 0);
  // assign singletonVecKey to mc_data - if PRT_INSTANCE_CNT
  SINGLETON_MC_DATA_ASSIGN
}

/**
  virtual function which delivers a pointer to the CANCustom
  specific CanPkgExt_c instance
*/
CanPkgExt_c& BaseCommon_c::dataBase()
{
  return mc_data;
}

/** every subsystem of IsoAgLib has explicit function for cc
void BaseCommon_c::close( )
{
  if ( ! checkAlreadyClosed() ) {
```

```
CAN1_MCR   = 0x00010000
CAN1_MSR   = 0x00000C00
CAN1_TSR   = 0x1C000000
CAN1_RF0R  = 0x00000000
CAN1_RF1R  = 0x00000000
CAN1_IER   = 0x00000003
CAN1_ESR   = 0x00000000
CAN1_BTR   = 0x034B000F
CAN1_TI0R  = 0xC757FFF4
CAN1_TDT0R = 0xFE894993
CAN1_TDL0R = 0x0000EE00
CAN1_TDH0R = 0x0052F408
CAN1_TI1R  = 0xC4541122
CAN1_TDT1R = 0x03421F06
CAN1_TDL1R = 0xA78292B3
CAN1_TDH1R = 0xD7D3535A
CAN1_TI2R  = 0xB2E0A8AA
CAN1_TDT2R = 0x35456C61
CAN1_TDL2R = 0x91F01487
CAN1_TDH2R = 0x29FA21A1
CAN1_RI0R  = 0xFDD5493E
  RTR      = 1
  IDE      = 1
  STID     = 0x7EE
CAN1_RDT0R = 0x9E1A898F
CAN1_RDL0R = 0xAB8AE40A
  DATA0    = 0x0A
  DATA1    = 0xE4
  DATA2    = 0x8A
  DATA3    = 0xAB
CAN1_RDH0R = 0x9F714616
CAN1_RI1R  = 0xF86D4CD5
CAN1_RDT1R = 0xB1E24EAF
CAN1_RDL1R = 0x9EA828EB
CAN1_RDH1R = 0x7CD053E9
CAN2_MCR   = 0x00010002
CAN2_MSR   = 0x00000C0A
CAN2_TSR   = 0x1C000000
CAN2_RF0R  = 0x00000000
CAN2_RF1R  = 0x00000000
CAN2_IER   = 0x00000000
```

Figure 7. Object oriented C++ codes for the implementation of GPS ECU by using the IAR Embedded Workbench.

Programming methodology for system level

The system level programming means working with any operating systems, in our case WinCE6.0. In this section we consider the device driver for CAN-bus and application program for VT in WinCE6.0.

Programming for 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 necessity of thorough knowledge about 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 architecture type in Win CE6.0 is a layered device driver structure. In this architecture, a driver consists of two parts, one is the Model Device Driver (MDD) library and the other is Physical Device Driver (PDD) library.

The operating system boot-up then configure device drivers for used system periprehals. It means CAN-bus device driver already initialized in the system starting. The application program of VT just call device driver DLL file (our case CAN.dll) from kernel. VT should access operating system based on the device driver for CAN module. The DLL port functions realized by CAN stream driver are shown: CAN_Init, CAN_Deinit, CAN_Open, CAN_Close, CAN_Read, CAN_Write, CAN_Seek, CAN_IOControl, CAN_PowerDown, and CAN_PowerUp. The figure 8 describes how to access application program to CAN-Bus device driver.

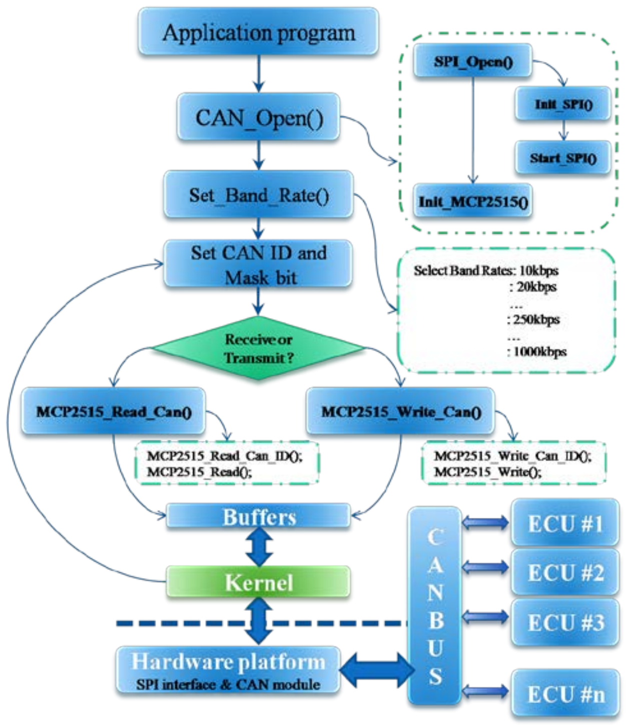


Figure.8 Sample flow chart for the application program of VT working with CAN-bus device driver.

Programming for VT application

The application program of VT implemented based on the IsoAglib open source library and codes. The VT working process and sample design standardized in the ISO11783 standard. VT related codes implemented in the IsoAglib library, but nowadays OSB&IT engineering company take care of all source codes and library. They use some tool for design of VT program, which is the vt-designer. It's commercial programming tool and annual fee of 1,450 Euros. Therefore we don't use the vt-designer tool and analyse source codes for VT2ISO tool. It 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 XML parser, which is Xerces-c_2_5_0D.dll file. The figure 9 describes the block diagram of VT application.

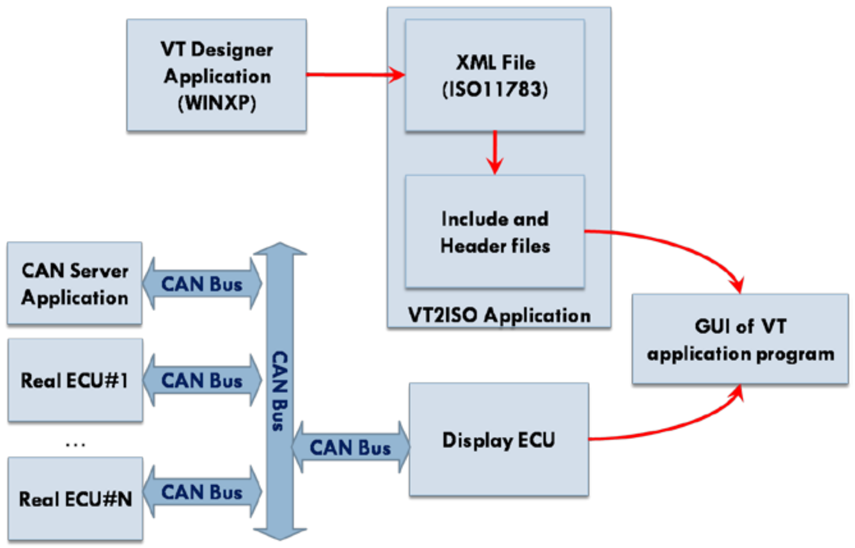


Figure 9. This block diagram shows experimental simulation of the implementation for ISO 11783 system.

EXPERIMENTS AND RESULTS

The main propose of this work is to implement VT application program with hardware based on ISO11783 and IsoAglib library. The figure 10 is shown in the implementation of whole embedded system for VT.

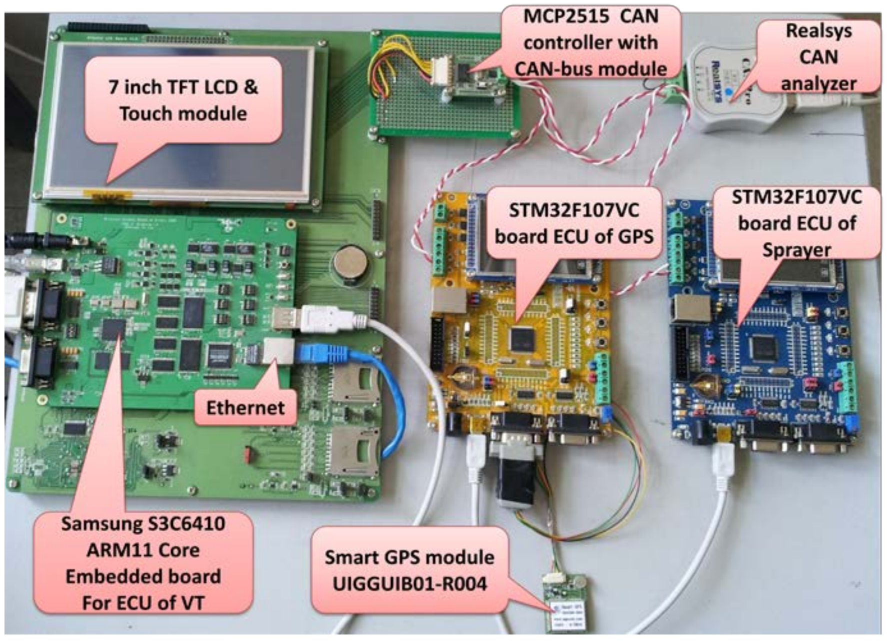

7 inch TFT LCD & Touch module
MCP2515 CAN controller with CAN-bus module
Realsys CAN analyzer
STM32F107VC board ECU of GPS
STM32F107VC board ECU of Sprayer
Ethernet
Samsung S3C6410 ARM11 Core Embedded board For ECU of VT
Smart GPS module UIGGUIB01-R004

Figure 10. Whole embedded system consists of the VT, ECU of GPS and ECU of Sprayer.

In the ISO11783 standard implemented by CAN protocol, which has messages of ECUs initialization, data exchange and status during the process, as shown in Table 1.

Table 1. Sample messages of ECU GPS, ECU Sprayer and VT.

Message	PGN			SA	DA	DLC	Data							
VT Address Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
GPS Address Claimed	00	EE	00	1C	All	8	25	B3	FF	E8	00	17	00	A0
Sprayer Address Claimed	00	EE	00	80	All	8	2F	B3	FF	E8	00	84	0C	A0
Request	00	EA	00	26	All	3	00	EE	00					
VT Address Claimed	00	EE	00	26	All	8	02	00	A0	E8	00	1D	02	A0
GPS Address Claimed	00	EE	00	1C	All	8	25	B3	FF	E8	00	17	00	A0
Sprayer Address Claimed	00	EE	00	80	All	8	2F	B3	FF	E8	00	84	0C	A0
VT Language Code	00	FE	0F	26	--	8	64	65	40	00	00	00	FF	FF
Sprayer Working Set Master	00	FE	0D	80	--	8	01	FF	FF	FF	FF	FF	FF	FF
Sprayer to VT	00	E7	00	80	26	8	C0	FF	00	00	00	00	FF	FF
VT to ECU	00	E6	00	26	All	8	FE	80	64	00	A0	0F	00	FF
GPS Position Data	01	F8	05	1C	26	8	86	00	00	FF	FF	FF	FF	FF
...														

CONCLUSIONS

With the development of this system, we check the possibility of using the IsoAgLib open source library is to implement the real implementation of ECUs for agricultural machinery. Then we successful implement the software of VT application and several useful ECUs based on IsoAglib library, also real hardware system based on advanced embedded boards. In this work, we use two kinds of the embedded boards. Also, we implement device driver for high speed SPI interface and MCP2515 CAN controller in the WinCE 6.0 operating system. Our future work is an improvement of application program for VT based on the technical specification of ISO11783. Our developed product with the standardization of the communication between tractor and implement gives convenience for tractor drivers and farmers. In our future work, we are going to develop application program that can be used for development of any proposed ECU of an agricultural tractor (for example ECU of Sprayer, ECU of Data Source, ECU Auxilirary Sensor, ECU Tractor Bridge, etc.) and development of our virtual terminal.

ACKNOWLEDGEMENT

This work was supported by the Post BK21 Project of Chonbuk National University and the Center for IT Convergence Agricultural Machinery (ITAM) grant (NO. R09-6)* funded by the Ministry of Knowledge Economy,Republic of Korea.

REFERENCES

Spangler, A. and Wodok, M. 2010. IsoAgLib – Development of ISO 11783 Applications in an Object Oriented way. In: *http://www.isoaglib.com*.

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 Conference, Louisville, Kentucky*. 7-10 February 1999.

ISO. 1998. Part 3: Data link layer. ISO 11783. *International Standard*, First edition 1998-07-01.

ISO. 2001. Part 5: Network management. ISO 11783. *International Standard*, First edition 2001-05-01.

ISO. 2004. Part 6: Virtual terminal. ISO 11783. *International Standard*, First edition 2004-06-15.

ISO. 2002. Part 9: Tractor ECU. ISO 11783. *International Standard*, First edition 2002-07-01.

ISO. 2007. Part 10: Task controller and management information system data interchange ISO 11783. *International Standard*, First edition 2007-03-19.

Seong-Min Kim, Seung-Jae Park, Cheol-Soo Kim and Myeong-HO Kim. 2011. Implementation of the communication model for ISO11783 standards based on AUTOSAR. ASABE Annual International Meeting.