Preprints of the 18th IFAC World Congress
Milano (Italy) August 28 - September 2, 2011

Development and Runtime Environment for Embedded Controller supporting ISO 11783 Standard

Timo Oksanen, Antti Kunnas and Arto Visala

Aalto University, Department of Automation and Systems Technology,
Otaniementie 17, 02150 Espoo, Finland (Tel: +358 9 4702 5562; e-mail: timo.oksanen@tkk.fi).

Abstract: ISO 11783 is becoming common to realize communication within a tractor implement system in an agricultural work machine. The ISO network minimally contains at least a tractor, a virtual terminal and a working set, which typically represents a single implement. The virtual terminal is a common device for all implements in the cabin and all the implements may upload their user interface to that. The developing process of an embedded controller that is capable of communicating within the network and use virtual terminal has been found to be challenging due to ISO standard complexity. In order to lower development costs, an objective is to provide development and runtime environment with which is quick and easy to design an embedded controller for agricultural machines. In the paper, this issue is discussed and the paper presents a software integration approach to reach the objective. The main parts of tool chain are PoolEdit (an open source ISO 11783 display editor), Matlab Simulink with C-code generation, and Visual Studio with Windows CE embedded target. The paper describes the tool chain and discusses challenges related to software integration. The paper presents also one machine controller implementation that was made with the presented tool chain, an ISO 11783 related test implement.

Keywords: embedded systems, control systems, function blocks, agricultural machines

1. INTRODUCTION

ISO 11783 is a standard developed for agricultural and forestry machines, to standardize communication between a tractor and implements (tools) connected to the tractor. The bottom layers of the communication are based on the CAN bus, and some additional layers are common to the SAE J1939 communication network, and ISO defines application layers for devices in the network. The core devices are tractor electronic control unit (TECU), virtual terminal (VT), and implement controller(s) forming a working set (WS). Additional devices on the bus are task controller (TC), positioning device (GPS), sequence controller (SC) and auxiliary inputs (AUX). The market name for the standard is "ISOBUS", and all the devices passed the conformance test can be marketed as ISOBUS compatible.

Virtual terminal provides a common user interface to all working sets on the bus. Virtual terminal contains a graphic display with a 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. (Stone et al 1999)

As soon as the working set is connected to the network and powered, the virtual terminal and working set start to communicate. After initial handshaking and requests, the working set starts to upload its object pool to virtual terminal, and the display appears on the virtual terminal screen. If the mobile system contains more than one working set, the active display can be changed on the terminal. For online operation, the standard provides messages to interact from VT to WS and vice versa, like an event message that soft key has been pressed or a needle position change message.

The variety of machine types and models is large in agricultural implements. On the other hand, the number of the produced machines of one model may be quite small, compared with tractors or cars for instance. Therefore an effort put on developing an embedded controller for each machine type needs to be small in order to keep development costs low. One way to reduce costs is to use a common controller platform and configure it in each machine. In ISO 11783 network the common parts are especially network communication protocols and application layers. In addition to real-time requirements, working safety and reliability are needed as well. In this paper, a framework is proposed to generic ISO 11783 compatible implement controller.

Öhman et al (2006, 2008) used RTI Constellation software with PoolEdit display editor to reach the objective. Constellation was an UML based tool which was especially designed for developing an embedded control system. Constellation provides a framework for building control systems from reusable software components. In the

implementation, the framework was not only a realization of protocol layers, but it also defined common machine modes (transport, manual, field mode) for all implements.

The other attempt to reach goal is IsoAgLib project. IsoAgLib provides open source libraries to implement ISO 11783 protocols. "ISOAgLib is an Open-Source programming library. As part of an ISOBUS-system, it takes over all functions embedded in an electronic communication system according to the standard ISO 11783 such as display of user interfaces on a Virtual Terminal or Task controller." Together with "vt-designer" sold by OSB AG this offers a tool chain to develop embedded controllers. IsoAgLib is written with C/C++ language. (IsoAgLib 2010)

2. REQUIREMENTS

A development environment has to provide means to design a user interface (VT displays), to connect behaviors with user interface events and to program functionalities to a system. The user interface development and functionality development go hand in hand. Some modern automation development systems integrate these requirements into single software, like NI LabVIEW. However, if no integrated development is available, the requirements can be obtained by software integration.

One of the requirements is to offer ready to use function blocks for a system engineer who does not need to be familiar with all the bit level issues in ISO 11783 standard. The function block approach offers graphical programming benefits, and easy integration to functionalities and control engineering.

Development and runtime environment should fit together, in order to allow advanced debugging and analysis tools. Simulation is a standard phase in product development and the tool chain should support it. Also remote debugging of code running on the target should be available.

3. MATERIALS AND METHODS

The proposed tool chain utilizes software integration. The main parts are 1) PoolEdit (PoolEdit 2010); 2) PoolEditParser (PoolEdit 2010); 3) Matlab, Simulink, Stateflow, Real Time Workshop, Embedded Coder, Stateflow Coder (Mathworks); 4) Visual Studio 2005; 5) Windows Embedded CE 6.0 compiler.

The selected hardware supporting the tool chain was Toradex Colibri running Windows Embedded CE 6.0. The hardware contains ARM4i processor (Marvell PXA320, 806MHz), an integrated CAN transceiver, Ethernet and 1GB integrated flash memory. The selected hardware provides a powerful processor, and a real-time operating system, and also Ethernet based communication allows remote debugging.

3.1 PoolEdit

PoolEdit is an open source, XML based, graphical editor for developing ISO 11783 graphical user interfaces. The editor was developed at Helsinki University of Technology as a part of the Farmix project. PoolEdit is written in the Java programming language. The editor implements a multiple document interface and it has been tested on both Windows and Linux operating systems. The editor interface is shown in Figure 1. (Öhman et al 2008)

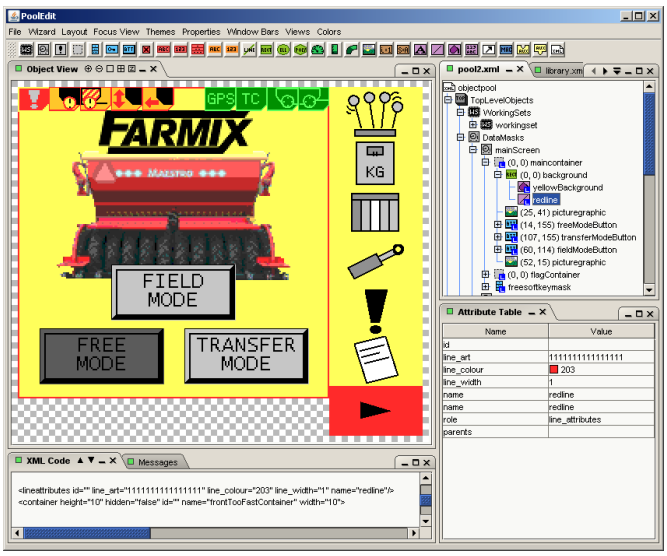

Figure 1. PoolEdit (Öhman et al 2008)

PoolEdit uses XML file to store the design. The native format is "PoolEdit XML" which is based on the IsoAgLib XML format (Spangler and Wodok, 2007). The biggest difference is that in PoolEdit XML only special link elements are used for linking, while in IsoAglib XML many attributes are also used for linking. In PoolEdit XML links can only point to objects on the root level and root level names have to be unique. (Öhman et al 2008)

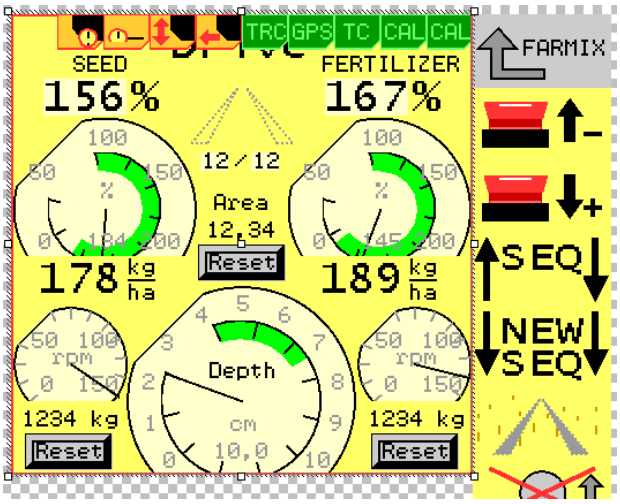

Figure 2. Example of data mask in seed drill display

Embedded XML is designed so that it can be transformed to ISO 11783 binary format and sent to a virtual terminal. It is derived from the PoolEdit XML format, but there are a few differences. Embedded XML files include base64 encoded bitmap data which removes the need for separate image files. In addition, every object has been given a unique object ID which makes subsequent processing very efficient. (Öhman et al 2008)

An example of typical screen is shown in Figure 2, which represents a drive display of seed drill. The display contains

some needle meters, numeric values, buttons to reset counters, and soft keys to operate machine.

3.2. PoolEdit parser

PoolEditParser is a tool with which the Embedded XML from PoolEdit can be converted to C-file header. In this case, the PoolEditParser was modified to produce Matlab m-file script that defines the object pool as a byte array in Matlab. The same parser also generates enumeration for object id's that provides easy usage in later phases.

The other parser generates a Simulink-type library from the Embedded XML object pool. The parser generates a function block into a Simulink-type library that correspond an object in the pool. At this phase of development, only the most common object types are supported: a meter, an arched bar, a linear bar, an output number field, an input number field, an input boolean, a button and soft keys. An example of a generated library is shown in Figure 3.

The function blocks generated into the library contain only object id's, parameters (for refresh rate for instance) and link to the realization. The realization of function blocks are in a common library.

This creates a one-way link from user interface editing to functionality editing. This does not allow two-way editing, that could be reached with integrated development environment. Simulink library format is based on ASCII-text and the blocks and wiring are defined using specific structure.

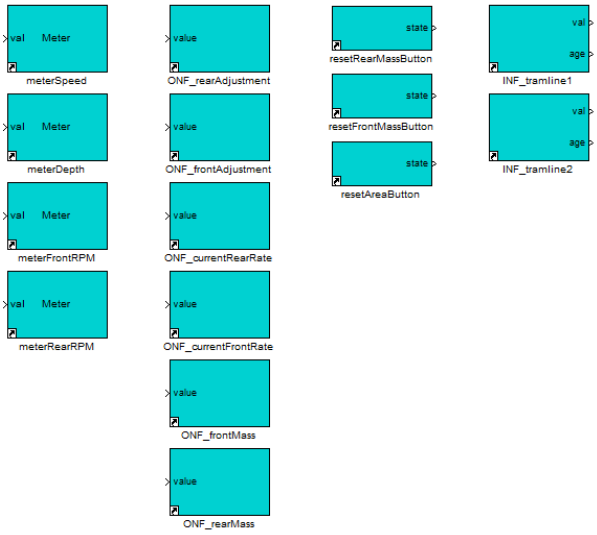

Figure 3. Simulink function blocks generated from Embedded XML.

3.3 Simulink framework

For any ISOBUS controller, the ISO 11783 parts 2-5 are common, including the address claiming and transport protocol. The address claiming and the lower level network management are programmed into one state machine. Diagnostics (part 12) is also considered as a lower level functionality that has to be supported in each ISO 11783 ECU.

In typical working set master up to three higher level layers have to be considered: virtual terminal, task controller and auxiliary inputs/functions. Interface to virtual terminal contains connection management, like requesting the properties of terminal (like resolution) and user settings (like language); and upload the object pool.

3.4 Function blocks for VT objects

The function blocks take care updating information between virtual terminal. For output type objects like meters, the designer of the control systems only needs to set minimum and maximum refresh rates for updating. The function blocks communicate at Simulink level with VT manager and communicate at the hidden layer to the network. An example of a function block interior is shown in Figure 4.

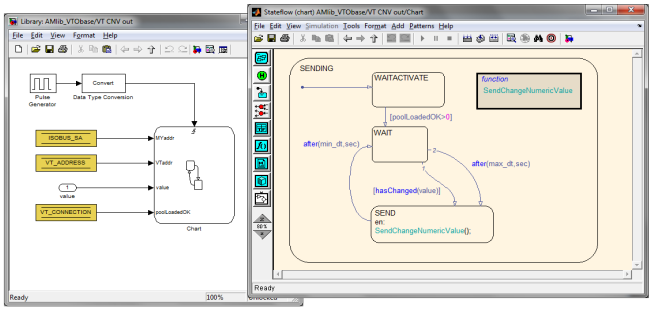

Figure 4. Realization of function block (output number field)

3.5 Hidden layer connecting Simulink blocks

CAN messaging is handled in a hidden layer. The communication between Simulink layer and hidden layer is done with C-functions, and function pointers. Inside the hidden layer the CAN messages are done by C-functions and callback functions. The blocks may register to global CAN receiver to receive messages from the network, and the incoming messages are delivered to all function blocks registered there. The filtering of messages is done at function block level, in order to keep the callback procedure simple.

The technology to realize hidden layer is Simulink "inline S-function". With basic S-function it is possible to connect blocks written with another programming language (like C++ or Fortran) to Simulink and run the simulation; the source code is compiled into Simulink S-function block. However, when using C-code generation from Simulink model, it is not possible to regenerate C-code unless it is specifically prepared. The solution to do the trick is to use inlining, when the self-written C-code will work both in simulation and in code generation. All the hidden layer functions are written in inline way, so it is possible deploy the code directly.

3.6 C-code generation and runtime

The address manager, the virtual terminal communication manager and tractor interface, virtual terminal objects, machine modes including I/O calibration and control loops are all programmed in Simulink. Only some low level functions, like conversion to the ISO 11783 frame format and unpacking of message data are done with C-code. The Simulink model is converted into C-code by using Real-Time

Workshop. The sample time of control system can be changed depending on machine requirements, the default is 10ms.

The generated code is integrated into small runtime code in Visual Studio 2005. The compiler does not make much difference; any other can be used too. In runtime three parts are crucial: CAN-hardware abstraction (driver for appropriate chip), direct I/O handling and real-timing of the main control loop. In the example, the amount of written C-code for runtime is about 250 rows, compared with generated code for functionality which is more than 16000 rows.

The embedded target in the project was Toradex Colibri with a PXA320 processor and preinstalled Windows Embedded CE 6.0. Visual Studio 2005 supports tool chain for this operating system and hardware manufacturer provides SDK for this target, so development in this level of tool chain is straight-forward. Toradex Protea carrier board uses a Philips SJA1000 CAN controller and the driver is provided by the manufacturer.

3.7 Remote debugging

In the presented tool chain there are three levels of debugging available. The highest level is the simulation, as Simulink is originally a simulation environment for control systems and operation of the system can be tested and verified at this level. The lowest level is in Visual Studio that allows remote debugging of the C-code running on target.

The third level is to use Simulink to follow code running on the target; for this purpose Real-Time Workshop provides "External mode" that requires certain C-files to be integrated in the project. When this mode is enabled, Simulink can connect to the target by using TCP/IP (or serial cable). The External mode provides at least three benefits: 1) the remote analysis of control loops by offering data display and data logging; 2) the remote analysis of finite-state machines; and 3) remote tuning of parameters by using Matlab-interface. Mathworks tool chain does not provide TCP/IP stack officially for Windows Embedded CE 6.0, but the stack can be modified from Windows stack.

3.8 Summary

The presented tool chain is summarized in Figure 5.

4. A TEST IMPLEMENT

With the presented tool chain it is possible to develop an embedded control system into machine connected to the ISO 11783 network. The first implementation was made to "Agromassi Test Implement", in the Agromassi project. The Test Implement is used to test commanding of a tractor in ISO 11783 network. A remote control interface is an additional feature of an ISO 11783 tractor, and the question is that in which conditions the driver may let an implement to command the tractor. The Test Implement was developed to support the development of a user interface of a tractor where the driver authorizes an implement to command the tractor. The Agromassi Test Implement is shown in Figure 6.

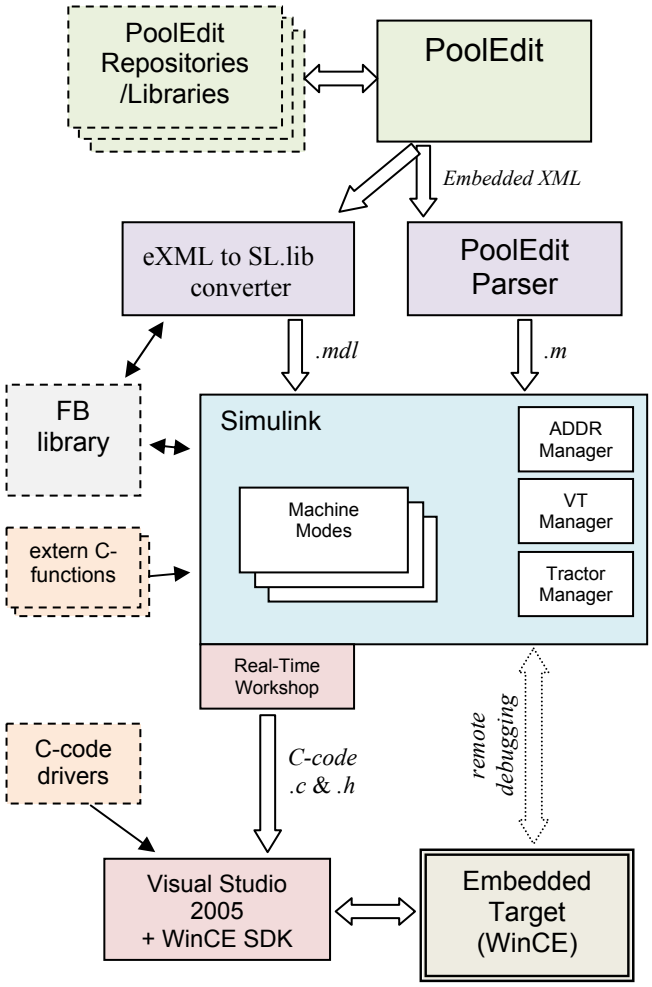

Figure 5. Summary of the presented tool chain.

Figure 6. Agromassi Test Implement.

The Test Implement has the following functions: markers on both sides, adjustable support wheels, a "drum" rotated with PTO making some noise, and a hydraulic motor powered "barrel" rotating in proportion to driving speed. The Test Implement is capable of commanding hydraulic valves, power take-off and hitch level in the tractor; as a part of the

control loop. The Test Implement operates in four modes: manual, transport, automatic, and TC direct. In automatic mode the Test Implement behaves like hitch connected seed drill, and in TC direct mode external simulator may cause trouble and unexpected behaviour to the machine.

The I/O of the machine is: a) measurements and states sent by a tractor; b) the remote controlled resources of a tractor (including hitch position, four hydraulic valves and PTO engagement); c) analogue position sensors in both markers and support wheels; d) a speed sensor in a hydraulic pump; e) a signal tower with four light elements; and f) two horns for indicating faults.

The Agromassi Test Implement was developed by using the presented tool chain. The Test Implement is equipped with a Toradex Protea and Colibri PXA320 controller and I/O is provided by custom-made electronics that connects Toradex with I^2C bus.

A user interface for an automatic mode is shown in Figure 7. The user interface is drawn by using PoolEdit. The display has meters that support changing of the background color and digital number, as well as hiding components. On top of the screen are flags showing online ISO 11783 devices. From display design the corresponding function blocks are generated, a partial presentation of the automatic mode is shown in Figure 8.

To support the development process, a realistic I/O simulation was developed for ECU, so before the machine was completed it was possible to test software, communication and functionality. I/O simulation in the ECU was found to be sufficient for developing a machine controller with this small functionality, but for more functionalities it would be better to have a full hardware-in-loop simulation possibilities.

In the implementation of Test Implement, no VT macros were used. With the use of macros it would be quicker to change displays, but on the other hand programming of functionality in two places (in VT and in ECU) would not make system engineering and analysis that straightforward. Still, the usage of macros will be studied in the future.

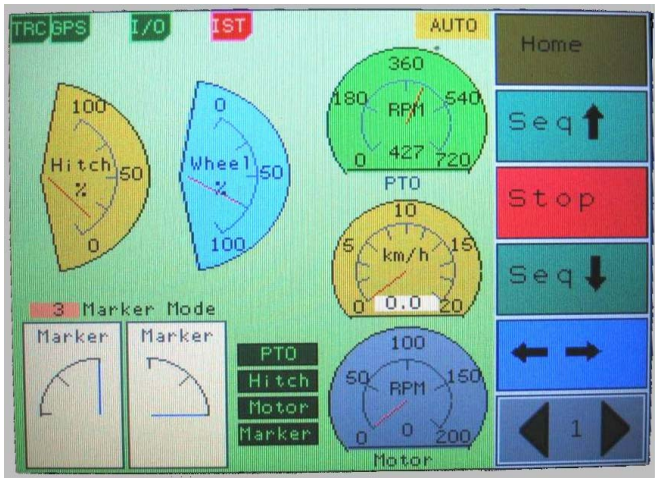

Figure 7. User interface of ”Agromassi Test Implement”.

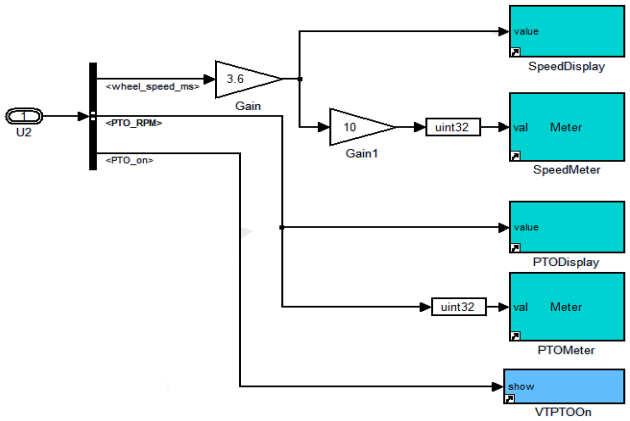

Figure 8. Partial realization of automatic mode.

5. CONCLUSIONS

Object oriented display architecture together with the CAN bus and dedicated protocol (ISO 11783) sets challenges to embedded development and runtime environment. The proposed architecture provides a function block approach to integrate user interface development and control programming. With an integrated tool, the most optimal solution could be reached, but the software integration provides a good way to do the work.

The proposed tool chain and runtime environment provides a good framework for any implement controller (working set master). The drawbacks are the fixed object pool size, which means that the runtime cannot actually adapt to the properties of virtual terminal but only select a proper version of pre-created object pools.

At this phase of development, the tool chain contains all the basic communication in ISO 11783, communication with virtual terminal, supporting most of the main objects in the display and support of tractor messages. However, the function blocks that would allow communication with a task controller or with an auxiliary input device are not ready yet.

The paper presented state-of-art software integration to realize ISO 11783 supporting embedded control system development environment that integrates user interface design and graphical higher level programming tools, and provides code generation for a real-time embedded target. The tool chain provides an intuitive design environment, remarkably simplifies using virtual terminal as a user interface compared with the traditional approach, by auto-generation of corresponding function blocks.

The paper showed one implementation that was made by using the presented tool chain. The implementation proved the concept and no major problems were found.

ACKNOWLEDGEMENTS

A part of this research is done in the project Agromassi that is part of FIMECC-program EFFIMA.

REFERENCES

ISO. (2004a). Part 6. Virtual terminal. ISO 11783. *International standard.*

PoolEdit (2010). Open Source XML ISO 11783 User Interface Editor. *http://autsys.tkk.fi/en/Farmix/PoolEdit* (accessed 2010-09-30).

Spangler, A. and Wodok, M. (2010). IsoAgLib – Development of ISO 11783 Applications in an Object Oriented way. *http://www.isoaglib.com (accessed 2010-09-30)*

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.

Öhman, M., Kalmari, J. and Visala A. (2008). XML Based Graphical User Interface Editor and Runtime Parser for ISO 11783 Machine Automation Systems. In proceedings of IFAC World Congress 2008. COEX, South Korea. 6p.

Öhman, M. and Visala A. (2006). Design and Implementation of Machine Control Systems with Modern Software Development Tools. In: Field and Service Robotics. Corke, P. and Sukkarieh, S. Ed, 377-388. Springer, Berlin.