Ada language for automation

Ada for Automation (A4A) is a framework for designing industrial automation applications using the Ada language. It makes use of Hilscher boards and can exchange process data via CANopen.

Today, control applications are getting more and more complex, involving motion control, databases to store recipes, parameters, alarms and events, networks to exchange information with other intelligent nodes (machine-to-machine communication), human-to-machine interfaces either rich or constrained, web user interfaces to allow interaction from abroad, and so on. Of course the situation won’t become simpler in the future with the Internet of Things or Industry 4.0. A wide range of automation products is available on the market and the borders between them are moving. For example, drives can come with an integrated PLC and a PLC can have encoder inputs and control loops. Some PLCs have integrated web HMI, some panels have a soft PLC.

Thus, the task of control engineers is becoming increasingly difficult, first to select the right combination of hardware and software and then to master all the necessary tools and languages. Languages created for control engineers like Ladder or Instruction List are not always adapted to the task at hand. Standards help maintain a kind of consistency and there is a chance to end up with a working solution, but at what price? Solutions are often like Frankenstein’s creature if you want them to be attractive: after some time, tools become unsupported and obsolete, if they are still available.

Most users of automation products fall either in the C/C++/C# industrial/embedded computer programmer or the IEC 61131-3 PLC programmer world. While coming from the world of computer science as a C programmer and then having learned object orientation using C++, I am a fan of the Ada language, which I think could be a very interesting bridge between those worlds. Users of the Structured Text (ST) language will find the syntax of Ada very similar and indeed they share same Pascal influence. Born in 1983 as a general purpose language suitable for long lived applications ranging from embedded systems to the largest ones, Ada evolved in 1995, 2005, and 2012 to provide every feature needed from low-level hardware interaction on bare metal to modularity, object orientation, tasking, and more, with a runtime system allowing the same application code to be compiled for any supported target Operating System. The main goals during the design phase of Ada were sustainability and reliability and there is no doubt industrial applications share the same requirements.

I wanted to learn the language and found that a framework for building automation applications making use of Hilscher boards would be great since I am a Hilscher France technical support guy. This is my own project, not endorsed by the Hilscher company.

Framework for design of automation applications

A4A is a framework for designing industrial automation applications using the Ada language. It makes use of the Libmodbus library to allow building a Modbus TCP client or server, or a Modbus RTU master. It can also use Hilscher communication boards allowing to communicate via fieldbuses like AS-Interface, CANopen, CC-Link, DeviceNet, Profibus, Ethercat, Ethernet/IP, Modbus TCP, Profinet, Sercos III, Powerlink, or Varan. Ada Core provides an open source IDE and compiler/runtime and a learning platform. Debian GNU/Linux provides the same IDE in an older version and many libraries one can use with Ada.

Thanks to Ada, an application built using Ada for Automation works with Microsoft Windows, Linux, and Linux with PREMPT_RT patches, down to the millisecond cycle time. It could also work on QNX or Vx Works without much, if...
any, code modification, since the Ada runtime takes care of providing the same services on all supported platforms. Thanks to GtkAda it can be built with a GUI. It can be built without of course. Thanks to Simple Components it can send you emails or talk to the cloud. Thanks to Gnoga, built upon Simple Components, it can be built with a Web GUI. It can also be built without one of course. Gnoga allows to manipulate the DOM of web pages, attaching to or creating HTML or SVG elements that can be animated from the Ada code. Ada for Automation is 100 % Ada, which makes all Ada resources, resources for Ada for Automation. Since Ada allows the use of any C library, Ada for Automation can also use them as well.

Some may consider Ada to be a language of the past but it seems pretty alive and in line with upcoming trends. “Over the past few years, Ada support on targets typically used by the automation industry has increased significantly. It started with a port to an 8-bit micro-controller, the AVR AT Mega256, in 2009, demonstrating that the Ada language was a viable choice for programming applications on low power low memory architectures. Since then, the main focus has been on ARM and in particular the ARM Cortex series. Bare metal cross compilers to Cortex-M and Cortex-R series were released in 2014, followed by Cortex-A microcontroller support for Embedded Linux, Raspberry Pi, VxWorks, and also Android and iOS. On the ARM micro-controller side of things, Ada support first appeared on community-available devices, such as the STM32F4 Discovery from ST, SAM4S from Atmel, and TMSS70 from TI. Since these beginnings, the set of devices with available board support packages and drivers has continued to expand, notably through Ada Core’s community Github. The CMSIS-SVD driver format provided by most hardware vendors has proved to be a significant help here. Ada specifications can now be generated directly for device memory and interfaces, providing solid foundations for driver development. As a successful demonstration of this effort, this year’s first Ada maker contest Make with Ada attracted more than 30 submissions, primarily targeting those ARM Cortex micro-controllers,” said Quentin Ochem, head of business development and technical account management at Ada Core. One of the bold features of Ada for Automation is brought by the Ada binding of the Hilscher cifX library, the Application Programming Interface (API), which allows the use of Hilscher communication boards.

**Data exchange via CANopen**

With any industrial communication protocol, data exchange is established either cyclically, for all process data like setpoints and measures, statuses and commands, or acyclically for configuration, parameterization and diagnostics. Taking CANopen as an example, the process data exchange is carried out by PDOs (Process Data Objects), while parameterization is done using SDOs (System Data Objects). In addition, services like node management are also provided as acyclic messages. Similarly, with Devicenet, which implements CIP (Common Industrial Protocol), assemblies carry such process data as implicit...
data, configured once during startup, while services like Get or Set Attribute allow so-called explicit access to parameters and configuration data.

For this reason, Hilscher has defined a generic interface, the Dual Port Memory (DPM), which has a standard layout independent of the protocol stack running on the netX system-on-chip. The DPM layout provides a process data image with input and output areas for the cyclic data exchange, and message boxes for sending and receiving packets for acyclic communication. Control and status areas for hardware and software identification, general commands, and diagnostics are also provided. The cifX API provides all necessary functions to access the DPM through the user interface of the cifX driver available on all major Operating Systems. Header files are provided in the C language.

With Ada for Automation, the user gets not only an Ada binding allowing the use in the Ada language but also a rather complete infrastructure. On top of the cifX API, an infrastructure has been built, which takes care of the driver initialization, hardware and firmware identification, cyclic refresh of the process image, watchdog management, monitoring of the communication status and more. This infrastructure also has a system managing acyclic communication with message queues, routing and function blocks for the user application program. Since it is necessary to be able to configure and diagnose those cifX boards, the Hilscher netX Diagnostic and Remote Access protocol has been integrated as well and allows the use of Hilscher’s configuration tool, Sycon, net via TCP/IP. As expected, several application examples are available which demonstrate the potential use of the infrastructure and, thanks to Ada, these applications are running on the supported operating systems without modifying a single line of code.

Since the Hilscher cifX API is available for most Hilscher products, including cifX boards, com X, Netrapid and Netjack modules, and also Nethost which is like a cifX board accessed via TCP/IP, Ada for Automation can make use of all those products and can be a CANopen Master or Slave, respectively a DeviceNet Master or Slave. Parallel to the CANopen communication, Hilscher boards also offer access to the CAN layer allowing to communicate with other Classical CAN devices (11-bit and 29-bit identifier) letting you implement other protocols like Layer Setting Services (LSS) for example or roll your own.

With Ada for Automation and Hilscher cifX boards, any industrial PC can become a powerful control target. But simple applications could benefit from the inexpensive CPUs available today on the market. Experiments have been conducted successfully using the Raspberry Pi with Netrapid modules via the fast SPI connection to the DPM and with Nethost via TCP/IP. The netX system-on-chip can have up to four communication channels. Hilscher has created boards with two CANopen channels, two DeviceNet channels, and even one CANopen and one DeviceNet channels. The cifX API allows up to four channels per board and there is no limit to the number of boards. On top of this communication infrastructure, the next step would be to implement Ada objects based on the objects specified by CiA in the CANopen profiles or by ODVA in CIP profiles. This could lead to an object library of devices implementing standard features like drives or encoders. Thanks to the object orientation and inheritance, a manufacturer could then provide a derived object implementing manufacturer specific features. Dreaming a little further, a motion control library like the one specified by PLCopen would allow a control engineer to build motion applications with standard components. I would love to create such CNC but using a CANopen drive, motor, and encoder. The project is of course looking for contributions and sponsors.
The Ultimate CAN FD Tool

neoVI FIRE 2

Vehicle Interface & Data Logger

Standalone gateway functionality between DoIP, CCP/XCP, CAN FD, CAN, LIN, and Automotive Ethernet.

Device includes:
- 8x ISO CAN FD
- 4x LIN
- Ethernet: DoIP/XCP
- Hardware Cybersecurity
- Standalone Logging, Scripting, & Simulation
- Full Vehicle Spy Software Support

Find out more at www.intrepidcs.com

INTREPID CONTROL SYSTEMS GMBH
USA Germany Japan Korea China India Australia
+49 (0)721 6633703 –4  icsgermany@intrepidcs.com