3rd International Conference on Innovations, Recent Trends and Challenges in Mechatronics, Mechanical Engineering and New High-Tech Products Development
MECAHITECH'11
International Conference
Bucharest, 22-23 September 2011

Design of a Mechatronic System for Application of Hardware-in-the-loop Simulation Technique

Daniel Chioran
Department of Automation / Technical University of Cluj-Napoca
15 C-tin Daicoviciu Street, 400020 Cluj-Napoca, Romania
daniel.chioran@yahoo.com

José Machado
CT2M Research Centre / University of Minho
4800-058 Guimarães, Portugal
jmachado@dem.uminho.pt

Abstract

Classical approaches, using Simulation analysis technique, use a controller model that can – or not – be coupled with a plant model. Usually, the controller and plant models are connected, in a closed-loop behavior, and this kind of Simulation is called Software-in-the-loop Simulation (SIL). However, recently, some directions are being assumed and some recent works deal with Simulation considering the real controller, instead of the controller model, in the closed loop behavior with the plant model and this kind of approach is called Hardware-in-the-loop simulation (HIL). In order to study and to propose some rules about the simulation of real-time systems considering HIL simulation, at the Automation Laboratory of the Centre for Mechanics and Materials Technologies of the University of Minho, Portugal – a workbench especially devoted to this study is being developed. This workbench considers an environment for Simulation, and the respective programming language, and a real controller that interacts with the simulation environment running on a PC. After looking at the available software tools and modeling languages, Dymola simulation environment and Modelica modeling language were chosen. The main reasons for this choice are associated with the unique multi-domain engineering capabilities of Dymola and Modelica that allow to deal, on the same environment, with many different engineering domains like hydraulics, power train, thermodynamics, air-conditioning, vehicle dynamics, mechanical, electrical, electronic, control, thermal, pneumatic, among others... As real controller, the choice was a Programmable Logic Controller (PLC) from OMRON company, the CPM2A model. This paper presents the first step, of this ongoing work, and is focused, mainly, on studying how to exchange information between a real PLC (used, as controller, on the designed workbench) and Dymola software that will run specific plant models, developed in Modelica language, on a Personal Computer.

Keywords

Hardware-in-the-loop simulation, Software-in-the-loop Simulation, Plant models, Safe controllers, Mechatronic systems.

Introduction

In recent years a significant increase in automated systems complexity can be witnessed. Building such systems is now a very complex task that requires the combined effort of engineers performing different tasks. Because of economic reasons, plants must be operational with no delays, working as planned the first time they are started. Obvious, this puts lots of pressure on the engineering team and some ways to ensure that these tasks are performed with no faults and no delays are needed.

The conventional way is to use manual testing methods but due to the increase in automated systems complexity, these methods have reached their limits. In this case, manual testing methods are not able any more to cover every possible system failure.

The new way is to build and simulate computer models of the plant before it is put into service. This is a big step forward and benefits are numerous. Design faults (inadequate components or links), hardware failures (components colliding) or software failures (control software bugs), plant critical states (building of overpressure / overheating in tanks) can all be identified and taken care of before the plant is built and started.

Simulation is one of the most used techniques for analyzing Industrial Safe Controllers in order to obtain safe behavior of Automation and Mechatronic systems. In this context, several approaches are used, with

different programming languages and software tools [1]. Usually controller and plant models are in a closed loop configuration [2].

Both Software in the loop (SIL) and Hardware in the loop (HIL) simulation approaches [3] are used in the industry and both of them have to deal with the Inter Process Communication (IPC) issue between system components. An early response to the IPC issue was introduced by Microsoft back in 1987, the Dynamic Data Exchange (DDE). As science and technology evolved rapidly, multiple solutions to this problem became available. This raised another issue as some of these were proprietary solutions, not widely available in the industry. In this context, in an international unified effort, the OPC (OLE for Process Control) specifications were developed to ensure a common communication interface for the industry to use.

This HIL approach has many advantages when compared with SIL approach because the simulation is performed with the real program running in the real controller. This can be very important, specially, when different timescales are considered from a real point of view, mainly in what concerns the real evolution of the controller and its timescale, running in real conditions of work. Of course that the interaction between the real controller and the plant model, that runs in a Personal Computer (PC), is still a complex and hard task to be performed, when real-time simulation and real-time behavior is critical for guaranteeing the safe behavior of the studied system [4].

This topic is of much interest in the industry so many of approaches to the issue of IPC within a HIL simulation system can be found. For this work, only those HIL configurations that include a plant modeled in Modelica [5] and simulated with Dymola [6] software were considered. In [7] and [8] a dedicated software link between a Dynamic Data Exchange (DDE) and OLE for Process Control (OPC) server was developed in order to exchange information among the components of a HIL system. Others tested the possibility of using the IPC method of „named pipes" as seen in [9].

A new and interesting real-time library developed for Dymola facilitates the communication with a PLC as described in [10].

In order to study and to propose some rules about the simulation of real-time systems considering HIL simulation, at the Automation Laboratory of the Centre for Mechanics and Materials Technologies at University of Minho, Portugal – a workbench especially devoted to this study is being developed.

This paper will present the first step, of this ongoing work, and is focused, mainly, on studying how to exchange information between a real PLC (used, as controller, on the designed workbench) and Dymola Software that will run specific plant models, developed in Modelica language, on a Personal Computer.

This apaper is divided into 6 sections. Section 1 introduces the reader to the context and work related to the simulation of automation systems and the IPC inside simulation systems. Goals that must be achieved through this work are also pointed out here. Section 2 presents the SIL and HIL simulation methods and also compares them. Section 3 presents the Modelica modeling language and one of its simulation environments, Dymola. Section 4 deals with the communication inside a HIL simulation system. It starts by presenting specification such as DDE and OPC and then goes to show how they can be implemented in industrial applications. Alternative solution such as communication using "named pipes" or using a special Dymola real time library are discussed towards the end of this section. Section 5 presents the outline of a easy to build HIL simulation workbench and looks at the IPC inside this system. The information is presented in grater detail. Section 6 draws conclusions about the study undertaken and possible future work.

Simulation of automation systems

In the context of modern plant engineering, simulation is the analysis method used by engineers for a number of different tasks.Regardless of the application where it is used, the final goals of simulation are the same: to save time and money in the development process, to avoid damage to property or to workers and to ensure that the system will behave as planned before it is built and put into action [11][12].

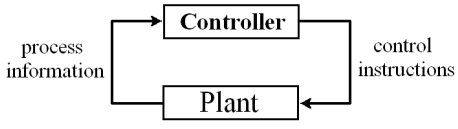

Figure 1: A basic automation system composed by a controller and a plant.

An automation system is always composed of two main entities,as seen in Figure 1. The plant (controlled process) and the controller (this controls the plant) are connected so process related information is sent from the plant towards the controller and control instructions are sent from the controller towards the plant.

In the case of SIL simulation, both the plant and the controller are converted to simulation models, linked and then simulated in a closed loop. This approach can be seen in [11] where aircraft control systems were designed and simulated. First aircraft control surfaces such as flaps, aileron, rudder and elevator were constructed using CATIA. Next, a dynamic model was designed and simulated using Dymola. An aerodynamic model that can be updated based on the aircraft model was then created in Tornado. Catia , Dymola and Tormado applications exchange information so different concept models of control surfaces are tested for different aircraft shapes and layouts.

In the case of HIL simulation there is an integration of real components and systems inside a common simulation environment. In the closed loop simulation, some components are real while others emulated on the computer, running as models. It should be obvious that HIL simulations can be obtained from SIL simulations.

Hardware in the loop simulations are adequate for closed-loop testing of logic controllers for manufacturing systems, as seen in [12]. Another possible application is proposed in [9] where the HIL simulation was realized using a personal computer with Dymola for component modeling and simulation, connected to a real energy storage test bench, with the purpose of testing energy storage systems.

To make the work associated with HIL simulation faster and easier, in [13] there is an example where automatic generation of microcontroller code is done.

SIL and HIL simulations are both very important methods in the development of automated systems controllers. They come as solutions to the problem of ever increasing complexity of automation systems, replacing old manual test methods that are no longer efficient. Both SIL and HIL allow function tests of the system or system components to be carried out under simulated (real) conditions. This is safer for the equipment used and for those involved and is also faster and less expensive that building and running tests on real prototypes (the old method). Another advantage is that both these methods are able to simulate closed dynamic loops (like fuel mixture in engines) while traditional manual tests can not.

The HIL approach has many advantages when compared with SIL because the simulation is performed with the real program running in the real controller. This means that additional work, time and money expenses are not needed to develop the simulation model of the controller and also using the real controller can be very important, specially, when different timescales can be considered from a real point of view, mainly in what concerns the real evolution of the controller and its timescale, running in real conditions of work.

Modelica and Dymola

Modelica [14] is an object-oriented, multi-domain, declarative modeling language used in component oriented modeling of complex systems (systems that contain mechanic, electric, hydraulic, thermal, control or electric power subcomponents).

It was designed to be domain neutral so it can be, and it is widely used in applications such as fluid systems (hydraulics or steam power generation), automotive (power-train) or mechanical systems (mechatronics or multi-body systems). Still, Modelica is just a language and in order to do something useful with it, there is the need to use a software tool that implements it. One of the most popular commercial implementations of Modelica is Dymola, from the Swedish company Dynasim AB. Modelica is a non-proprietary modeling language developed by the non-profit Modelica Association (which also develops the free Modelica Standard Library containing about 920 generic model components and 620 functions in different domains).

Dymola [15] stands for Dynamic Modeling Laboratory and is a commercial modeling and simulation software tool based on the open Modelica object oriented modeling language. It is developed by the Swedish company that belongs to Dassault Systems AB group.

It is available as standalone product or as integrated part of CATIA Systems V6. Dymola has unique multi-domain engineering capabilities. Models can consist of reusable components from many engineering domains. These components are stored in domain specific Libraries. The available libraries cover many engineering domains such as control, hydraulics, pneumatics, mechanics, thermal, and others. The user is free to build his personal library with developed models.

Dymola also gives the user access to the Modelica code used to define the models. The Modelica Standard Library contains many different components and the equations for every single model can be seen. This

helps the user better understand what the model does and how it does it. If the model does not exactly suit his needs, the developer can duplicate that model and modify it.

When finished, models in Dymola are transformed from Modelica to simulation code using Symbolic Manipulation and this process optimizes the simulation code.

The communication between HIL system components inside a PC (the IPC)

When building a HIL simulation system, special attention must be paid to the communication between the plant model (in this case developed using Dymola) and the real controller (usually a PLC).

The simulation model provides in real-time all the process signals needed by the PLC to exercise its control function. These signals usually pass a D/A converter before being sent to the PLC as voltages. In reply, the PLC sends control signals that pass through an A/D converter before reaching the plant model.

Some ways to link the plant model and PLC will be next presented.

- **Dynamic Data Exchange (DDE)**

DDE [16] is a generic protocol that allows any application to monitor changing data provided by any other application. It allows the exchange of information between applications that run on a Windows PC and was first introduced on Windows 2.0 in 1987.

DDE makes it possible to connect a Graphical User Interface (GUI) to a device driver in a custom made system control application. The device driver that connects to the real PLC or other process equipment will run as a stand-alone application acting as DDE server. The GUI software acts as DDE client application, exchanging data with the DDE server. Usually, it is the DDE client that initiates the communication.

The communication between the DDE client and the DDE server has to pass through a slow software message routing mechanism in Windows, reducing the efficiency of the system.

Figure 2 below shows the basic setup of a HIL simulation system that uses DDE standard for inter-process communication. Inside the Windows PC a plant simulation model acts as the DDE client while the DAQ software acts as DDE server. This setup uses a DAQ card to route signals between the PLC and personal computer. The physical connection options between the PC, DAQ card and PLC depend on the models and capabilities of these three devices and will not be discussed at this paper.

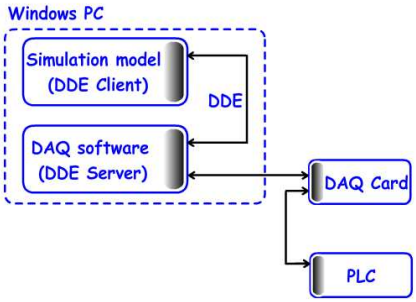

Figure 2: Outline of a HIL system that uses DDE communication.

- **OPC (OLE for Proces Control)**

OPC [17] is a standard similar to OLE (Object Linking and Embedding) but aimed at the automation industry as OLE did not meet all the unique communication needs of the industry. Data is exchanged between programs in a similar way to OLE but as a plus, one can identify if the date is accurate and also check the time it was received. Specific needs of the automation industry such as real time communication (through data access), alarming, historical data access and treading are all considered in the specification of OPC.

OPC is implemented using a client/server approach. The program that requests information is client and the program that provides the information to the client is server. The same program can act as client to some servers and as server for other client programs.

In the case of an application, that contains a Windows PC linked to a PLC, the OPC server is a software program that converts the hardware communication protocol used by the PLC, into the OPC protocol. Such programs are specific to the PLC type and some are allready available for free download on the internet. The OPC client software is any program (ex: a Dymola plant model) that needs to connect to the hardware. In a HIL setup that uses the PC to run a plant model linked to a PLC (that runs the control software), the OPC client (sim. model) uses the OPC server to get the control commands from, and send process parameters to the PLC.

Being an open standard, communication based on OPC involves lower costs and much more configuration options for the designer. It is enough for a hardware manufacturer to provide a single OPC server for its device, and that device can instantly be linked to any OPC client. On the other hand, software developers only have to include OPC client capabilities in their products and those products become compatible with thousands of hardware devices, thus results the multitude of configurations mentioned above.

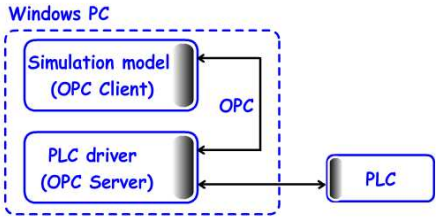

Figure 3: Outline of a HIL system that uses OPC communication.

Figure 3 shows a basic setup of a HIL simulation system that uses OPC standard for inter-process communication. Inside the Windows PC a plant simulation model acts as the OPC client while the PLC driver acts as OPC server. This setup does not use a DAQ card to route signals, instead it uses the PLC analog and digital converters (if needed) and an Ethernet connection (build into the PLC or PLC expansion module).

The typical OPC connection scenario is a single client-server connection made on a single computer, as showed. Depending on user needs, other possible connection scenarios such as OPC aggregation / tunneling or bridging are possible.

- **DDE and OPC communication interface**

A common practice seen in industry is to use both DDE and OPC [7][8] on the same project. This approach combines the benefits of both methods but introduces the need for an extra software application (usually proprietary as Simlink in [8]) to interface the two standards.

One such example can be seen in [7]. The dynamic simulation system described there is used to train operators in running a new process before that process is actually put into service. The same simulation system is also used to validate complex control logics. It is interesting how the real control system and operator displays of the plant are included in the real-time dynamic simulator.

Simulation models are developed using Dymola and then compiled into a Windows application that is run and used as real-time DDE Server. The simulation system also includes special software that transfers the signals from the DDE Server in the real-time simulation to the OPC Server in the control system (figure 4).

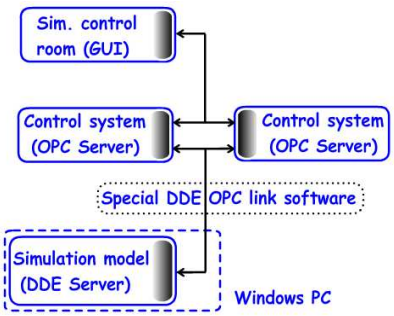

Figure 4: Outline of a HIL system that uses both DDE and OPC communication.

Because almost all modern control systems offer an OPC Server that allows opening a communication with any OPC Client, this architecture can be used without changes regardless of what control system the customer selects. Only one small configuration must be done, and that refers to a cross-reference list between signal names used in the simulator and the corresponding signal names used in the control system.

Considering the fact that OPC is becoming increasingly popular in the industry (the market already offers OPC client and server applications free of charge), it is expected that DDE servers in current applications will be changed to OPC servers. This will eliminate the need for an intermediary special software between DDE

and OPC servers resulting in a decrease in system complexity combined with an increases in both compatibility and communication performance.

- **The named pipe mechanism**

An interesting way to communicate between HIL simulation system components is to use the Named Pipe mechanism, as seen in [9]. The HIL system built at *Arsenal Research* is composed of a normal Windows PC with Dymola simulation environment, a data acquisition card and an energy storage test bench with batteries.

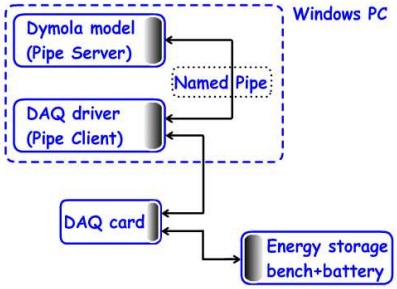

Figure 5: Outline of a HIL system that uses Named Pipes for communication.

The communication between the PC simulation tool and the real component (energy storage test bench) is done through a commercial data acquisition card from National Instruments (using the DAQ functions from NI and the analog inputs and outputs).

Inside the PC there are two important processes that run (the Dymola application and the process that provides the input and output functionalities of the interface card). These two processes communicate using the IPC (inter-process communication) mechanism of "Named pipes", provided by the operating system.

Named pipes are designed to allow communication between a pipe server and one or more pipe clients. Behind the communication is a first-in-first-out behavior that can be accessed like a file. This type of behavior also provides an implicit synchronization between the two processes.

In the example under discussion, the Dymola process is the pipe server; it generates the pipe and then waits for connections with pipe clients, like the I/O process. After executing the simulation steps, the pipe server writes data in the communication pipe for the clients to read and gets data written in the pipe by the clients.

- **The Dymola Real Time library**

The HIL simulation system used in this case consists of a Windows running PC, a PLC and a commercial DAQ card in between.

The BG_RT Library [18] seems to be a good solution to the problem of IPC inside the computer, between the simulation model and data acquisition card driver. This library includes components (like functions for reading and writing analog and digital data) that allow to access data acquisition cards directly from a Modelica-based simulation. It is design in such a way as to allow the usage of the Dymola Realtime option with data acquisition cards without any further programming effort.

The BG_RT Library is a commercial product that allows easy HIL simulation. It eliminates the need to write a custom application or function that does the communication between simulation model and other components of the HIL system.

Discussion of the proposed solution

In order to study and to propose some rules about the simulation of real-time systems considering HIL simulation, at the Automation Laboratory of the Centre for Mechanics and Materials Technologies at University of Minho, Portugal – a workbench especially devoted to this study is being developed and this chapter has, as purpose, to propose a solution for adopting the best way to connect Simulation and Real world, being Simulation done with Dymola and Modelica and the real controller an OMRON PLC.

Based on the research done until this point, the best configuration to meet the needs at UMinho is:

i. the plant simulation model built in Dymola software (version newer then 7.4)
ii. connected to an Omron PLC model CPM2A that runs the control software
iii. the physical connection between components done using a serial port
iv. and the communication between processes inside the PC done using OPC

The figure bellow shows the basic outline of this system configuration:

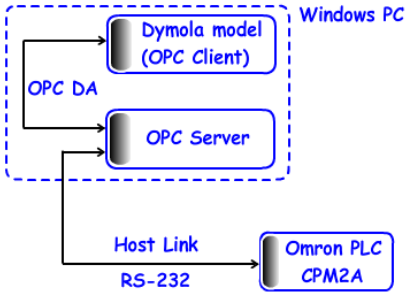

Figure 6: Setup of the a HIL simulation system that uses OPC and serial communication

- **The plant simulation model**

Dymola will be used to build the plant simulation model [15]. For better performances and to ensure maximum connectivity to other applications, it is better for the simulation model to a have OPC interface, behaving as client in an OPC connection with a serve (as will be presented later).

At this moment, the version 6.0 of Dymola available at the University of Minho does not have such an interface but it seems that the next release will benefit from such an improvement. (a version newer than 7.4)

As alternative, it would be possible to use version 6.0 of Dymola to build the plant model then generate the Dymosim executable file, running as a Windows application used to initiate a DDE communication.

What is Dymosim? The term stands for DYnamic MOdel SIMulator and is an executable generated by Dymola in order to simulate a model, and then used to perform simulations and initial value compilations. Dymosim contains the code necessary for continuous simulation and event handling.

Dymosim is essentially a stand-alone program without any graphical user interface. It reads the experiment description from an input file, performs one simulation run, stores the results in an output file and terminates. This sequence of actions can be called either from Dymola's graphical user interface or it can be called directly by the user.

In order to use DDE communication, Dymosim must be compiled as a Windows application using an additional library with Windows interface routines. Now, running Dymosim as Windows application instead of console application, Dymola can act as a real time DDE server. First, the environment variable Dymosim-realtime must be defined. Then, Dymosim will start in real-time mode. In this case the simulation time will be synchronized with the real (OS) time.

As mentioned earlier, compiled as a Windows application, Dymosim can act as a DDE server, allowing other applications to retrieve data values or to set parameters. Dymosim must be started before it can accept DDE operations. Keeping the rest of our HIL system unchanged, a way must be found to link the Dymosim DDE server to the OPC server (that connects to the PLC). The next figure shows such a way.

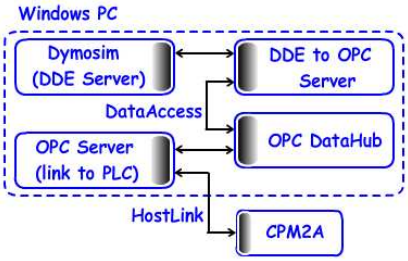

Figure 7: Possible connection between DDE and OPC servers

While Dymosim compiled as a Windows application runs as DDE server, an OPC tool can be used to convert the information between DDE and OPC specifications. Then, an OPC DataHub is placed between the (plc) OPC server and this DDE to OPC converter. The role of the DataHub is to route information between the OPC servers according to OPC Data Access specification. Last, the (plc) OPC server links according to HostLink protocol to the CPM2A controller from Omron.

This setup has been proven successful before in industrial application like the ones seen in [7] or [8] but the design can be much simplified if a new version of Dymola (with OPC interface) is used to develop the plant simulation model.

- **The controller**

The CPM2A PLC from Omron was chosen because the controllers in this series are among the most powerful small-scale control solutions in the micro PLC industry. The CPM2A has 20, 30, 40 or 60 I/O points, advanced motion control features, multiple communication ports and a powerful instruction set.

Beside the peripheral communication port, this controller features a RS-232 serial port that will be used for the connection with the Windows PC.

The serial port [19] in basically a serial communication physical interface used to transfer information in or out, one bit at a time. As they are cheap and their console functions are highly standardized and widespread, serial ports are still in use today.

The RS-232 standard defines the electrical characteristics and timing of signals, the meaning of signals and physical size and pin out of connectors. It is straight forward and a good choice for a laboratory test bench.

It would be possible to use Ethernet for the connection but that implies the use of additional components (Ethernet to RS232 converters) as the chosen controller does not have an Ethernet port. This alternative to the RS232 communication was not investigated as it is not part of the chosen system setup.

Host Link is the protocol used by Omron's CPM2A PLC for serial communication. Host Link communications enable the host computer to monitor and control the PLC's operations and to read and write the I/O memory of the device. Communications can be performed either by sending a command from the host computer and having the PLC return a response or by sending data from the PLC to the host computer.

The command or response data transferred in one exchange is known as a frame and one frame can contain up to 131 characters of data. The PLC automatically returns an ASCII code response when it receives an ASCII code command from the host computer. The host computer must have a program that controls the transition and reception of the commands and responses.

Concerning data transfer accuracy, both vertical and horizontal party checks are performed on communication data so that error-free communication is achieved. All effects of communication problems are eliminated by combining error checking and retry processing.

Before the use of OPC, developers always had to build custom drivers to communicate between different applications and serial connected devices. Today using an OPC server for interfacing the serial device (such as PLC), the developer must not worry about the communication standard or protocol as the OPC server will do the necessary conversions.

- **OPC communication**

There are many advantages to choosing OPC communication *[17]* as there is no need to buy or develop complex and expensive custom application programs for IPC. Today there are numerous OPC servers / clients / tool available for free download on the internet. The problem is to find the right combination of these tools needed to solve the task.

As Dymola already experimented with OPC interfaces, it is expected that in near future there will be a version of Dymola released with OPC interface capabilities. Using that version of Dymola, the plant simulation model will be developed. This model will act as client, providing information to the server and reading information, according to the OPC specifications. At this moment, it is possible to have the plant model running as DDE server and use specialized software tools to convert data from DDE to OPC (as in section 5.1).

This is a software application that acts as an API (Application Programming Interface) or protocol converter. An OPC Server will connect to a device such as a PLC or to a data source such as a database or user interface, and translate the data into a standard-based OPC format. OPC compliant applications such as HMI (Human Machine Interface), historian, spreadsheet, trending application, etc. can connect to the OPC Server and use it to read and write device data.

The exchange of process control data is specified by the OPC standard but there are many OPC specifications that serve different communication types and purposes.

The OPC Data Access is a group of standards that provides specifications for communicating real-time data from data acquisition or control devices such as PLCs to display and interface devices like Human-Machine Interfaces (HMI). The specifications focus on the continuous communication of data.

The OPC Data Access specification is also known as OPC DA and will be used as communication standard in the chosen HIL system. There are three attributes associated with OPC DA. These are (1) the value,

(2) the quality of the value, and (3) a timestamp. The OPC DA specification states that these three attributes have to be returned to an OPC client making a request. Therefore, if the data source is not capable of providing a timestamp, for example, the OPC DA server must create a timestamp.

A dedicated tool (available online) can be used to configure the link between the OPC server and client, ensuring the OPC Data Access. According to the application, other OPC specification may be of interest: the OPC Historical Data Access, OPC Data Exchange, OPC Alarms and Events or the OPC Unified Architecture.

As these specifications are constantly being update by the OPC Foundations, the user should be aware of what specification version it uses or needs to use.

Conclusion and future work

Because of economic and safety reasons, HIL simulation systems are widely used in the industry. How to build such a system is a very interesting topic to study. Even though HIL systems have the same basic configuration (a software components, a real component and a communication link between these two), these systems can be very different as both the software components and the communication link could be separately developed and propertary solutions.

The introduction of OPC simplified and unified these solutions but still, the interaction between the real controller and the plant model (that runs in a PC) is a complex and hard task to perform when the real-time simulation and real-time behavior are critical for guaranteeing the safe bahavior of the system.

It was proposed a simple HIL simulation system setup along with a solution to the IPC issue. This is the first step of an ongoing work to develop and build a HIL workbench that can be used to study the simulation of real-time systems at the Automation Laboratory of the Center of Mechanics and Materials Technologies of the University of Minho, Portugal.

Bibliography

[1] Machado, J. et al. " Safe controllers design for industrial automation systems. Computers & Industrial Engineering 60 (2011) 635–653.
[2] Baresi, L., Mauri, M., Monti, A., & Pezzè, M. " PLCTOOLS: Design, formal validation, and code generation for programmable controllers". Simulation Software, 2000.
[3] Ferrarini, L. and Dede, A. " A Model-Based Approach for Mixed Hardware In the Loop Simulation of Manufacturing Systems". In 10th IFAC Workshop on Intelligent Manufacturing Systems (IMS), 41-46. Lisbon, Portugal, 2010.
[4] Ljungkrantz, O., Akesson, K., Fabian, M., Chengyin Y. (2010). Formal Specification and Verification of Industrial Control Logic Components. IEEE Transactions on Automation Science and Engineering. 2010, 7; 3, 538 – 548.
[5] Fritzson, Peter, Vadim E. "Modelica, a general object-oriented language for continuous and discrete event system modeling and simulation". 12th European Conference on Object-Oriented Programming (ECOOP'98). Brussels, Belgium, 1998.
[6] Dymola software. Available at: http://www.3ds.com/products/catia/portfolio/dymola (accessed on June 27th, 2011)
[7] Johan Bäckman, Mattias Edvall "Using Modelica and Control Systems for Real-time Simulations in the Pulp & Paper industry" ; Modelica Conference of 2005, March 7th-8th, 2005
[8] M. Gafvert, T.Skoglund, H. Tummescheit, J. Windahl, H. Wikander, P. Reutersward "Real-Time HWIL Simulation of Liquid Food Process Lines" – Modelica Conference of 2008, March 3rd - 4th, 2008
[9] Arno Ebner, Anton Haumer, Dragan Simic, Franz Pirker "Interacting Modelica using a Named Pipe for Hardware-in-the-loop Simulation"– Modelica Conference of 2006, September 4th-5th 2006
[10] Bausch-Gall GmbH, Muenchen, Germany (www.bausch-gall.de)
[11] Raghu Chaitanya.M.V, Mehdi Tarkian, Christopher Jouannet : „Model Based Aircraft Control System – Design and Simulation"; 27th International Congress of the Aeronautical Sciences – ICAS 2010
[12] Sebastian PreuBe, Christian Gerber and Hans-Michael Hanisch : „Hardware-In- The-Loop Simulation and Verification of Manufacturing Control Systems"; 2011
[13] Marco Bonvini, Filippo Donida, Alberto Leva : „Modelica as a design tool for hardware in the loop simulation"; Proceedings of the 7th Modelica Conference, Italy, 2009
[14] Modelica Association – "https://www.modelica.org/ " accessed in March 2011
[15] Dymola version 5.3 user's manual; copyright by Dynasim AB; www.Dynasim.com

[16] Wikipedia – "http://en.wikipedia.org/wiki/Dynamic_Data_Exchange " accessed in June 2011
[17] OpcDataHub – "http://www.opcdatahub.com/WhatIsOPC.html" accessed in June 2011
[18] Bausch-Gall GmbH, Muenchen, Germany (www.bausch-gall.de)
[19] Host Link Communication (Manual's section 4) www.omron.com