Creative Simulation Environment for Automation Control

V M Pomazan¹ and S R Sintea²
¹Ovidius University of Constanța-Romania, Mechanical Engineering Faculty, Blvd. Mamaia, No. 124, 900524, Constanța, Romania
²Constanta Maritime University, Naval Electromechanical Faculty, Electrical Engineering Sciences Department, 104 Mircea cel Batran Street, 900663 Constanta, Romania

E-mail: vpomazan@univ-ovidius.ro

Abstract. Modern automation control design requires new technologies to develop, test and transfer technology from design to real world. One important step in automation design is testing automation algorithms in a simulated environment appropriate by real world. Our scope is to help designers to create a simulated environment for testing automation control. We created an interface and programming tool which help designers to simulate industrial installation and can connect with automation control equipment, using real electrical signals. Our base equipment is an industrial input/output controller, which connects with the computer. On the computer, is defined and runs a simulation environment, using a software tool. This environment communicates with the real world via an input/output controller. Design engineers can create their own simulation environment, using high level programming language (C++.Net, VB.Net or C#.Net) and can connect their software with external controller, using the proposed tool or a set of objects, from the library that might be added to it. This tool is useful for the creation of a simulation environment for industrial installation or subsection of a plant.

1. Introduction

The new millennium synchronizes with the beginning of new era in industrial development. Computer monitoring and control in industrial installation is the remarkable steppingstone of the new industrial paradigm. The development of automation control theory brought new methods of control optimisation into the new control installations. Applied software engineering, with applications and programming concepts as Object Orientate Analysis, Object Orientate Programming and Actor-Model-View Programming, is paramount for nowadays industrial control.

In this context, the automation control becomes more structured, more organized and more flexible. Using the communication news achievements (4G+, 5G, IoT and Industry IV), automation control systems will grow in the next years. They will become more flexible, more cooperative and more scalable.

Automation designers and engineers will need to be ready to cooperate with customer automation systems and automation control. They need a tool to communicate with customer automation system (this tool can use IoT and Industry IV technology). And, also, the engineers must have a simulation tool which can create and simulate the customer environment, in the scope of automation and control testing. This work refers to such a tool, designed for a generic automation and control.
Another important area, where one can use this tool, is in education and professional schools. This simulation tool can behave as a flexible and easy to use equipment for simulations of various installations and control systems, aiming to train students or professional technicians. Using this tool, one can create various simulated industrial installations, one can connect the simulated tool external connections with programmable logic controllers, to test control algorithms. The students can develop their own control algorithms, as direct applications. The professional technicians can use this tool, with programmed PLC, with the scope of understanding of automation control rules and its running mode.

The proposed tool will be able to simulate a part of industrial environment needed for remote control and automation. It will consist of two components, a hardware component and a software package. The functional industrial environment is implemented in software, using a specific description language (like BASIC language). The interface to connect with this tool is implemented in the hardware using specific input/output modules.

2. System Architecture
This tool has two layers (see figure 1):
- first is the hardware layer and consists of simulation station and simulation IO controller,
- the second is a simulator software, installed on the simulation station.

![Figure 1. Simulator architecture.](image1)

The simulation station is a PC computer running Windows Operating system. This computer is connected with SIMULATION IO CONTROLLER, to be able to communicate with programmable logic controller (PLC) from Automation Control Environment. The communication has two information flows (figure 2) [1]: Stimulus - represent information flow with signals generated by simulation environment and send to automation control environment through electrical signals generated using SIMULATION IO CONTROLLER and connected with inputs of programmable logic controller and Responses – represent information flow with signals generated by automation control environment, through programmable logic controller outputs, which are connected on SIMULATION IO CONTROLLER.

![Figure 2. Control information flow.](image2)
The Simulation Environment Tool is based on generic simulator software. This software consists of three components (see figure 3):

- SIM-KRN – is the simulator kernel module. This kernel module is the main simulator module. This module ensures communication between simulator modules; manage simulator modules activities using a specific scheduler.
- SIM-BAS – is a BASIC language software interpreter. This interpreter opens external files. Each file is a BASIC source, which describes the simulation algorithms. One can create more than one task which describes the simulation algorithms. This module is controlled by SIM-KRN and it is instantiated for each running task. Each task has synchronous execution.
- SIM-GDA – is a graphical device application, which communicates with Windows GDI. This module is configured by external special configuration file. This file configures SIM-GDA module to create a dynamic graphical application page. This page is shown on simulator console. The dynamic elements of the page are controlled by tasks, executed by SIM-BAS module.

3. Hardware Simulation Environment

Hardware Simulation Environment is consists of two components:

- Simulation Station
- Simulation IO Controller

3.1. Simulation Station

Simulation Station is a computer running Microsoft Windows 8.1/Microsoft Windows 10 with:

- Graphical interface
- User console (keyboard and mouse)
- one or more USB ports,
which is running Simulation Software Environment.

Using this computer, the user can control Simulation Software and create simulation scenarios and tasks.
3.2. Simulation IO Controller
Simulation IO Controller is an embedded controller consists from one or more modules:

- first base module is SIM-CPU-IO (figure 4). This module can communicate with Simulation Station over USB bus. It can generate signals to programmable logic controller or can read signals generated by programmable logic controller, in according with simulation algorithm described in simulation software. Also, the SIM-CPU-IO has I2C bus connector, which gives the possibility to connect with extension modules. SIM-CPU-IO module is a hardware interfaces based on STM32 CPU [2].

![Figure 4. SIM-CPU-IO.](image)

- another important module is SIM-EXT-IO (figure 5). This module connects with SIM-CPU-IO over I2C bus connector and it extends the number of inputs/outputs signals of base SIM-CPU-IO module.

![Figure 5. SIM-EXT-IO.](image)

To connect with SIM-CPU-IO connection we are using I2C Slave connector, from SIM-EXT-IO, and we introduce an I2C Master Connector, which allows connecting a next SIM-EXT-IO module. In this way, we can create a chain of modules for extension of SIM-CPU-IO module. SIM-EXT-IO modules are created based on STM32 CPU.

4. Software Simulation Environment
The Software Simulation Environment consists of three components:

- Simulator Kernel - SIM-KRN
Simulator Basic Code Interpreter – SIM-BAS
Simulator Graphic Device Application Interface – SIM-GDA

4.1. Simulator Kernel
The Simulator Kernel SIM-KRN module is an application created using Microsoft .Net technology, which will be running on Windows operating system computer. This module is in charge with starting of simulation environment. The simulation algorithm is described in external basic language sources, saved on disk files. The simulation algorithm can be written in on basic file or in multiple basic files.

If are more than one description files, they can be executed in parallel. Each simulation file is executed in one different thread. Synchronisation, time allocation and message communication between simulation threads are ensured by SIM-KRN simulation software module.

Functional description of SIM-KRN is presented in below pseudo-code:

```csharp
SIM_APP_Structure SIM_APPs [3]; // here we store the simulator applications code
SIM_KRN_module
{
    InitialiseDrivers();
    Initialise_SIM_IO_Extention(); //initialise communication with SIM-CPU-IO
    // and SIM-EXT_IO
    LoadSimApi( SIM_APPs); // load simulator Basic Applications

    foreach ( app in SIM_APPs) {
        app.Init();  // initialize each loaded basic simulator application
    }
    stoppedEvent = false; // internal stop event of simulator
    for ( ; ; ) { // for ever loop
        foreach ( app in SIM_APPs) {
            app.Execute();  // execute a thread from loaded basic simulator application
        }
        if ( stoppedEvent) {  // if a stop event appears, we stop the main loop
            break;
        }
        foreach ( app in SIM_APPs) {
            app.TxMessageProcess();  // process any message send from an application
        }
        foreach ( app in SIM_APPs) {
            app.RxMessageProcess();  // process any message received by an application
        }
        SIM_GDA.RxMessageProcess(); // send messages to SIM-GDA module
    }
    foreach ( app in SIM_APPs) {
        app.Done();  // close each loaded basic simulator application
    }
}
```

4.2. Simulator Basic Code Interpreter
Simulator Basic Code Interpreter SIM-BAS module is an application created using Microsoft .Net. This application module interprets the code from basic file sources. The basic function of this module is to interpret the source file in commands and to generate external signals through the SIM-CPU-IO and on graphics interface (using SIM-GDA) module. The signal generated is using virtual objects [4].

Each virtual object is allocated in simulated environment. We have four types of virtual objects:
- **SIM\_DO** - digital output signal – is a binary Boolean output object, the value written in this object is found on corresponding output of SIM\-CPU\-IO (or SIM\-EXT\-IO) module;
- **SIM\_DI** - digital input signal – is a binary Boolean input object, the value read on this object is corresponding with selected input of SIM\-CPU\-IO (or SIM\-EXT\-IO) module;
- **SIM\_AO** - analogue output signal – is a binary numeric output object (with value between 0…100%), the value written in this object is found on corresponding analogue output of SIM\-CPU\-IO (or SIM\-EXT\-IO) module;
- **SIM\_AI** - analogue input signal – is a binary numeric input object, the value read on this object (in domain 0…100%) is corresponding with selected analogue input of SIM\-CPU\-IO (or SIM\-EXT\-IO) module;

**SIM-BAS** module has three basic functions:
- **Init()** is a function executed at module initialized;
- **Execute()** is a function executed each time the thread is fired to be executed;
- **Done()** is a function executed when module is closed.

### 4.3. Simulator Graphic Device Application Interface

**SIM-GDA** module is application created using Microsoft .Net technology. These module applications get simulator virtual objects status and represent them on a schematic on Windows system GDI. The graphic schematic of simulator is obtained from a special description file, stored on computer’s disk. **SIM-GDA** can display special messages and signal send by **SIM-BAS** module in Execution phase.

### 5. Conclusions

In the new industrial paradigm, advanced tools to test and easy design automation equipment in real environment are paramount. Using the proposed tool, the engineers with some basic programming knowledge will be able to create a self-simulating environment. It offers an interactive and configurable virtual environment, which facilitates the automation programming. As, in general, automation systems do not support all features required to be fully simulated and automation testing cannot be faster than the real time process, the tool proposed can sustain certain tuning operations and can be used as a plastic programming environment, to be adapted according to real-life industrial context.

Designers will be able to accelerate the time between PLC programming and testing the automation solution. They will benefit of a tool to debug their automation algorithms and correct them. In some cases, one can simulate and can correct, in real time, some simulation profiles, according to the real environment signals. This tool is useful for automation engineers and in automation laboratories, both for educational and commercial purposes.

This tool is prone to be augmented with a graphic design interface and a library with control functions, for basic interpreter. The library will relate to the graphic design interface.

### 6. References

[1] Benedettini Ornella and Tjahjono Benny 2008 Towards an improved tool to facilitate simulation modelling of complex manufacturing systems, The *Int. J. of Advanced Manufacturing Technology*. 4(1–2) 191–199

[2] STMicroelectronics 2015 - STM32F10xxx/20xxx/21xxx/L1xxxx Cortex-M3 programming manual - PM0056 Programming manual - DocID15491 Rev 5 - 2015

[3] Velazco E 1994 Simulation of manufacturing systems *Int. J. of Continuing Engineering Education and Lifelong Learning* 4(1–2) 80–92

[4] Laakso P and Paljakka M et al. 2005 Methods of simulation-assisted automation testing VTT Industrial Systems *ESPOO Tiedotteita vtt research notes* 2289 p.44