Candle Plant process automation based on ABB 800xA Distributed Control Systems

Yousef Iskandarani and Karina Nohammer
Department of Engineering
University of Agder
Jon Lilletuns vei 9, 4879 Grimstad
Norway
yousef.iskandarani@uia.no

Abstract: Distributed Control Systems setting up, programming and implementation has been reviewed in this work. The following development has been based on ABB 800xa System. Due to the fact that automation plants are getting bigger and more complex it is not possible to avoid using industrial IT tools such as those from ABB, more specific Industrial IT. To fully automate a plant with all its reliability and safety requirements the work of an engineer is needed. However, a guidelines must be developed in order to go from a concept to the completed simulator achieving optimal cost and time efficient solution. In this paper, the process of developing a simulator for a candle manufacturing process is explained from the basic concept to the simulator layout, the following development phases are implemented successfully using the ABB Industrial IT software running on a softcontroller environment achieving a clone to the real plant requirements for control, administrative access, network topology and last but not least the human machine interface.

Key–Words: 800xa system, IEC 61131-3, Industrial IT, Object Oriented Programming, Simulator, HMI

1 Introduction

The extents of automation plants grew significantly within the last decades and with it also the installation and administration of the IT infrastructure. In addition the demand on higher reliability and security requires considerably more elements as sensors, motors and PLCs to ensure a failure-free system. To be able to automate as much as possible to save money and time, it is necessary to use industrial IT. Furthermore an automated plant also gets safer by using industrial IT. As a consequence the trend goes to more automated and less manual work. To set up a fully automated plant industrial automation systems like the ABB800xa are used. Despite to the access to all the technical resources it is still a challenge for engineers to work this out. Hence, we give you a step-by-step introduction how to create a simulator out of the concept of a plant in the next sections. Starting with the idea and concept layout in section II, continuing with how to structure your concept into a workflow in section IV and recently how to implement your process in the industrial automation system of ABB in section VI. [1] [2]

2 The Candle Process

If any process is need to be controlled by a PLC (programmable logic controller), it is necessary to determine all relevant inputs and outputs as well as all relevant actors within the process. Shortly, it is required to establish a reasonable process description. The most efficient way to check out the relevant facts for the PLC is to create a sketch of the described process. As an example in figure 2, a candle manufacturing process with all its inputs and outputs as well as actors and other components is demonstrated. For a future application in the PLC, every component of the process has got an individual address. The sketch of the process is now like an instruction guide for the further steps in the process flow of the project.

3 Object Oriented Programming

3.1 Function Block Diagram (FBD)

Function Block Diagram (FBD) is a high level graphical programming language. The considered relevant function is graphically represented as a box. The box receives several input variables and provides the output variable, depending on the used function. A comprehensive basic range of function blocks and
functions are available.

The execution order of the function blocks is defined at first by the order of their creation. It is graphically represented by the order of the boxes in FBD (from the left to the right and from the top to the bottom). It is also possible to change the order by drag and drop. Some examples of possible functions inside a single function block are given in figure 2. In figure 3 an example for a simple Function Block Diagram process is shown.

3.2 Relay Ladder Logic (RLL)

A RLL program consists of a several number of rungs and connections (lines). The inputs are only binary. The logical functions AND, OR and NOT are represented by the symbols: .+!. Thus a RLL code is considered as a graphical description of a Boolean statement, as illustrated in figure 4. [2]

The output variable of the equation is represented by a circle. The inputs are represented by pairs of two parallel bars. A diagonal line between the bars indicates the variable as NOT. The AND function is constructed by placing several (minimum two) variables in series (like in an electrical circle). On the other hand the OR function is exposed by placing two or more variables in parallel.

This RLL code represents the following Boolean expression:

\[ q_1 = (u_1 \cdot \overline{u_3} + q_1) \cdot \overline{q_2} \cdot u_2 \]  

3.3 Instruction List (IL)

Instruction List (IL) or also Statement List (STL - Siemens) is a low level programming language. The variables and function call are defined by the common elements so different languages can be used in the same program. This simple example represents the following Boolean expression: \( ST2 = (ST1 \text{ OR } T4 \text{ OR } T5) \text{ AND } T1 \).

\[ \text{A(O ST1 O T4 O T5 ) AN T1 = ST2} \]

3.4 Structured Text (ST)

One of the opportunities of writing a Code in ABB is the Structured Text (ST) - Code. It is very easy to read and understand due to its logical and structured layout. It contains a broad range of possible features for assignments, expressions, conditional statements, iterations and more.
It is possible to write the code in a text editor and paste it into the ABB ST-Editor. It is the basic language to create any expression in the SFC editor. The code is written in English and for this reason easy to read and to handle.

Figure 5: Example for a Structured Text-code.

An example for an implemented ST - Code in SFC is given in figure 5. This example contains an easy if - condition which is necessary to distinguish between two cases of states (red frame). The if - condition is written like in almost any other programming language, e.g. C or Visual Basic. The ST - Code uses Parameters and Variables which have to be defined before in the parameter editor (blue frame).

3.5 Sequential Function Charts (SFC)

A Sequential Function Chart (SFC) offers efficient opportunities for translating the whole process of an application, which is for example given as a flow chart, into the ABB control system. SFC is reasonable structured into States (frame orange) and Transitions (frame green). A sequence is a closed loop. The first step is reactivated when the sequence is finished.

Another feature is to create a link from one state in the loop to another state. For example in figure 5: if the transition failure is true, then it will jump to the stop of the process immediately (frame red).

States are certain values of a process which can be reached by transitions. States also can have some sub-systems like a timer. Transitions get their information from the internal ST-Code. On the other hand parameters and Variables of objects provide the base for the transitions. Creating SFC gives a good overview about the whole process and about its behavior. A well-structured SFC makes large and complicated processes easier to read and understand.

ABB800xa Now we finished all the basic preparation and we are able to start with the real engineer work. Therefor we begin with the Plant Explorer like My ePlant from ABB where we are able to administrate all our control systems. It is the origin from where each action happens. To maintain a good overview over your control systems and their objects, My ePlant uses different structures for controls, functions and libraries.

After creating a new control network and in it a new control project it is possible to go on with the Project Explorer like ABB’s Control Builder M. Control Builder is used to create new hardware, to build up libraries and applications, to set task connections, to simulate applications and much more.

4 Single Control Modules

A special kind of control modules is the single control module which has to be created under the menu item ‘Control Modules’ in the application. It includes parameters, variables as well as other control modules and function blocks.

In our case we used one single control module to combine all the control modules we created in our own library in one program.

The parameters of the control modules need to be
connected to the variables of the single control module. You can see how to do this in figure 8.

Figure 7: Single Control Module in the application structure.

Figure 8: Connection between Control Modules.

5 ABB800XA

Now we finished all the basic preparation and we are able to start with the real engineer work. Therefore we begin with the Plant Explorer like My ePlant from ABB where we are able to administrate all our control systems. It is the origin from where each action happens. To maintain a good overview over your control systems and their objects, My ePlant uses different structures for controls, functions and libraries. After creating a new control network and in it a new control project it is possible to go on with the Project Explorer like ABBs Control Builder M. Control Builder is used to create new hardware, to build up libraries and applications, to set task connections, to simulate applications and much more.

The first step is to include all the libraries which are needed for implementing the process. ABB already provides lots of useful libraries. If specific elements are necessary, you have the option to create your own libraries as mentioned in section VII. Subsequently we continue with the applications. An application includes programs and control modules which contain the executable code (section VIII on page 4). It is important to connect to all the libraries which are used in the application. Furthermore we have to create a controller in the menu item Controllers. We can connect the application to the controller on which it should run.

6 Runtime Setup

Once we included all the required libraries it is necessary to create applications. One controller can run different applications which include programs as well as control modules. While a program is the traditional way of executing a code, it has many advantages to use control modules. They are much faster and get along with less memory. The opportunity to choose between five coding languages of the ICE 61131-3 standard is given in both cases. The possibility to create own libraries, function blocks and control modules is one of ABB’s most significant benefits. These components are the fundament for ABB’s object orientation. [4] [1] [4] [?]

Basically are control modules similar to data types, but control modules do not just include a structure of basic data types, they also contain
functionalities. In addition control modules allow splitting the code and thereby the application gets well-arranged. However, to write the application gets more time-consuming because it is much more complex, although the control modules can be reused if they are created in a library. That means, the control modules can either be scheduled in a library or in the application itself.

![Figure 10: Control module for a failure pump created in the library structure.](image)

In figure 10 a control module for a failure pump was created in the library structure. The parameters which you can see above have to be connected to the variables of the application as soon as the control module is used there. Whereas variables and function blocks work like private variables and they are only visible in the control module itself. As mentioned before the major advantage of control modules is their reusability. Due to this a lot of time can be saved, because the functionality has to be verified and tested just once.

A special kind of control modules is the single control module which has to be created under the menu item 'Control Modules' in the application. It includes parameters, variables as well as other control modules and function blocks. In our case we used one single control module to combine all the control modules we created in our own library in one program.

![Figure 11: Single Control Module in the application structure.](image)

8 Implementing HMI in ABB800xa

The HMI is another application which is located in the 'Functional Structure' of the Software 'My ePlant'. The Software where the HMI is edit called 'Visual Basic'. The design of the HMI is based on different symbols from the library. The library consists of the standard Visual Basic Library and a special library with symbols, which shows different components of a factory like motors or pipes. It is also possible to integrate own pictures. We had done this with the candle press. [3]

In the 'animation menu' you can set visual effects of a symbol for different kind of states. The connection between the program and the HMI is set by the expression builder. The variables of the created program are linked with the 'DiscreteValue' field. For running the HMI, the system has to be online.

9 Test and Verification

The most important part after we set up the system is to test and verify it. Therefore the control modules and function blocks are very helpful because they have to be tested just once which saves a lot of time.
Now it is possible to go through the program step by step and verify if everything works the right way and does not cause any problems. We are able to change the values of variables and parameters and to test and verify each Control Module separately.

The next two figures show a testing situation in our program. If the temperature sensor T1 measures a temperature with 85°C or less the process should hold on until the temperature rises over 85°C. Then the mixture pump should go on and pump the liquid from the candlemix tank to the candle press.

![Figure 12: Testing of the mixture pump 1](image1)

![Figure 13: Testing of the mixture pump 2](image2)

10 Conclusion

Process automation has been implemented successfully using Object Oriented method. The following has been implemented using the 800xA Industrial IT software from ABB. The Control software is objective oriented in a structured way, which means that changes made to an object type or instance, only affects that type or that instance only. Therefore ABB 800xa Control Software is very adequate for applications of huge industrial processes and complex control loops.

The Control Builder supports all the five IEC 61131-3 standard languages. In the control builder all relevant data types like Booleans, integers, floating point numbers, etc. are supported. Programs can be developed off line and execution simulated without having a controller connected. Projects in Control Builder contain one or several applications. Each of these applications contains a set of programs and control modules. Program code, functions, Function Blocks and Control Modules can be placed freely in any of these. The users are able to create their own data, function-block and control module types.

The possibility to create Control Modules provides a powerful concept for establishing often used functions and processes into the control loop. Control Modules can include program codes, visual representations, interaction, etc. The objects are stored in libraries and therefore ready for re-use in current and future applications.

The opportunity to create a Human Machine Interface (HMI) provides an easy visualization for every implementation. Another advantage is the simulation of the whole control system without any hardware. Thus, it is possible to guarantee a safe and reliable program for the final application.

References:

[1] Industrial it - compact control builder ae 800m - basic control software - introduction and configuration. volume 5. ABB.

