PLUG AND PLAY IN CONTROL LOOP DESIGN

Lars Pernebo and Bengt Hansson

ABB Automation Technology Products
lars.pernebo@se.abb.com bengt.g.hansson@se.abb.com

Abstract: Most modern control systems have libraries of building blocks for control loops. A high functionality and a high degree of flexibility can often be achieved. Unfortunately, it requires quite a lot of knowledge in control loop design and in the function of the participating building blocks to construct and maintain the control loops. In this paper a standardized interface for the signals between the building blocks is presented. It is shown that control loops can be configured and maintained with less detailed knowledge and a more reliable result with this interface.

Keywords: Control loop, PID-control, function block, industrial control system, design, configuration.

1. INTRODUCTION

Before the time of the computerized control systems, control loops were constructed by combining pieces of hardware. Two PI-controllers were e.g. used, together with sensors and an actuator, to build a cascade control loop.

When computerized control systems entered the scene this modular approach was retained. The reason was that it gave a high degree of flexibility. The pieces of hardware equipment were, in many cases, substituted by software function blocks. Function block diagrams were used to graphically connect the function blocks to make control loops.

It is, of course, much easier to implement complex control algorithms with computer programs than with hardware equipment. This fact, together with the rapid development of control algorithms in recent decades, has lead to a tremendous increase of possible control loop functionality. Autotuners, adaptive controllers, optimising controllers and fuzzy controllers are examples of this development.

Function blocks are still used in many systems as building blocks for control loops and function block diagrams are popular configuration tools. The function blocks often have a very high functionality and are quite complex. They also have many configuration parameters, sometimes called terminals.

There are two drawbacks of the increased complexity of the function blocks used to build control loops. The first drawback is that the user often has a lot of parameters to connect and therefore needs quite a detailed knowledge of the different function blocks. The second drawback is the risk of making mistakes when many parameters are to be connected.

This paper discusses a way to overcome these two drawbacks. A common interface between all function blocks, used to build control loops, is defined. The interface makes it possible to build a control loop, with full functionality, using only one connection between any two participating function blocks. In other words a “plug and play” method is used when building control loops.

It is also shown that traditional function blocks, e.g. as defined in the IEC 61131-3 standard, are not quite sufficient for implementation of the suggested interface. Some generalizations are needed and the concept of control modules is introduced as an alternative to traditional function blocks.

2. INTERACTION WITHIN THE CONTROL LOOP

A cascade control loop may be used to illustrate that quite a lot of information has to be sent between the different function blocks in order to achieve the desired functionality of the loop.

Fig. 1 shows the main signal flow through a cascade control loop. The loop works fine in normal operation, but it is not able to handle exceptional cases and is therefore not acceptable in a modern control system.

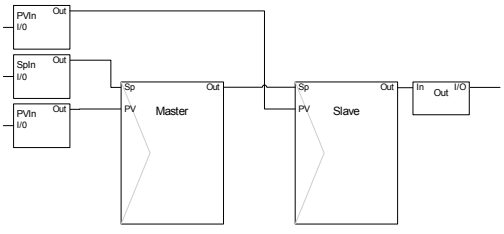

Fig. 1. The main signal flow of a cascade control loop.

3BSE028758

2.1 Integrator wind-up, also called reset wind-up.

For a single PID-controller integrator wind-up may occur when the control deviation Sp-PV, where Sp is the setpoint and PV is the process value, has the same sign for a long time.

The output of the controller is then driven to its endpoint and remains there as long as the control deviation has the same sign. The problem is that the value of the integrator of the PID-controller continues to grow even though the output has reached its limit. When the control deviation changes sign it may take a long time before the value of the integrator has decreased enough to make the output decrease. The result will be bad control loop behaviour with e.g. large overshoots.

In a cascade control loop integrator wind-up may, of course, occur in the slave controller as well as in the master controller. But even if it is prevented in both controllers integrator wind-up may occur in the cascade loop. The reason is the interaction between the two PID-controllers.

When the output of the slave controller has reached e.g. its upper limit the output of the master controller may still continue to grow, because it has not yet reached its limit. When the control deviation of the master controller then changes sign it may take some time before the output of the master has decreased enough to cause the output of the slave to decrease.

To be able to prevent this type of wind-up the master controller has to receive additional information from the slave. The output OutMaxReached in Fig. 2 is set when the output of the slave has reached its upper limit. The input InhibitInc will then prevent the output of the master to increase further. OutMinReached and InhibitDec work analogously.

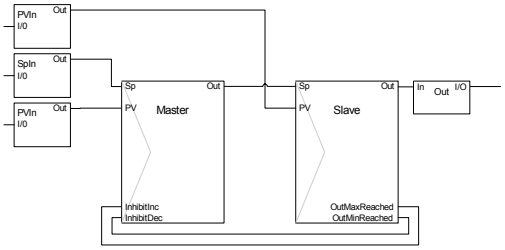

Fig. 2. A cascade loop with prevention of integrator wind-up.

2.2 Bumpless transfer.

A PID-controller may work in different modes, e.g. manual, automatic or tracking. The controller has to "behave bumplessly" when the mode changes. This is usually interpreted as a requirement that the controller output shall be continuous.

Let us take the transfer from manual to automatic mode as an example. In a single PID-controller bumpless transfer is achieved by adjusting the integrator so that the output becomes continuous. In a P-controller this is not possible.

A cascade control loop is often configured with the slave as a P-controller and the master as a PI-controller. Even though the slave is not by itself able to achieve bumpless transfer from manual to automatic mode, the cascade loop is. The integrator of the master may be adjusted so that its output gives a control deviation, Sp-PV, for the slave, such that the slave output becomes continuous.

To be able to adjust its integrator correctly the master controller has to receive additional information from the slave. It has to know when to adjust the value of its integrator and the required value of its output.

The cascade loop in Fig. 2 has been extended with two additional signals in Fig. 3. The Backtracking signal is set when the slave is in manual mode. BacktrackingValue gives the desired output value for the master in order to achieve bumpless transfer to automatic mode in the slave.

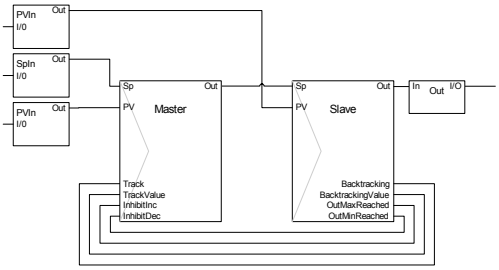

Fig. 3. A cascade control loop with backtracking.

If the slave is a PI-controller it is able to achieve bumpless transfer, in the sense of a continuous output, by itself. But the behaviour of the cascade loop can be approved if also the master participates. In this case, BacktrackingValue from the slave is equal to the process value of the slave. The effect is that not just the output of the slave becomes continuous, but also its derivative.

2.3 Signal quality.

The process value from the I/O-system often contains some quality information. It may be anything from hardware malfunction to overflow and underflow in the converters.

The quality information may be used to influence the behaviour of the loop. If the quality of the process value of the master controller is bad the slave controller may be required to go to some kind of safe state. A quality signal has been added in Fig. 4.

In this case the quality information may be altered by the master controller. If the quality of the process value of the master is bad and the controller is in automatic mode, the output of the controller will also have bad quality. But if the master is in manual mode

the value of the output is entered by the operator and the quality of the manual value is considered good.

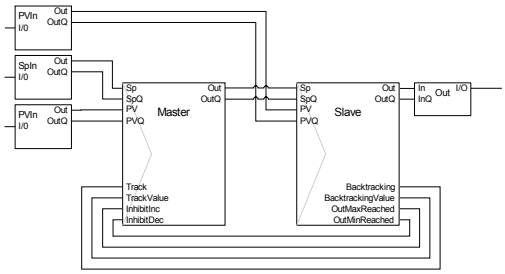

Fig. 4. A cascade loop with added quality signal.

2.4 Signal ranges.

The first computerized control systems often normalized the signals between the function blocks to the interval 0 to 100. In modern control systems the signals are usually represented in engineering units. In this paper we assume that signals are represented in engineering units.

The scaling of physical signals is done in the I/O-system. The largest and smallest values of the signal ranges are entered. Let us, for the sake of simplicity, call the largest and smallest value the range of the signal.

The signal ranges are needed in some of the function blocks of the control loop. A PID-controller needs the ranges of its inputs and output e.g. to scale its gain, which traditionally is presented as a dimensionless entity. If the PID has an autotuner, the autotuner needs to know the signal ranges in order to design its probing signals. The PID controller also has some kind of graphical interface with e.g. bar graphs and trend graphs. This interface clearly needs the signal ranges.

The range information is often entered manually in each function block that needs it. The drawback is that the same information has to be entered in many different places. This makes the configuration difficult to maintain.

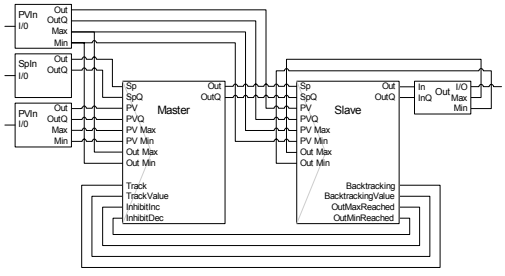

Fig. 5. The resulting cascade control loop with signal ranges added.

An alternative approach is to enter the range information only at one place, often the I/O-system, and distribute the information via signal connections. In Fig. 5 the range information, i.e. Max and Min of the signals are distributed in this way.

The resulting cascade control loop in Fig. 5 has the high functionality that is needed in most industrial applications today. It has quite a lot of signals transmitting information in the forward, as well as backward, direction of the loop.

2.5 Override control of a heat pump

The compressor of a heat pump is controlled with an override control strategy. A picture of the heat pump is shown in Fig. 6 and the control scheme in Fig. 7.

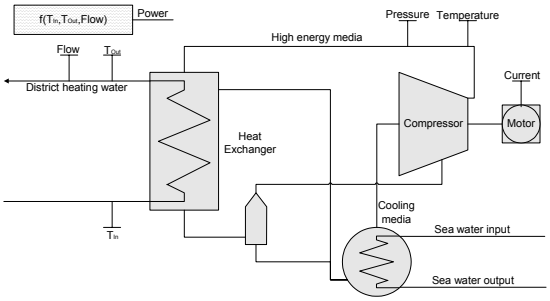

Fig. 6. The heat pump

The main control loop is a cascade loop with the power controller as a master and a positioner for the angle of the compressor blades as a slave. The angle of the compressor blades controls the pressure of the compressor.

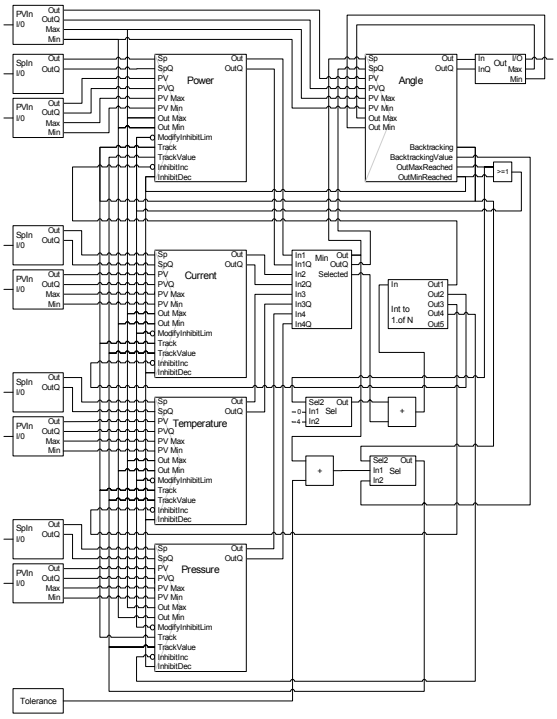

Fig. 7. Override control of a heat pump compressor.

Between the master and the slave there is a Min-selector. The Min-selector transmits the smallest of its inputs to the output. There are three other alternative master controllers connected to the Min-selector.

The alternative masters control the current of the motor of the compressor, the output temperature of the heat exchanger and pressure in the compressor. The setpoints of these controllers are upper limits of the current, temperature and pressure, respectively.

The problems with integrator wind-up, bumpless transfer, signal quality and distribution of signal ranges are basically handled in the same way as in the cascade loop. There are, however, additional problems due to the Min-selector. Let us take the problem of integrator wind-up as an example.

When the current controller is not active, its output is on its upper limit. During start up of the heat pump the current of the compressor often tends to become too high. The output of the current controller then starts to decrease and eventually becomes smaller than the output from the power controller and the current controller takes over the control of the loop.

The problem is that it takes too long time before the current controller takes over. This is another type of integrator wind-up in the current controller. To prevent this wind-up the current controller has to have information about the value of the output of the active master controller, in this case the power controller.

The problem is solved by adjustment of the integrators of the non-selected masters. They are made to follow the output of the selected master. A small tolerance is needed to take care of noise.

2.6 Industrial control loops

The cascade control loop and the override control loop are examples of control loops that often occur in industrial applications. Many other types of loops occur. A library of function blocks for construction of control loops may typically contain some 40 to 80 different block types. The following list contains some typical types.

PID-, adaptive-, fuzzy controllers
Input-, output blocks
Max-, Min-selectors
Other types of selectors
Filters, delays
Arithmetic functions
Branches, splits
Integrators, differentiators
Supervisors

In view of the two examples in this paper it is easy to imagine that a large control loop, with many function blocks, will result in a complex network of signals, passing information forwards as well as backwards in the loop.

Constructing such a control loop requires good knowledge, not only in control loop design, but also in the function of the function blocks of the control library. The risk of connecting the signals incorrectly is large and it will be difficult to maintain the loop.

3. THE CONTROL CONNECTION INTERFACE

The complexity of the loops can be reduced considerably if a standard interface is defined for the signals between the blocks. To be able to handle this interface the blocks have to be constructed according to object oriented principles.

3.1 A standard interface

A first attempt to define a standard interface to all the function blocks of the control loop can be done by studying the control loops in Fig. 5 and Fig. 7. Let us define a data structure, which we may call ControlConnection, which contains all the signals that are sent between the function blocks of the control loops.

Some of the signals are sent in the forward direction of the loop and some are sent backwards. Let us collect all the signals sent forwards in a substructure called Forward and the signals sent backwards in a substructure called Backward.

The ControlConnection structure may then look as follows.

```
ControlConnection.Forward.Value
                         .Status
                         .Range.Max
                               .Min
ControlConnection.Backward.Value
                          .Backtracking
                          .MaxReached
                          .MinReached
                          .Range.Max
                                .Min
```

The Value component in the Forward structure represents the main signal flow of the loop. The Status component contains information about the quality of the loop.

When the Backtracking component of the Backward structure is set, the output of the preceding block is requested to follow the value of the Value component of the backward structure.

When the MaxReached component of the Backward structure is set the output of the preceding block is requested to treat the value of the Value component of the backward structure as its upper limit. The component MinReached is handled analogously.

The Range has to be included in the Forward, as well as in the Backward, structure of ControlConnection. The reason is that the Range sometimes has to be transmitted in the forward direction of the loop and sometimes in the backward direction.

The data structure ControlConnection is deduced from the two examples in Fig. 5 and Fig. 7, but it turns out that it contains all essential information needed in the general case.

If the blocks of the cascade loop in Fig. 5 are connected with graphical connections of ControlConnection type the loop appears as in Fig. 8. Note that the loop in Fig. 8 looks graphically like the loop in Fig. 1, but it has the complete functionality of the loop in Fig. 5.

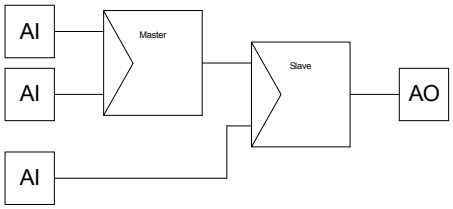

Fig. 8. The resulting cascade control loop but the blocks are connected with ControlConnection structures.

3.2 The blocks of the control loop as objects

In the previous section we saw that a control loop is configured in a much simpler way if the blocks are connected with the ControlConnection structure. To be able to maintain full flexibility in building control loops all the blocks of the library have to use the ControlConnection interface.

Thus, all blocks have to handle all the components of ControlConnection in such a way that every block behaves as anticipated in all conceivable configurations.

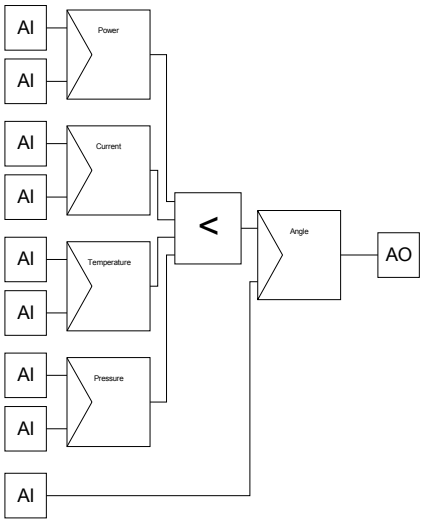

Fig. 9. The override control loop when the blocks are connected with ControlConnection structures.

Let us consider the Min-selector of an override control loop as an example. The override control loop of Fig. 7 will appear as in Fig. 9 when ControlConnection structures are used.

All the components of ControlConnection are transmitted unaffected between the selected master and the slave.

The Backward components of the non-selected inputs are computed from the Forward components of the selected input in such a way that integrator wind-up is handled, even in the noisy case. They are computed from the Backward components of the output in order to handle bumpless transfer when the slave goes from manual to automatic mode.

3.3 Ranges in the control connection

There have to be defined rules, which makes it possible for a particular block to determine if the Range in the Forward, or in the Backward, component shall be used. Let us take the cascade loop of Fig. 8 as an example.

The range information, i.e. Max and Min are entered in the I/O interface and are known to the control loop via the input and output blocks.

The Range of the process value of the master controller is transmitted in the forward direction from the input block to the master controller. The Range for the control output to the process is transmitted in the backward direction from the output block to the slave controller.

The Range of the process value of the slave controller is transmitted in the forward direction from the input block to the slave controller. This range is transmitted further in the backward direction from the setpoint of the slave controller to the output of the master controller.

In this example of a cascade loop, as well as for the override loop, it is clear how the range information shall be transmitted to all points of the loop. In the general case there may be parts of the loop where the Ranges are undetermined or overdetermined. Rules have to be specified how to take care of these cases.

4. REQUIREMENTS ON THE CONTROL SYSTEM

In the previous section it was shown that a data structure, called ControlConnection, can be defined as the only signal interface between the blocks in a control loop library. It was also indicated that it is possible to define a behaviour of all the blocks in a control loop library in such a way that any appropriate combination of the blocks constitute a control loop with high functionality concerning e.g. integrator wind-up or bumpless transfer at mode changes.

The implementation of such a control loop library poses, however, a number of requirements on the control system, in which it is implemented.

The first requirement is that the system allows variables of structured data types. The IEC 61131-3 standard for control system languages does specify structure data types. Therefore many modern systems allow such data types.

The second requirement is that it must be possible to send information in both directions of a structured interface. Also in this case IEC 61131-3 defines an appropriate item, namely a parameter of direction in_out.

The third requirement is that the system must have a graphical editor with graphical as well as non-graphical connections. IEC 61131-3 suggests a graphical representation of function block diagrams. This representation is, however, not good enough. Parameters with direction in_out are represented in a way, which leads to messy control loops. This is also true for most control systems on the market today.

The fourth requirement is that the system must have building blocks, which handle transmission of information more efficiently than traditional function blocks do.

In a traditional function block diagram the blocks are executed in an order determined by their positions in the diagram. It is therefore always possible to order the function blocks in such a way that the information transmitted in the forward direction of the loop will be transmitted without delay. The information transmitted in the backward direction of the loop will, however, be delayed one or several task execution scans.

In our case where all information is transmitted via the ControlConnection interface it has to be transmitted through all blocks of the loop. This means that a large control loop, implemented with a traditional function block diagram, may have severe delays in the information transmitted in the backward direction of the loop.

The building blocks must thus be able to transmit information in the forward, as well as in the backward, direction of the loop without delays.

Let us introduce the concept of *control modules* as a generalization of function blocks and *control module diagrams* as a generalization of function block diagrams. The control modules can contain two parts of code. The first part transmits information in the forward direction of the loop and the second part in the backward direction. The information is transmitted without delay in both directions.

The control modules have graphical, as well as non-graphical, connections and can be freely placed in a control module diagram. The control module diagram for the cascade loop and the override control loop are shown in Fig. 8 and Fig. 9, respectively.

5. CONCLUSIONS

Through the history of computerized control systems the requirements on control loops have increased steadily. This has lead to a high functionality of the building blocks for the control loops. It has also lead to libraries of building blocks with a high degree of flexibility. It is thus possible to build a large variety of complex control loops with a very high functionality.

Unfortunately, the tools for implementation of building blocks for control loops have not developed in a pace to match the requirements on the control loops. Traditional function blocks and function block diagrams are still used to a large extent.

It has been shown in this paper that, even though it is possible to construct control loops with high functionality using traditional function blocks, the solutions tend to be more complex than necessary. Good knowledge is required, not only in control loop design, but also in the function of the individual function blocks. The resulting control loops will be difficult to understand and hard to maintain.

In this paper the concept of control modules has been introduced. It has been shown that a library of building blocks for control loops can be built with control modules and a standardized, structured graphical interface for all signal transmissions between the blocks.

It is then possible to achieve, in addition to a high functionality and flexibility, a simplicity of configuration, which makes the control loops easy to configure and maintain. The risk of making mistakes when configuring the loops is reduced drastically, which increases the reliability of the loop.

The control module concept is implemented in ABB:s new control system Control IT as a generalization of the traditional function blocks, as they are specified in e.g. the IEC 61131-3 standard. For more information see Control IT, AC800M/C Control Functions – User´s Guide (2001).

With the ControlConnection interface and control modules we have, in fact, achieved a plug and play situation in control loop design.

6. REFERENCES

Control IT, AC800M/C Control Functions – User´s Guide (2001). ABB Automation Technology Products 3BSE 021 351 R201.