ISSN 0280-5316
ISRN LUTFD2/TFRT--5833--SE

Developing graphical user interface for a model predicative controller in an ABB software environment

Fredrik Wåhlin

Department of Automatic Control
Lund University
February 2009

Lund University
Department of Automatic Control
Box 118
SE-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
February 2009

Document Number
ISRN LUTFD2/TFRT--5833--SE

Author(s)
Fredrik Wåhlin

Supervisor
Per-Erik Modén ABB, Västerås
Tore Hägglund Automatic Control (Examinator)

Sponsoring organization

Title and subtitle
Developing graphical user interface for a model-predicative controller in an ABB software environment (Design av ett grafiskt användargränssnitt för en MPC-regulator i ABB mjukvarumiljö)

Abstract
A project at ABB CRC in Västerås aims at investigating the use of an MPC algorithm as a control module in 800xA. For users to interact with such a controller, faceplates need to be designed. This report describes the process, and results of designing such faceplates using among other the ABB Graphics Builder application. The challenge with the design work was mainly to create new and unique components and layout that seem intuitive to the user and at the same time looks and works similar to existing standards. Most of the planned faceplates were created and are working as intended, although a few functions are left for future work.

Keywords
MPC, Model Predictive Control, 800xA, control module, faceplate, design, layout, Graphics builder, Process Graphics 2, PG2, usability test.

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280-5316

ISBN

Language
English

Number of pages
60

Recipient's notes

Security classification

http://www.control.lth.se/publications/

Contents

1 Introduction

A brief introduction to the thesis work and the ABB development project that it is part of will be presented in this part. Some of the words and concepts used herein need a more thorough explanation and this will be given under section 1 Important concepts.

1.1 Background

This is the master thesis report by Fredrik Wåhlin. The thesis work took place at ABB Corporate research in Västerås during the autumn of 2008. The master thesis is part of the program Engineering Physics at the Faculty of Engineering at Lund University. This reports intended audience is the examiner at the department of automatic control in Lund and control engineers working at ABB. It will when finalized be added to the project documentation for the project that it is part of.

1.2 Introduction to the thesis project

The multinational engineering company ABB describes itself with the following statement: "As one of the world's leading engineering companies, we help our customers to use electrical power effectively and to increase industrial productivity in a sustainable way." One of ABB's main business areas is Automation technology wherein various automation systems are developed and marketed. One of those automation systems is, sometimes referred to as the flagship of ABB automation systems, the Industrial^{IT} System 800xA. It is a complete solution for process industries containing all the necessary parts from controller hardware and control algorithms all the way to user interface and alarm monitoring. A project that this master thesis is related to, aims at investigating the use of MPC, Model Predictive Control, in 800xA. The thesis work consists of designing and developing the graphical user interface for such a controller.

1.3 Problem formulation and items of concern

Predict & Control is an ABB product that has been on the market for several years. It uses MPC and is well suited for large, relatively slow, systems in the process industry for example oil refineries, paper pulp production and various chemical plants. Predict & Control is used as base for investigations of how its algorithms may fit into 800xA. The graphical user interface is to be developed in the form of faceplates for a possible control module in the new library ControlMpcLib in the Process Portal A, PPA, library structure. The newly developed Process Graphics Editor should be used and the faceplates will follow the new upcoming graphics standard of 800xA and is to be developed according to a previously made list of specifications, [2]. All of these concepts are described further below. The faceplates are to be developed from scratch, although there are two documents, [8], written at an early stage of the project that outlines the faceplate design.

The MPC controller in this project will be limited to being able to manage five signals from the physical process that will be controlled, five calculated signals from the controller to control the process with, and five assisting variables to help with the calculations. This might be a fairly low number of signals compared to many other MPC applications although compared to other existing objects in 800xA this brings about a large amount of data and will probably cause some problems concerning the faceplate design since there is limited space at hand. There are many features of an MPC controller that are not part of any existing item in 800xA, which calls for unique and specific graphical items. The challenge with this lies in the fact that, although new and unique, these items must have the same look and feel as other items in 800xA. An operator experienced in other 800xA applications must feel comfortable working with the MPC controller as well. To accomplish this, following existing design standards is important and in cases where this is not possible the design should be "as similar as possible" to existing standards. These kinds of decisions will not always be straightforward but must be thoroughly considered.

2 Important concepts

In this part, some important concepts that need to be understood to appreciate this report will be explained. The reader is assumed to have basic knowledge of control theory and programming. Some words are in *italic* the first time they are used. This means a further explanation of them can be found in the section 1 Glossary.

2.1 MPC – Model Predictive Control

2.1.1 Overview

MPC, short for Model Predictive Control, is a relatively young, advanced control theory and method that is primarily based on prediction and optimization. MPC has numerous applications especially within the process industry. MPC originated from the industry and was developed and practiced long before the academic world got its eyes on it. So, for a long time MPC was used without even having been proven stable.

One of the big advantages with MPC compared to other control theories is its ability to handle *constraints*, both in the *process variables*, that are to be controlled and in the *manipulated variables* that are the calculated output of the MPC and are used to control the plant. Many industrial processes benefit from operating close to or at their limits, on one hand due to production effectiveness and consequently profit maximization and on the other hand to manage to stay within safety or environmental restraints. MPC is well suited for multivariable systems but even smaller feedback systems can be rendered more effective with the use of MPC, especially if the application contains deadtimes. Since MPC algorithms are more computationally demanding than simpler control algorithms, such as PID control, the advantage of MPC keeps growing due to the ever-increasing computational speed of modern computers.

2.1.2 The MPC idea

The fundamental idea of predictive control is to predict what will happen in the future. This is accomplished with a mathematical *model* of the *plant* which is to be controlled. By applying the right *control signals* in the present time one can optimize the future plant behavior. In every sample the prediction is remade with current process values taken into account and a new optimal output is calculated and applied. The mathematical model of the process will always differ somewhat from the actual plant and since it is impossible to predict most disturbances the prediction will diverge from the true state. This effect is counteracted by the prediction being totally remade for every sample.

The *prediction horizon* determines how far into the future the predictions are made, see Figure 2.1 and Figure 2.2 for examples. Usually it is given in number of samples and therefore its length in actual time depends on the rapidity of the physical process. Another important term is the *control horizon*, see Figure 2.2. It denotes how far into the future the control signal at that time is taken into account in the calculations, in other words how many changes in the control signal the controller is allowed to make, which in turn determines number of degrees of freedom in the solution of the algorithm. It is only natural to imply that the control horizon is always shorter than or equal to the prediction horizon. It is not assumed that the *setpoint* is invariable; on the contrary, the setpoint is

presumed to vary throughout the prediction horizon, see Figure 2.1. By optimizing for every sample and taking into account both constraints and varying setpoints the MPC algorithm can not only make the plant optimally track the setpoint, but also keep it in its restricted working zone as well as from diverging from a given interval.

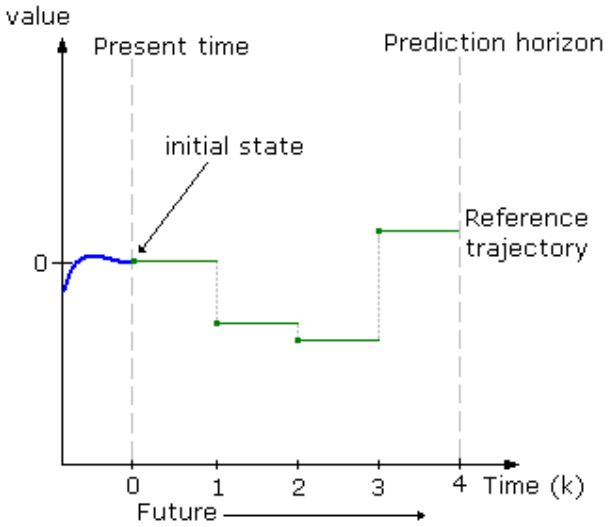

Figure 2.1. A chart of an example process. Shows the value of the process up till present time, and the reference trajectory that is defined up to the prediction horizon. It is does not have to be piecewise constant as shown in this figure but it has the same effect since sampling is assumed to occur at the start of every time step. Note that in practice the length of the prediction horizon is in general considerably longer than four samples.

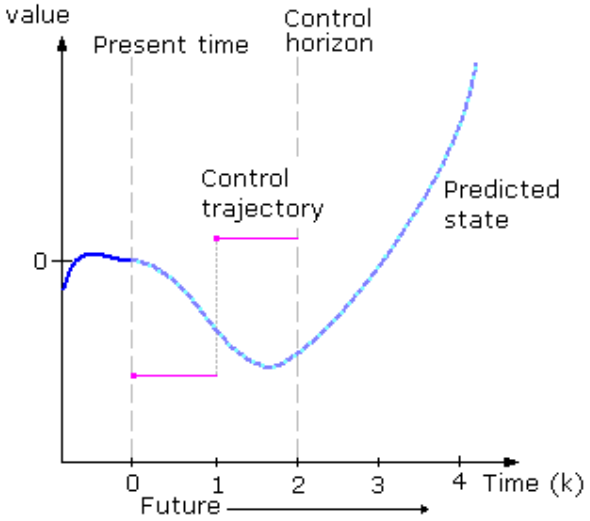

Figure 2.2. A chart showing an example of a process with the predicted value calculated up to the end of the prediction horizon and a control trajectory that ends at the control horizon. Note that in practice the control horizon is generally longer than two samples.

2.1.3 MPC theory

To better understand how MPC works, it can be helpful to start with imagining the simplest case with only one process value, one control signal and the prediction horizon just one sample long. Also assume that any time delay is much shorter than one sample. This means that the control horizon is only one sample long and the only output that needs to be calculated is the one to be used in the next sample. In this case the solution is unique and this solution will bring the plant output to the setpoint in the next sample, provided there are no disturbances and that there are no limitations on how great the control signal may be. The next step is to change the length of the prediction horizon to three samples long but keeping the control horizon at one. If not the plant dynamics and set point configuration match perfectly it will be impossible to control the plant to coincide with the set point at all three samples with just one adjustment of the manipulated variable available. This calls for an approximate solution. There are various methods and teachings to use, and an example of a solution to such a linear quadratic problem is given in section 2.1.4. Wherein, in the optimization algorithm, a *cost function* is set up. In the discrete case it has the form of a summation and its terms are quadratic. The optimal output is calculated as the state that gives the minimal value of the cost function. By using weighted variables, one can set so-called penalties or punishments to determine how important a deviation of a specific variable, at a specific time, is. However, if taking the constraints into account the problem becomes more complicated and must be solved with other methods such as quadratic programming.

Sometimes so-called *feedforward variables* are available and many MPC algorithms can use such data in the prediction and hence the optimization. A feedforward variable is neither a process variable nor a manipulated variable, but contains information about something that will affect the process momentarily or in the near future.

Just as it is natural that the process variables have constraints, for example tanks that fill or empty or temperatures that are not allowed to become too high so as not to damage the equipment, it is obvious that the manipulated variables have constraints too. Valves can only be opened fully or closed completely, pumps can only give a certain maximum pressure and a fixed volume of a solution can only contain so much of a solvent. The optimization algorithm in an MPC will also take into account these constraints and keep the manipulated variables inside their respective limits. In some cases, it is not possible to respect all the constraints on all the inputs and all the outputs to an MPC controller. If that is the case, the state that gives the minimum cost function will be used. By manipulating the weighting variables, the engineer can set up a priority list of the constraints, thus determining which constraints should be violated first should the situation arise. For example, this grants the possibility of giving priority to the maximum level of a tank over a fan speed that creates unnecessarily loud noise.

2.1.4 MPC optimization

This section will show in more detail how an MPC uses prediction and optimization to find an optimal value of the control signal.

First some definitions need to be made, Table 2.1 shows variable definitions.

Table 2.1 Variables used in MPC calculations in section 2.1.4.

Variable	Description
k	The current time step.
$r(k)$	Reference trajectory.
$u(k)$	Control signal trajectory. Exists only up to time $k-1$. (Manipulated variables)
$\hat{u}(k)$	Predicted control signal trajectory. Starts at time k.
$y(k)$	Output of the actual plant. (Process variables)
$\hat{y}(k)$	Output of the model and hence predicted output trajectory.
$x(k)$	State of the actual plant.
$\hat{x}(k)$	Predicted state. Starts at time $k+1$.
$\tilde{x}(k)$	Estimated state. Exists only at time k.
H_p	Prediction horizon.
H_u	Control horizon.

From Table 2.1 we understand that variables denoted with a hat, ^, above are predictions and are thus created by the MPC prediction algorithm, variables with no superscript are actual values that we have real knowledge about. In almost every real world scenario the state variable x is not known but has to be estimated usually by using a Kalman filter. How this is done will not be considered here and we will just let $\tilde{x}(k)$ denote the by-some-means estimated state variable at time k.

Assume that the plant is given on standard form as

$$\begin{cases} \dot{x} = Ax + Bu \\ y = Cx \end{cases} \tag{3.1}$$

and its model as

$$\begin{cases} \dot{\hat{x}} = A\hat{x} + B\hat{u} \\ \hat{y} = C\hat{x} \end{cases} \tag{3.2}$$

In discrete time the model will then be

$$\begin{cases} \hat{x}(k+1) = A\tilde{x}(k) + B\hat{u}(k) \\ \hat{y}(k) = C\tilde{x}(k) \end{cases} \tag{3.3}$$

For this example assume that $H_p = 4$ and $H_u = 2$. This is also the case in figure 1 and figure 2. This means that the control signal needs to be determined at two time steps namely $\hat{u}(k)$ and $\hat{u}(k+1)$. It shall also for convenience be assumed that the control signals previous to the present moment were zero.

$$u(k-1) = \overline{0} \tag{3.4}$$

The predicted state is calculated by iterating, starting from the initial state $\tilde{x}(k)$ and $\hat{u}(k)$.

$$\hat{x}(k+1) = A\tilde{x}(k) + B\hat{u}(k) \tag{3.5}$$

$$\begin{aligned}\hat{x}(k+2) &= A\hat{x}(k+1) + B\hat{u}(k+1) = A(A\tilde{x}(k) + B\hat{u}(k)) + B\hat{u}(k+1) = \\ &= A^2\tilde{x}(k) + AB\hat{u}(k) + B\hat{u}(k+1)\end{aligned} \tag{3.6}$$

$$\begin{aligned}\hat{x}(k+3) &= A(A^2\tilde{x}(k) + AB\hat{u}(k) + B\hat{u}(k+1)) + B\hat{u}(k+2) = \\ &= A^3\tilde{x}(k) + A^2B\hat{u}(k) + AB\hat{u}(k+1) + B\hat{u}(k+2)\end{aligned} \tag{3.7}$$

But since $H_u = 2$ and the control signal is constant thereafter

$$\hat{u}(k+2) = \hat{u}(k+3) = \hat{u}(k+1) \tag{3.8}$$

which gives

$$\hat{x}(k+3) = A^3\tilde{x}(k) + A^2B\hat{u}(k) + (A+I)B\hat{u}(k+1) \tag{3.9}$$

and

$$\begin{aligned}\hat{x}(k+4) &= A(A^3\tilde{x}(k) + A^2B\hat{u}(k) + (A+I)B\hat{u}(k+1)) + B\hat{u}(k+1) = \\ &= A^4\tilde{x}(k) + A^3B\hat{u}(k) + (A^2 + A + I)B\hat{u}(k+1)\end{aligned} \tag{3.10}$$

A commonly used cost function is

$$V = \sum_{i=1}^{H_p} \left\| \hat{y}(i) - r(i) \right\|_Q^2 \quad + \sum_{i=0}^{H_u - 1} \left\| \Delta\hat{u}(i) \right\|_R^2 \tag{3.11}$$

The notation $\|y\|_Q^2$ is called quadratic form and is a compact way of writing $y^T Q\, y$. Q and R are symmetric, positive definite matrices. They are often referred to as weighting matrices and their values can be modified to tune the optimization.

As can be seen in equation 3.11, the cost function does not include the control signal explicitly but instead the change of the control signal. A change in the control signal can be expressed as

$$\Delta\hat{u}(k) = \hat{u}(k) - u(k-1) \tag{3.12}$$

which can be rewritten as

$$\hat{u}(k) = u(k-1) + \Delta\hat{u}(k) \tag{3.13}$$

which of course apply to the next time step as well

$$\hat{u}(k+1) = \hat{u}(k) + \Delta\hat{u}(k+1) = u(k-1) + \Delta\hat{u}(k) + \Delta\hat{u}(k+1) \tag{3.14}$$

These two expressions are applied to the equations 3.5 – 3.7 and 3.10 and it is recalled that $u(k-1) = \overline{0}$. Then it is obtained that

$$\hat{x}(k+1) = A\tilde{x}(k) + B\Delta\hat{u}(k) \tag{3.15}$$

$$\hat{x}(k+2) = A^2\tilde{x}(k) + (A+I)B\Delta\hat{u}(k) + B\Delta\hat{u}(k+1)) \tag{3.16}$$

$$\hat{x}(k+3) = A^3\tilde{x}(k) + (A^2+A+I)B\Delta\hat{u}(k) + (A+I)B\Delta\hat{u}(k+1) \tag{3.17}$$

$$\hat{x}(k+4) = A^4\tilde{x}(k) + (A^3+A^2+A+I)B\Delta\hat{u}(k) + (A^2+A+I)B\Delta\hat{u}(k+1) \tag{3.18}$$

This may be written in matrix form as

$$\begin{bmatrix} \hat{x}(k+1) \\ \hat{x}(k+2) \\ \hat{x}(k+3) \\ \hat{x}(k+4) \end{bmatrix} = \begin{bmatrix} A \\ A^2 \\ A^3 \\ A^4 \end{bmatrix} \tilde{x}(k) + \begin{bmatrix} B & 0 \\ AB+B & B \\ A^2B+AB+B & AB+B \\ A^3B+A^2B+AB+B & A^2B+AB+B \end{bmatrix} \begin{bmatrix} \Delta\hat{u}(k) \\ \Delta\hat{u}(k+1) \end{bmatrix} \tag{3.19}$$

Two matrices are introduced as

$$\Phi = \begin{bmatrix} CA \\ CA^2 \\ CA^3 \\ CA^4 \end{bmatrix} \tag{3.20}$$

$$\Theta = \begin{bmatrix} CB & 0 \\ CAB+CB & CB \\ CA^2B+CAB+CB & CAB+CB \\ CA^3B+CA^2B+CAB+CB & CA^2B+CAB+CB \end{bmatrix} \tag{3.21}$$

And the estimated output is now conveniently written as

$$Y(k) = \Phi\tilde{x}(k) + \Theta\Delta U(k) \tag{3.22}$$

where the capital letters Y and U are vectors containing the predictions of their corresponding signals at all of the considered time steps.

Now, consider the cost function again. The reason that it is important to have a cost function in quadratic form is that a quadratic form only has one point where its gradient is zero. This point is an extema of the quadratic function itself and will be either a minimum or maximum. If the weighting matrices in the cost function are positive definite, as is the case in this report and which means that their elements will never become negative, the zero of the gradient is in fact a global minimum of the cost function. That is to say, by finding the zero of the gradient to the cost function, the global minimum of the cost function is found. This strong and useful result is the foundation of the optimization part of MPC. It will now be shown how this is carried out in practice. By using equation 3.22 and expressing the reference trajectory $r(k)$ as a vector $T(k)$, the cost function in equation 3.11 can be rewritten as

$$V = \left\|\Phi\tilde{x}(k) + \Theta\Delta U(k) - T(k)\right\|_Q^2 + \left\|\Delta U(k)\right\|_R^2 \tag{3.23}$$

and by introducing a help variable

$$E(k) = T(k) - \Phi\tilde{x}(k)$$

which can be interpreted as the tracking error, the difference of the reference and the free response of the system, it will be written as

$$\begin{aligned} V &= \left\| \Theta\,\Delta\hat{U}(k) - E(k) \right\|_Q^2 + \left\| \Delta U(k) \right\|_R^2 = \\ &= \left[\Delta U(k)^T \Theta^T - E(k)^T \right] Q \left[\Theta\,\Delta U(k) - E(k) \right] + \Delta U(k)^T R\,\Delta U(k) \end{aligned} \tag{3.24}$$

This is simplified to

$$V(k) = const. - \Delta U(k)^T G + \Delta U(k)^T H\,\Delta U(k) \tag{3.25}$$

where

$$G = 2\Theta^T Q E(k) \tag{3.26}$$

and

$$H = \Theta^T Q\,\Theta + R \tag{3.27}$$

The next step is to find the gradient of the cost function and set it to zero.

$$\nabla_{\Delta U(k)} V(k) = -G + 2H\Delta U(k) = 0 \tag{3.28}$$

The solution of this equation is

$$\Delta U(k)_{optimal} = \frac{1}{2} H^{-1} G = \frac{1}{2} \left[\Theta^T Q\,\Theta + R \right]^{-1} \left[2\Theta^T Q (T(k) - \Phi\tilde{x}(k)) \right] \tag{3.29}$$

We have now obtained a solution to a quite simple but rather general MPC problem. Although this is in fact a linear quadratic problem and in most MPC calculations constraints are considered. Note that in the calculations above nothing was said about the number of input and output and that theoretically there is no upper limit to how big a system that can be controlled this way. A more thorough survey of this subject can easily be found in modern literature, for example Maciejowski J. M. (2002) [4].

The conclusion of this section is that an expression have been found, containing only variables that are known at time step k, for calculating an optimal value of the control signal, and this control signal will minimize the cost function. It will also control the plant to follow optimally close to the reference trajectory excluding the effects of disturbances.

2.2 ABB Extended automation system 800xA

2.2.1 Overview

To control a modern process industry, that is to say a production or enhancement industry where a number of more or less complicated physical, mechanical or chemical processes are used, a control system is a necessity. Most industrial systems are very complex and often make use of several different raw materials or chemicals that are processed using a number of ovens, scales, tanks, mixers, conveyers and so on. These subsystems are operated through actuators, for example pumps, fans, heaters, servos, motors and valves. Data about the current state of the process is collected with different kinds of sensors. All this information needs to be collected, handled, sorted and analyzed somehow. Depending

on the result of the analysis some automatic controller action needs to be taken and finally, only information that might be important needs to be presented to the operator. This is what a control system does.

ABB Industrial^{IT} extended automation system *800xA* is a complete modern distributed control system. It includes everything needed to control a facility of almost any size or sort. It was first released in the end of 2003 and the latest version was released in November 2006. It is often referred to as the flagship of ABB automation systems. 800xA has a great deal of features and it is said to improve customer profitability and system integrity.

800xA is a huge system including numerous hardware units and a large number of programs and applications. A few of these were used in the design and development and they will be further described in this report.

2.2.2 PPA and Control modules.

One application included in 800xA that will receive additional attention is Process Portal A, or *PPA*. The PPA is an operator and maintenance work environment that is frequently used when working with 800xA. It is composed of three main views as can be seen in igure 2.3. The left view shows the current structure. A structure can be thought of as a specific part of the system viewed from a specific viewpoint. The structures contain objects that in turn contain aspects. Aspects are characteristics of an object and can be of one of many different types. The upper right panel shows the aspect of the selected object and the lower right panel is the view of the selected aspect. This show various information depending on the type of aspect currently selected. For example if a faceplate aspect is selected, it will show the looks of this faceplate, as is the case in Figure 2.3.

A *control module* is an object type of special importance. It is as its name suggests a module. It has its specific purpose and contains some programming logic that can perform a particular task and will react in a certain way depending on the context. For example a PID controller, a standard valve or pump or a signal handling object, could be, and has been, implemented as a control module in 800xA. There are more ways of building functionality into an application using 800xA than with control modules but the project that this thesis is part of makes use of a control module to implement MPC functionality.

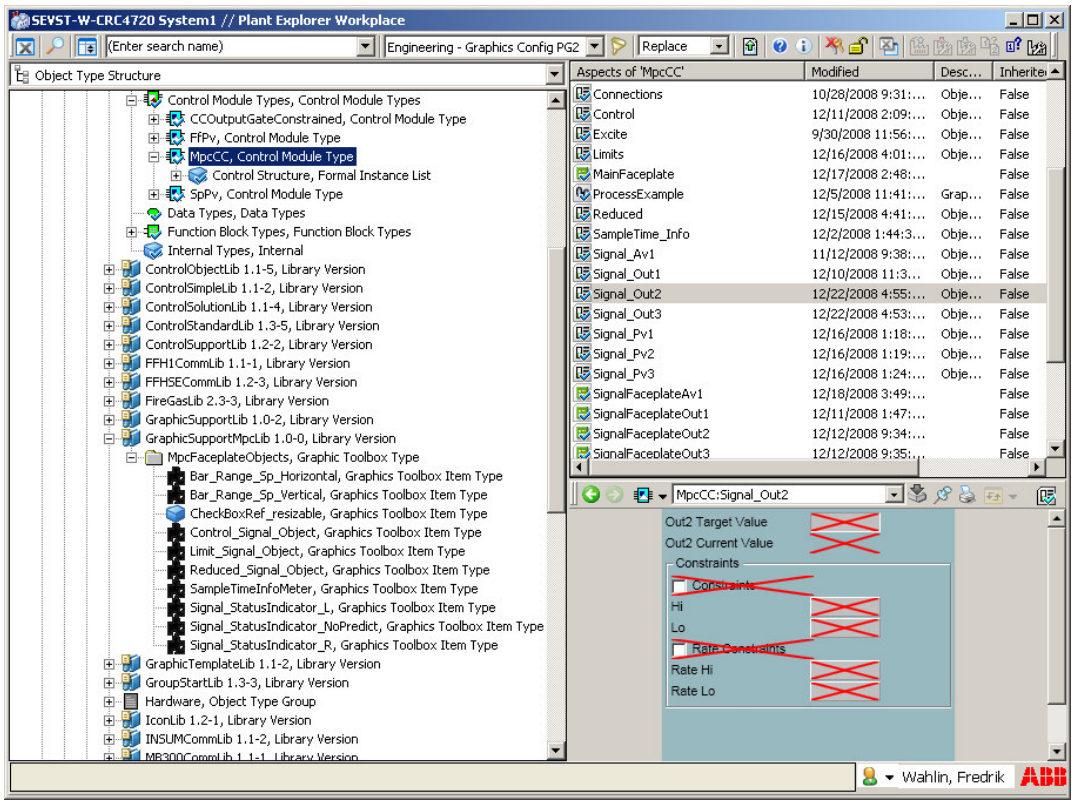

Figure 2.3 The process portal A application. A faceplate element of the "MpcCC" control module is selected.

2.2.3 Faceplates

A *faceplate* is an aspect type and is designed to be the graphical user interface for the parent object. Its purpose is to be an interaction window where the operator can easily access the appropriate information and controls. A faceplate for an object makes use of several faceplate elements, which is another aspect type. A faceplate element is in its standard form a 250 by 260 pixel surface where various controls and graphical items can be added, for example buttons, text fields, checkboxes, icons and colored shapes. The faceplate, i.e. the frame around the faceplate element, has three standard views: *reduced*, *standard* and *extended view*. In reduced view, a smaller faceplate element is shown. It is supposed to present only the most vital information about the object. The standard view shows a bit more information and usually presents some configuration possibilities. The extended view is wider and divided into two or three view panels, each containing a standard-sized faceplate element. The view panels in general have several tabs, which the user can switch between to show additional faceplate elements. The extended view is intended to present to the user all information and all configuration possibilities that might be of interest for that particular object. The faceplate itself holds buttons and *status icons* as well as the three buttons for selecting current faceplate view, see Figure 2.4. When configuring a faceplate, a built in user interface is used. With this interface, it is possible to add buttons and connect them to variables, add indicator icons, select faceplate elements to show, and configure several view options.

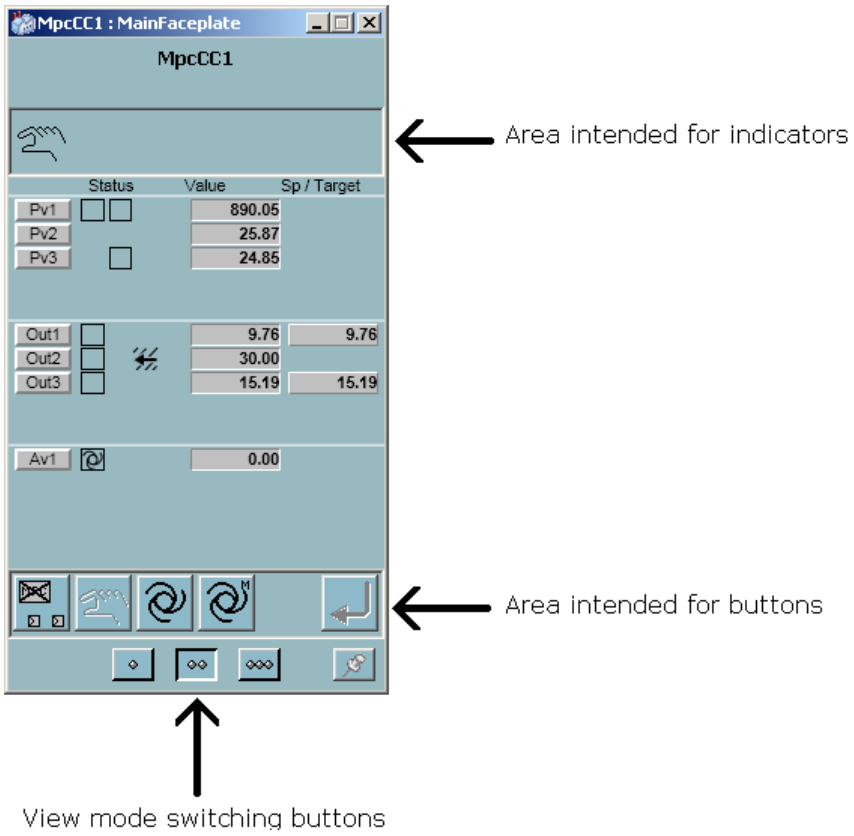

Figure 2.4 A faceplate with buttons for changing view mode at the bottom.

2.3 Process graphics editor

This section describes the tool that was used for the creation of faceplates. There are many words and phrases that have a specific meaning in this tool and the 800xA system, and these will be marked with bold font where they are defined.

2.3.1 Overview

ABB has decided to update their current process graphics and decided to develop their own process graphics editor, currently named **Graphics Builder**. The new graphics is based on *WPF*, Windows Presentation Foundation, which is also used in Microsoft's Windows Vista. It is a graphic subsystem in the .NET Framework and includes a new programming architecture and many new features including hardware acceleration, vector graphics and integrated animation system, [1].

Inside ABB the new process graphics was for a long time simply called "new graphics", but has now been named "Process Graphics 2", in short PG2. In this report the name **new graphics** will be used to refer to the process graphics created with ABB's process graphics editor, "Graphics Builder". The creating of entirely new faceplates using the new graphics routine led to a few problems due to the fact is was still under development during the time of the design work.

The Graphics builder is as stated above used to create **graphical objects**, in the editor referred to as Graphic elements or Graphic displays. The system is object oriented and any

graphical object can consist of any number of instances of other objects. In fact the Graphics builder is used to create object types that may be instantiated and used as building blocks in other object types.

2.3.2 Using the Graphics Builder

To open the editor an aspect of a supported graphics type needs to be created in PPA and the edit command in the right click menu of that object selected. The Editor will then start and the marked object will be opened. The Graphics Builder's layout is divided into several main parts. The layout is mostly window based and may be changed as preferred by the user. In the standard appearance it looks like Figure 2.5 and consist of:

- The **drawing area** is the large middle part of the view that shows the graphical object currently being edited. This is used to place, draw, mark and move objects. The result of all editing is directly shown here. There are zoom controls and grid settings to aid the user.

- The **toolbox** is the left side panel and contains several categories of objects. An object is selected and placed/drawn in the drawing area. A different tab at the bottom allows the user to access the **Element Explorer**.

- The right side of the view is divided into a lower and an upper part. The upper part contains the **Item list** that lists all instances of objects currently used in the opened graphical object. Objects can be renamed, grouped and selected from here. The list also shows all input items of the current object.

- The lower right part shows all the **properties** of the selected object. These can be edited and the result is shown in the drawing area as soon as they are applied. From here the expression editor, that provides help when writing an expression or connecting variables, can be opened. An example of a set of properties can be seen in Appendix A.

- The top part of the view provides some **shortcuts** to commonly used commands, such as save, undo and redo. Above these are the **drop-down menus** File, Edit, View, Format, Tools, Window and Help that contain commands in those categories.

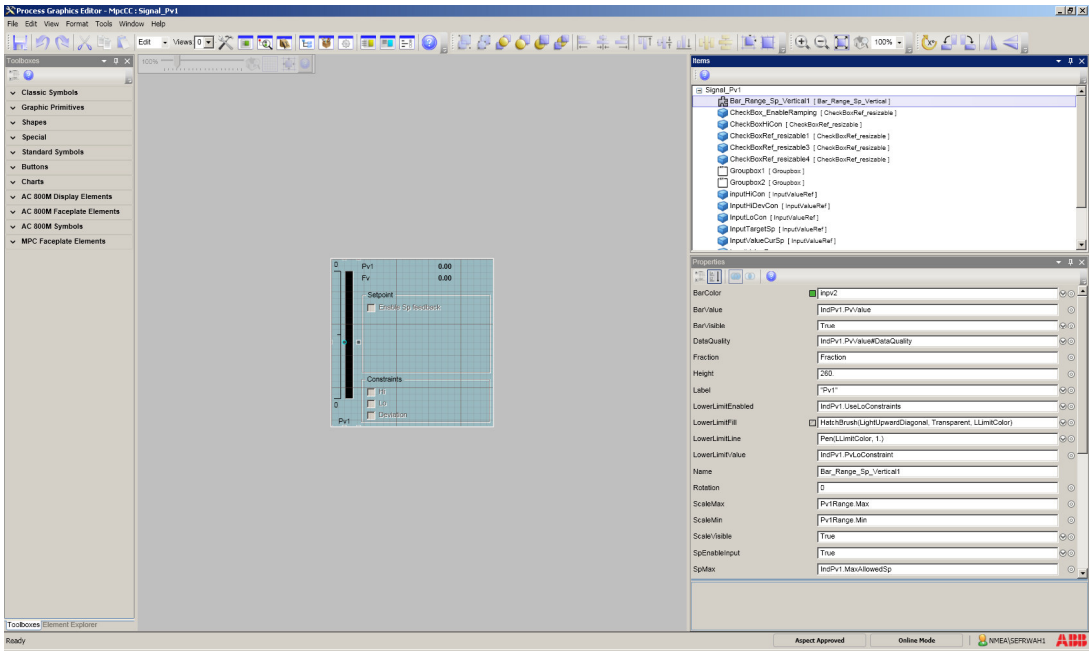

Figure 2.5 The standard view of the application Graphics Builder.

Either from the drop-down menu View or from shortcuts at the top of the window a couple of other important editor windows can be opened. These are:

- The **Element Explorer** allows the user to browse the aspect structure and select aspect objects of appropriate type to be added to the graphical object.
- **Solution Libraries** contains reusable solutions that can be used when building a large number of similar graphical objects.
- **Reference map** shows all data variables and elements used. References can be replaced and resolved from this view.
- **Resource Reference map** shows all resources used. Resources are items such as icons, logical colors and fonts.
- **Test Data** is a window where all data variables are shown. Their values may be edited temporary for testing purpose.
- **Input Properties** are defined and configured here. Input properties are variables that are used for the data that is transferred in and out of the object. If a graphical object is used as a building block in another graphic object the input properties are the ones that are shown in the properties panel and are therefore the only means of communicating with that object.
- **Expression Variables** are used when an expression is recurring in several places. An expression variable, which can only be used internally in the current object, can be defined by this expression and be used instead.
- **User types** handle different users and their permissions for the graphical object.

Input items are special properties of a graphical object that are not objects or graphic primitives themselves. Two examples of input items are Property Writer and Drag Handle. The Property writer writes a value of some variable in the graphical object to an external variable in the system. The drag handle is used to pick up mouse click-and-drag operations by the user.

Every property of an item has a **data type**. There are many different types[1] of which the most frequently used are Boolean, Real, Color, Integer, Pen, PropertyRef, Font and String. Some of these have a predefined set of selectable values as well as a means to enter a custom value. For example the type Color can be selected from either a set called **Logical colors** or it can be entered as a custom ARGB[2] value. These predefined sets of values for different data types are called **resources.**

A detailed example of an object created with Graphics Builder is given in Appendix A. Many of the notions mentioned above will be addressed and used in practice there.

[1] For a detailed description se for example Expressions for Graphics DoF [6].
[2] A color type defined by the three color channels Red, Green and Blue plus the Alpha channel that defines the transparency of the color.

3 Faceplate design

This section will treat the faceplate design work performed during the thesis work. The section is divided into parts where each part treats a subject or one type of faceplate. Each part is in turn subdivided with the headings Overview, Specifications, Considerations, Solutions and Future work. Under the headings Overview and Specifications the faceplate and the functionality it should support is described. Considerations will treat the general ideas, the problems encountered and the reasoning behind the decisions that led to the solutions. The resulting design and solutions to problems will be presented under the heading Solutions. Under the Future work heading, suggestions for future work and probable continuations will be presented and discussed.

3.1 Method

3.1.1 Overview

This section will treat the methods used during the design work and the routine of the everyday work.

3.1.2 Specifications

The project group consisted of five people: Per-Erik the supervisor for this thesis work, two other experts, a project leader and the thesis worker and author of this report. For the work of designing faceplates a PC, 800xA installation disks and hardware controller of model PM866 was provided.

Assistance to the design work was to be provided by the Library Team in the Engineering Environment department (PA/XA/ACE) from the ABB Malmö office. They have expertise both in the workings of several control modules of 800xA and in the conversion to the new graphics.

3.1.3 Considerations and solutions

The first weeks were used to get the 800xA system up and running as well as connecting the hardware controller. With the MPC testing software, provided by the project group as a control module and new graphics package by Malmö installed the design work could start. As mentioned above there existed preliminary design suggestions for some of the faceplate elements, [8], and these were used as a base. However, they were only picture mockups and they said nothing of the logic or functionality. Therefore, in a sense the faceplates were created from scratch.

Since every graphical component is connected to one, or more often several parameters, a large portion of the work effort during the early stages of the design work was to analyze and determine which parameter was to be connected to what component. As the writer of large parts of the code and a skilled expert, Per-Erik was an excellent source of information and numerous "mini-meetings" were held. With time passing, the structure of the MPC prototype control module became more familiar and the work could progress smother in this sense.

About once each month a project meeting was held. It did not affect the faceplate design directly but it sometimes inspired discussions that did.

About midway through the faceplate design work time period, the Malmö team had to down-prioritize the assisting of the MPC faceplate design and therefore lack of expertise and knowledge made the design work much harder. Some problems had to be left for future work to be solved.

At a rather late stage of the project it was decided to try out the design in a usability test. With limited time and resources at hand it was decided to do a small scale usability test with three to four participants and that these were to be picked out from the ABB office.

The usability test was expected to give answers to a few specific questions and to give an overview image of how the interface was appreciated. In addition, a user with a pair of fresh eyes can give constructive feedback about details that the developers have foreseen. The main concern about the faceplate interface at the time of the usability test was the mode switching of the MPC and its signals. The way a user perceive what mode is active was of great interest. A scenario was made up and the user was guided through it while asked to think aloud. The actual tasks that needed to be performed were never explained in detail, and the user had to make decisions based on the information given. During the scenario, questions were asked. For example: "What gave you a hint to push that button?" "How do you perceive the current status?" "What do you think would have happened if you pushed that instead?" The reactions and opinions of the users were noted and sometimes led directly to a change in the faceplate design.

3.1.4 Future work

For the continuation of work on the MPC faceplate design, it is recommended that the designer gain decent understanding of the prototype control module in order to understand the thought behind the current design and the workings of an ABB MPC. It would be a big advantage if the designer has experience in or knowledge of faceplate design in 800xA. This because many of the questions that arose during the design and development were usability related, thus a close cooperation with a usability expert would prove advantageous.

An elaborate usability test, with users having the appropriate background and knowledge and performed in an authentic environment could produce a great deal of useful feedback.

3.2 General design and the main faceplate

3.2.1 Overview

The MPC graphical user interface consists of one faceplate for each of the signals and the main faceplate. The main faceplate acts as a container for all of the faceplate elements that belong to the MPC and it can be viewed in three view modes. This part treats the general layout and the interface design not connected to any specific faceplate element. Matters such as size and view modes of the main faceplate are discussed herein.

3.2.2 Specifications

The MPC controller will have support for five process values, five out values and five assisting variables. All of these carry information that must be accessible to the operator.

The MPC-specific features such as constraints should be presented in an understandable and non confusing fashion. The faceplate should be able to be viewed in all three viewing modes: reduced, standard and extended view. The reduced view should only present the most elementary data like current state of the signals. The standard view should present a little more information, perhaps data such as process values and setpoint. Also, some information about the constraints should be presented here. In extended mode all information should be accessible. An operator could benefit from being able to see some information at the same time as some other. This should be taken into consideration when designing faceplates in the extended view mode.

The faceplate, regardless of viewing mode, should present the possibility to change the current operating mode as well as showing the current state in an intuitive manner. Standard indicators and button functions should preferably be used but when this is not achievable the solution should be similar in philosophy.

The user should be able to easily switch operating modes of the MPC and there should not be any confusion about which mode the MPC is currently operating in. The buttons and indicators were to be created using the faceplates built in interface.

All graphical objects used as components should have appropriate tooltip texts that explain their use or current state in short.

3.2.3 Considerations

The preliminary design[3] suggested that the faceplate in extended mode should be divided into two view panels each containing faceplate elements with the width of 1.5 times the width of a standard faceplate. This was mainly to be able to show the predictions in a trim curve and partly to be able to show the data from the Control faceplate and information about the constraints and forecast value at the same time. After discussion with the team in Malmö it was decided that the faceplate elements were to be of standard size and the extended faceplate was to be divided into three panels instead of two. The reason being that should the MPC faceplates be included into a ControlSollutionLib build the faceplate elements need to be of standard size. Three panels were thought to be needed to avoid too many tabs in each one and to increase the simultaneous view capabilities.

When considering what faceplates that should be able to be viewed simultaneously it was thought that the Limits and Connections faceplate elements should benefit from being viewed at the same time as the Control faceplate element. This because of the similar layout of the three, with each row representing one signal. Another trio of faceplate elements that probably would be viewed simultaneously by an operator is the Control – Pv Bar Graphs – Out Bar Graph. This was thought to be a good way to get an overview of the system operations.

A very important topic that was thoroughly investigated and discussed was the MPC mode switching. Since switching operating modes is a primary task it should be easily accessed. The buttons and indicators were created using the faceplates built in interface. The difficulty with designing the buttons and indicators was that, since there is both the MPC mode as well as the fifteen different signals individual modes, there are a large number of special situations that each needs to be considered. The general idea is that the MPC can

[3] Partly presented in the document FaceplateMockups.ppt [8], made by Per-Erik Modén.

run in off, manual or auto mode. The out signals have three closely corresponding modes as well as backtracking and tracking mode, the Pv is more complicated and treated in more detail in section Pv Signal faceplates, while the Av signals have only two. The MPC mode is affecting all of the signals but the reverse is not true. Although in many control systems as well as in 800xA there is a general safety principle regarding operating modes that states that a "lower" mode shall override a "higher". Lower in this sense mean less automated or more limited. This means that if a signal is set, for example to off mode, it does not matter that the MPC mode is auto, the signal will still not be used.

Situations where a user has set the MPC mode to manual or off, while trying to get a Pv to work in auto was considered and it was clear that the user needed information about why the Pv did not change mode, about what mode was requested and about what mode it was actually operating in. Another tricky situation is when all but one out signal is operating in manual mode and the user changes the mode of the last one to manual. The MPC is still operating in auto mode but that since the MPC has no possibility to control a Pv without any of the out signals operating in auto mode, it is in function operating as if in manual mode. How to indicate this and what should happen if one or more of the out signals were in backtracking mode as well, was discussed.

3.2.4 Solutions

A detailed explanation of how the buttons and indicators were implemented is given in Appendix B.

The faceplate will support all three of the possible view modes. The extended faceplate will have thee view panels containing faceplate elements of standard size. The extended view of the faceplate was configured so that the left tab contains the Control and Av Bar Graphs faceplate elements. The middle tab contain Pv Bar Graph, Limits and Trim Curve Pv faceplate elements while the right contain Out Bar Graph, Connections, Excite, Tune, Trim Curve Out and Sample info. The general idea is that the left tab should mainly present an overview through the Control faceplate element, the middle tab should hold information about the Pv signals and the right about the Out signals as well as some other more advanced options. This gives the possibility to simultaneously view the following trios of faceplate elements: Control – Limits – Connections, Control – Limits – Sample info, Control – Out Bar Graphs – Pv Bar Graphs, Control – Trim Curve Pv- Trim Curve Out, Av Bar Graphs – Out Bar Graphs – Pv Bar Graphs. Of course, many other combinations are available as well but these were considered to be of special interest.

3.2.5 Future work

Although the indicators and buttons were thoroughly discussed and tried out, they were not subject to any serious systematical testing. Since this is the preliminary design for a prototype, that was never the intention and therefore this work remains. In addition, alarm handling and other details such as operator note has not been considered at all. For the faceplates to follow the standard for 800xA faceplates, this needs to be addressed in the future.

3.3 Structural layout

3.3.1 Overview

This part will treat the considerations and solutions about how the faceplates and other data should be structured. Questions such as the following are treated herein: where in the system structure should the faceplate objects be places? The sub objects? What resources can and should be used?

3.3.2 Specifications

Integrating the new controller faceplates into the 800xA system implies that it should follow standards and be structured in a similar way as other controllers and similar objects already in the system. This project aims at creating a prototype and therefore the requirement are not as strict as would it be a product. The following are a few things that are not requirements but desirable. To use appropriate logical colors and logical fonts and no others. That all texts are links to a NLSID database. That all graphical objects used as sub-objects in a faceplate are placed in a standard library so that the linking does not have to be changed later on. If icons or images are used they should be of the already existing ones.

3.3.3 Considerations

Many of these matters where discussed with the Malmö team since they have the expertise in this area. The question of where in the system structure to put the faceplates and faceplate elements arose. Usually for control objects, the faceplates and faceplate elements are simply aspects the object and this was to be the case for the MPC as well. Although no other object contain multiple faceplates that link to each other in the way the MPC faceplates was planned to do. Use of logical colors and fonts was not a problem but really straightforward and the existing logical colors and fonts are sufficient to do for any of the faceplates. The NLSID is a database of short texts that are linked to in the faceplates. Some of the texts used in the MPC controller faceplates do not exist in this database, and these would have to be created. However creating these texts in the standard NLSID database would be problematic since at first they would be created in the local system where the faceplates are developed. Then this database would have to be sent to Malmö and included into their standard database. Then, a release would have to be made and installed at the local system. The big problem lies in that it is apparently not possible, or practical, to merge the databases together to just add the new items from the local system but it would have to replace the standard database and this means all other changes made to it during the development time of the MPC faceplates, would be lost. Therefore it was decided that the texts in the prototype MPC faceplates would not be links to a NLSID database but ordinary text strings.

The topic about where to place the graphical objects used as sub-objects was discussed back and forth. At first it was said that when the object were finished they were to be sent to Malmö and included into a standard library which was to be released and installed back on the local system where the development took place. This was however changed later on and due to the fact that the MPC controller is only in prototype stage and the standard libraries are used in other releases that should not include some objects not used anywhere in that release. It was instead decided that a new library containing the new objects was to be created on the local system for possible merging with the existing standard library.

Many of the icons are standard icons used in many other control objects, like the "Auto" and "Manual" icons. But since there is currently no MPC in 800xA a few new icons will have to be custom made to fit the unique needs of an MPC.

Another small issue was how to store data only needed and used in the faceplates. For example, each faceplate element need to have a parameter, "fraction" that determines how many decimals all numerical values should be presented with. Also some faceplate element may need to store additional data locally such as the connections faceplate element discussed in section 3.6. A solution to this was never found and for the time being all faceplate elements that need it, have an expression variable named fraction and with an initial value of two. This is not changeable to the user through the faceplate interface and needs to be changed.

3.3.4 Solutions

All faceplates as well as faceplate elements are aspects of the MPC prototype control module named "MpcCC" and placed in a library namned ControlMpcLib. Existing logical colors and fonts were sufficient and used. The texts and labels in the MPC controller faceplates are ordinary text strings and not NLSID links. The graphical objects that are used as components in the faceplates elements were placed in a for this purpose created library on the local system. The library was called GraphicSupportMpcLib and has the same structure as the already existing standard library GraphicSupportLib. The standard icons needed were sent as .ico -files from the team in Malmö and added to the faceplates. Seven icons were custom made in Visual Studio and then added to appropriate faceplates via the "config view" interface. The icons created are shown in Figure 3.1. How they are used is thoroughly described in Appendix B.

Figure 3.1. The new icons created and used for the MPC faceplates.

3.3.5 Future work

Likely, a number of changes to the structural layout will have to be considered in the future if this prototype is going to become a finished product. Many of these may not at all be predicted by the author due to lack of knowledge about future product specifications and system planning and layout. However a few can. Certainly the issue with the NLSID needs to be resolved and the graphical components used needs to be inserted into a standard library. Probably the special components used for the MPC faceplates currently located in the GraphicSupportMpcLib library will have to be moved to a more fitting location inside a standard library. The idea of keeping all of the faceplates and faceplate elements aspects directly under the "MpcCC" control module may need to be reconsidered although no problems due to this has been noted, except that the correct title is not shown for the faceplates and it may very well be connected with the structural layout. The issue of how to store local data such as the "fraction" parameter needs to be resolved; there is probably a standard way to do this and it might not be problematic at all.

3.4 Control faceplate element

3.4.1 Overview

The control faceplate is the only faceplate shown in standard view mode of the main faceplate. It provides the user with an overview of the system where he or she can monitor the status of the signals and change the most crucial parameters. It also serves as a platform and starting point to access any of the signal faceplates.

3.4.2 Specifications

The control faceplate should show the value of all of the fifteen signals. For the process variables, it should provide a means to view and change the setpoint and for the out signal, the target value. The faceplate element should give the user some indication of what mode the signal is operating in and a way to monitor the status of each signal. The faceplate should also provide access to the signal faceplate in an intuitive way.

3.4.3 Considerations

The main problem with designing this faceplate was space. With fifteen signals to display information about and standardized text and input field sizes, this was a challenge. The design mockups [3] gave a suggestion and it was used as a good foundation. The surface was divided into four fields from top to bottom. The uppermost field held only headings for the columns below. The other three contained the five signals of each sort. The fields were separated with a thin white line. Group boxes was discussed and tested but that left no place for the headings and they were thought to be more important than the group boxes that instead name the fields by type.

The first designs included showing the unit text for each signal, this was however discarded to allow for wider input fields making them standard size. It was thought to be of more importance to have the standard size of input fields, allowing them to show more digits, than to show the unit texts since these are mainly static and soon learned by heart by the operator and hence not very helpful. Ideally, both standard size of input fields and unit texts would be used since that is the standard in most other control objects. However as mentioned, the main problem was pixel space and therefore a discussion about which will give the most benefit to the operator took place. This is often the way the considerations about the design in this report are handled.

The indicators shown in the control faceplate element was definitely the major issues in the design of the control faceplate element. To start with, two indicators were created: Signal_StatusIndicator_L, and Signal_StatusIndicator_R, one to the left and one to the right, hence the “L” and “R”. Both were sixteen pixels high to fit one row, the left indicator as wide but the right wider. The major difficulty was to make the indicators intuitive to a user and at the same time display useful information. The information that the indicators should communicate to the user differs depending of what type of signal to which it belongs. However, there was not nearly enough pixel space to give any explanation to the user. The first idea for the left indicator was for it to be either: “inactive”, “partly active” or “fully auto”. The “fully active” and “partly active” would relate to if all or some of the control methods configured to be allowed, were actually used. That would mean that the indicators are relative and depend on the configuration, leading to situations where the indicators correctly show the same status for in reality different operation states. Instead, the general idea became to link each column of indicators with a similar, intuitively

connected meaning regardless of signal type and configuration. This is most apparent in the right indicator which was decided to give information about constrains. This seemed to be a good solution, however the need to show information about the configuration of allowed control methods for a Pv signal remained. Therefore it was suggested that the thin black line that bounds the indicator should be hidden if the connected control method was not allowed. An empty frame is easily recognized as something that is changeable and not fully active while no frame at all gives the impression that there is nothing to change or activate. This was implemented and the usability test concluded that it was straightforward to distinguish if a Pv signal was configured to allow control with setpoint and/or constraints or not.

The usability test also made it clear that a small black square filled with green color is not obviously perceived as "active" or "on". Instead, a miniature version of the standard auto symbol was used to fill the square, see Figure 3.3 or Figure 3.6.

For out signals the right indicator was also required to indicate whether the signal was backtracking or if the subsequent object had reached a limit. The solution for this was suggested in the mockups [8], but a few changes were made including indicating backtracking with a miniature version of the standard icon and showing whether the high or low limit of the subsequent object was reached. The right indicator shows information about the constraints. It was designed to be able to show: upper/lower constraint active, upper/lower constraint violated, increase/decrease constraint violated, backtracking, subsequent object upper/lower limit reached, constraints not set and any combination of these. For an out signal all of these are possible but for a Pv signal only some are, see Figure 3.2. When backtracking is active it was first decided that the indicator should not show any other status as well, for the reason that it should not be confused with the case where a subsequent object has reached a limit, in which case it is indicated if it is the upper or the lower that is reached. However, this was changed so that other indications could be shown at the same time with the motivation that it might be useful to see that constraints are violated since this can hint why the MPC reacts in a certain way and that the manual value set in the subsequent object might need to be reevaluated.

Since more than one of the indicator cases can be true at the same time, a priority sequence had to be made to determine what to indicate. The case of backtracking overrules all other and if the subsequent object is set to manual/internal the out signal is locked in backtracking mode and there is no action that can be made in the MPC user interface that will change this. After, backtracking, the indication of increase/decrease constraint is prioritized. This is because these constraints are reached only for short periods of time, when the signal makes rapid changes. As soon as the value of the signal has reached its target and stabilized the indication can change and show for example that this value is violating the upper or lower constraint. Furthermore, there are several other means by which a user can be made aware that the value is outside the constraints, but not that it is changing too rapidly. If there is no backtracking and no constraints violated the indicator will show whether the lower and/or upper constraints are active. If not active the indicator will be blank except for the frame. Although, as stated above, if constraints are not allowed to be used the frame is not visible. However this option does not exist for out signals and the constraints are always available for use, leading to the frame always being visible for the out signals. Figure 3.2 shows the indication priority of the right indicator.

Disabling a signal and thereby isolating it, not taking its value into account in the prediction calculation, is a rather drastic action and if a signal is isolated a user must be made aware of

it through the control faceplate. Therefore a third indicator was created and placed next to the right indicator. Thus making it the rightmost one, but to avoid confusion it was named Signal_StatusIndicator_NoPredict and the others were left with their names as they were.

3.4.4 Solutions

Each signal is represented by one row in the control faceplate element. An object called Control_Signal_Object was created in GraphicSupportMpcLib, and all components that were to be shown for a signal were added to this. The control faceplate element consist of fifteen instances of the Control_Signal_Object type, three separator lines and three labels at the top. The control signal object has an aspect link button, with label, to the far left which allows other faceplates to be opened. The three indicators, which in turn are objects in GraphicSupportMpcLib, are placed next to the aspect link button. Two input fields fill the rest of the space to right. The Control_Signal_Object can be seen in Figure 3.7. The Pv and out signals are connected to all components while the Av signal have no connection to the right input field and the right indicator, therefore these are hidden.

The left indicator tells whether the setpoint is configured and active for a Pv. Since there is no equivalent to a setpoint in an out signal, it tells if the out signal is in auto mode or not. The right indicator indicates constraint related issues both for the Pv and out signals. An Av signal has no constraint functionality and hence it is not shown for these.

The purpose of the NoPredict indicator is exclusively to indicate that the signal is not used in the prediction. The indicator shows the text "MPC" with a red cross over if the signal is isolated. If the signal is not isolated it shows nothing at all. The indication of isolated signals can be seen in Figure 3.5.

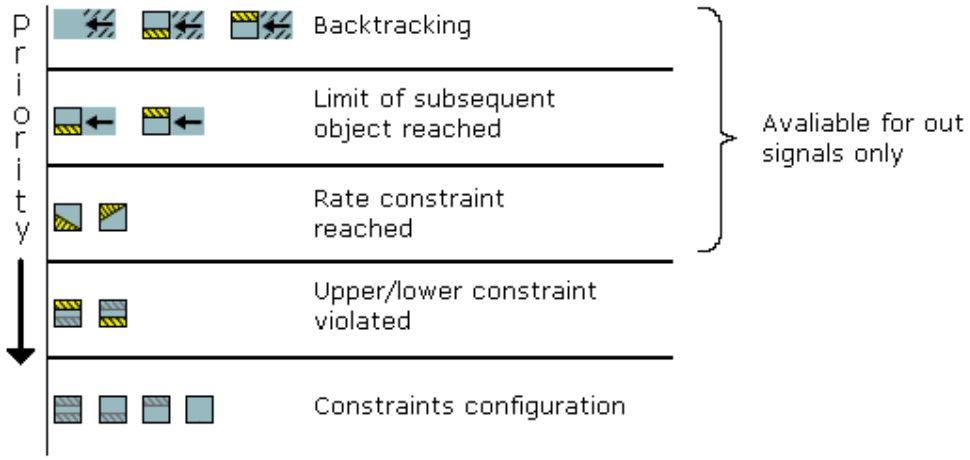

Figure 3.2 The indication priority of the right indicator.

Figure 3.3 shows the control faceplate element when all signals but one are operating in auto mode. The last signal can not change mode since it is backtracking. This is typically how the faceplate element would look after the MPC full auto button has been pushed. Figure 3.4 shows the appearance of the control faceplate element when the MPC manual button has been pushed and all signals operate in manual mode. If all signals are isolated, for example by pushing the MPC off button, the control faceplate element would look like Figure 3.5. Figure 3.6 shows a situation where various indications can be seen.

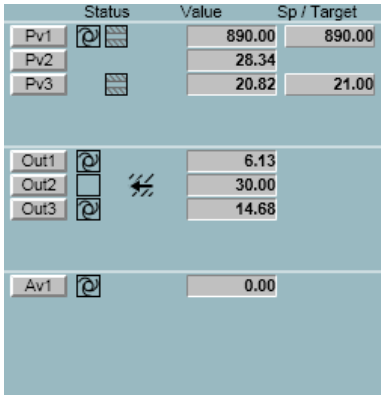

Figure 3.3. The control faceplate element. Indicators show that most signals operate in auto mode. Out 2 is backtracking.

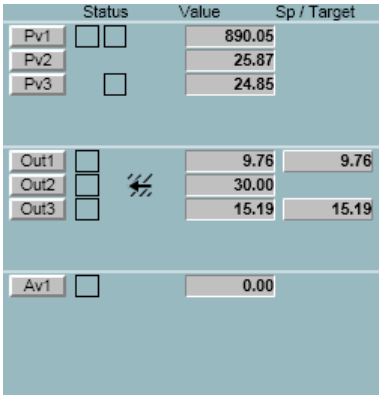

Figure 3.4. The control faceplate element. Indicators show that most signals operate in manual mode. Pv3 is configured to allow control by constraints but not by setpoint, Pv2 can use neither.

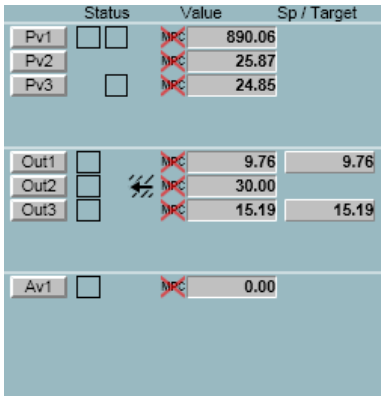

Figure 3.5. The control faceplate element. Indicators show that the is turned off and all signals are isolated.

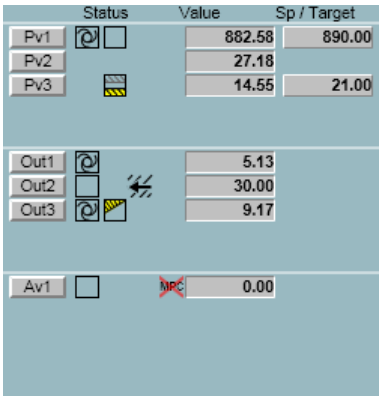

Figure 3.6. The control faceplate element. Example of different status indications.

Figure 3.7. The Control_Signal_Object. It contains a few components and the indicator objects. It is not connected to a specific type of signal but is designed to be used for all.

3.4.5 Future work

Certainly, the design of the faceplate element may be improved but without expanding the surface area or remaking the entire design the changes will be limited to being details. One of these details may be the labels and tooltip texts. They were chosen by the author alone and might need revision from an expert or a native English speaker.

3.5 Reduced faceplate element

3.5.1 Overview

The reduced faceplate element is the faceplate element shown in the main faceplate when selecting the reduced view mode. It is smaller than all the other faceplate elements discussed in this report with a surface of 250 by 80 pixels. It is in many ways similar to the control faceplate element and all of the components that it uses are also used in the control faceplate element.

3.5.2 Specifications

The reduced faceplate should provide an overview of the system status. It is the simplest and most compact of the viewing modes and a user only selects this mode when he or she does not need detailed information. Therefore it should contain limited amounts of data, and not present too many options. The presentation of the status should be intuitive and it is advantageous if it is similar to the presentations in other faceplates. The reduced faceplate element should also provide a way to access the signal faceplates.

3.5.3 Considerations

Many of the questions that arose when designing the reduced faceplate element were the same as for the control faceplate element. For discussion about the indicators see sections 3.4.3 Considerations and 3.4.4 Solutions for the control faceplate element.

In the same way as for the control faceplate element, a graphical object was created in GraphicSupportMpcLib and instantiated for each signal, except for the Av signals. Because of the limited space the Av signals were not represented in this way but as described below. The graphical object used was called Reduced_Signal_Object and it was decided to include an aspect link button, which also serves as a label for the signal, and the status indicators.
Due to the limited pixel space, the NoPredict indicator was placed on top of the right part of the right indicator. It indicates as normal except for when the signal is in backtracking mode or when the subsequent object has reached a limit. In which case it is hidden since the pixel space is then used by the right indicator. Each Av signal was represented with an aspect link button and the left indicator, which is sufficient for showing the status of an Av signal.

To get all fifteen signals to fit, the components had to be place a bit closer together than in the control faceplate element.

3.5.4 Solutions

A thin vertical line split the surface into three columns. The Pv signals were placed in the column to the left, the out signals in the middle and the Av signals in the narrowest column to the right. The layout of the reduced faceplate element can be seen in Figure 3.8.

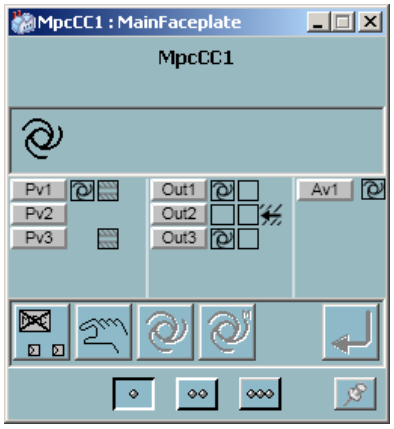

Figure 3.8. The main faceplate in reduced view mode, holding the reduced faceplate element.

3.5.5 Future work

There is not any pixel space left in the reduced faceplate element and therefore there is no room for additional components. However the current layout is thought to fulfill the specifications and hence additional components may not be needed. The tooltip texts may need revising as mentioned in the discussion of future work for the control faceplate element in section 3.4.5. The texts should probably be the same.

3.6 Connections faceplate element

3.6.1 Overview

The connections faceplate element is meant to give the user a short description of each of the fifteen signals. It should brief the operator of which signal is connected to what input or output.

3.6.2 Specifications

Usually at a plant, a lot of abbreviations and special names are used for the I/O equipment. There should be a text in the connections faceplate element that connect a signal with the name, usually called tag, of the physical object it is connected to. There should also be a brief description of the signal. Additionally, if there is room, the physical unit of the signal could be displayed here.

3.6.3 Considerations

Without any obvious other options, the connections faceplate element was designed to have a similar layout as the control and limits faceplate elements. Each row represents one

signal with the Pv signals at the top, followed by the out signals, and the Av signals at the bottom. Since the faceplate is not always used in combination with the control or limits faceplate, a signal label was considered necessary.

The idea to place a text presenting the physical unit of each signal was discarded due to lack of pixel space. This information is presented on the individual faceplates of the signals and this was thought to be enough.

A more problematic issue was how to store the data. The users should be able to write the tags and descriptions themselves, and this information should be stored. The connections faceplate element was not highly prioritized during the design work and a solution to this was never found.

3.6.4 Solutions

The signals are listed from top to bottom of the element, each row representing one signal. A row is divided into three columns: the signal label to the left, the tag in the middle and a larger description field to the right. Except for the signal label, the fields are editable and the users can thereby enter texts themselves.

3.6.5 Future work

A solution to the problem on how to store the tag and description texts needs to found. The sizes of the text fields may very well need to be changed according to standards or common lengths of tags.

3.7 Bar graphs faceplate elements

3.7.1 Overview

The use of bar graphs is a standard way to represent values of signals graphically in many 800xA applications. The idea is that it is easy for a user to get a good overview of the signals value even with just a quick glance at the bar graph. This is why it was suggested in the mockups, [8], that it might be good idea to have overview faceplate elements where many bar graphs were shown side by side.

3.7.2 Specifications

In the mockups it was suggested that the process variables and the out signals were to be shown in the same faceplate element. The Pv signals should be represented by vertical bar graphs and the out signals by horizontal. Every bar graph should have its own range and a scale to show it. The value should also be presented in numbers next to the bar graph. The bar graphs should be able to show constrains and setpoint for the signals for which it is appropriate.

3.7.3 Considerations

The decision to make the faceplate elements in extended view mode of standard size, discussed in section 3.2.3, made it impossible to fit bar graphs for all signals into one faceplate element. Instead they were divided into three categories: Pv, out and Av signals, and put on three separate faceplate elements.

3.7.4 Solutions

Three separate faceplates were created, one for each type of signal. They were made available in the extended view mode of the main faceplate. To follow the standard the Pv and Av signal bars were vertical while the out were horizontal, as in the separate signal faceplates.

Text fields, to show the value of the signal in numbers were added next to each bar graph, but later discarded since there were not enough room to make them standard sized.

3.7.5 Future work

The idea of adding text fields to show the numerical value of each signal may need to be reconsidered since it could indeed prove helpful to a user. The problem with placement and size needs to be solved in that case. Tooltip could be added, either to show the signal names or perhaps even the value, or both.

3.8 Sampling time info

3.8.1 Overview

Since the sampling time in an MPC can be relatively long and a change made in the interface might not be registered until the next sample a user may get confused as to why the MPC is not reacting to the change. To counteract this and to give information about what the sampling time for the current system is the sample time info faceplate element was created.

3.8.2 Specifications

There were no previous specifications for this faceplate but it was thought up during the design work. The sample time info faceplate element should show the user how long the sampling time is. It should also show how long time is left to next sampling instant or how much has time that has passed since the last one. The faceplate element is to be added to the main faceplate and visible in the extended view mode.

3.8.3 Considerations

Per-Erik suggested a clock-like symbol with a hand moving around and showing the time left to next sampling instant. One lap around the circle would represent one sample. This was a very good idea since the clock symbol is very intuitive and most people are used to it. It was created and a small triangle was added at the top of the circle to indicate that that is where the sampling happens. The triangle temporarily changes color the moment the hand passes it.

A difficulty encountered was to get the hand moving in a smooth fashion. Some effort was put into this but without satisfying result. The hand will make a move and rotate each second, each rotation being 360 divided by the sample time degrees.

The prototype control module has two parameters that hold the data shown in the faceplate element, one of them tells the time since last sample. The value of this parameter seems to be drifting slightly. This might be because of numerical calculations or some other approximation error. Either way it bring about the effect that the hand might not point straight up at any time witch led to the condition of changing the color of the triangle had to be remade.

3.8.4 Solutions

At the top of the faceplate element two text fields show the value of the sampling time and the time since last sample. Below these a large circle and an arrow with one end in the center of the circle, see Figure 3.9. The arrow is rotated by setting its transform property as Expression 3.1. At the top of the circle is a triangular frame that is filled with green color at the time of sampling. The condition for changing color in the triangle becomes true if the hand is about to pass it or if it has just recently passed it. Expression 3.2 shows the expression, the numbers are 0.9 and 0.11 since the sampling time for the graphics is one second and thus it will always be lit during one sample each lap.

```
RotateAt(125., 175., ev::RotationAngle)

→ev:RotaionAngle definition←
360. * ip::TimeSinceSample#RealVal / ip::SampleTime#RealVal
```

Expression 3.1. Shows how the transform property of the hand was set.

```
if ev::Sampling then
    DarkGreen
else
    Transparent()

→ev:Sampling definition←
if ip::TimeSinceSample#RealVal + 0.9 > ip::SampleTime#RealVal ||
ip::TimeSinceSample#RealVal – 0.11 < 0. then
    True
else
    False
```

Expression 3.2. Shows how the transform property of the hand was set.

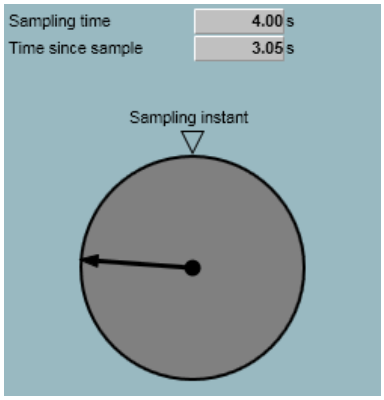

Figure 3.9. The sampling time info faceplate element.

3.8.5 Future work

There is considerable amounts of pixel space left for added components and if more is needed the circle and arrow can be shrunk without loss of clarity. This faceplate may be a good place to add additional information about the MPC and its operating environment.

3.9 Pv Signal faceplates

3.9.1 Overview

The signal faceplates are the fifteen faceplates that each corresponds to a signal used by the MPC controller. There are five faceplates, one for each, for the process variables, Pv, and its setpoint, five for the five output signals, and one for each of the five assisting variables, Av. As just mentioned the process variable and the setpoint are not to be thought of as deferent signals in this point of view. In similar manner as the process variables might have setpoints, the out variables might have optimal values. The assisting variables are used only to support the prediction and optimization and have neither setpoint nor optimal values.

There are two types of constraints used in this MPC controller. One type is the fixed value constraints and the other is deviation constraints that limit the discrepancy from the setpoint or optimal point value. The Av signals can not have constraints of any kind since they are either feed forward variables or process variables solely used for estimation and hence they are not to be controlled in any way. Both types of constraints can be active at the same time as well as the setpoint.

The Pv signal faceplates will be accessed by the user by clicking the corresponding button in the Control faceplate described in detail below. This will open a new faceplate window, with only the standard view mode available, containing the signal faceplate element.

3.9.2 Specifications

The process variable signal faceplate should show a standing bar graph of the current value and the setpoint. Both the current value and the target value in the case of limited changing speed, or ramping as is it called, of the setpoint should be shown. The two setpoint values and the process value should also be shown in text. The target setpoint should be changeable by the user, if the signal is in the right operating mode. The Bar graph should also be able to show the constraints in an intuitive way. The faceplate should also give the

user some way to decide what, if any, type of constraints he or she wants to use and the value of those. The faceplate should react on the current mode somehow so that it is clear what mode that is currently active and only present options appropriate for that mode.

Possible operating modes are:

- Automatic mode
    - Setpoint used for feedback control
        - The setpoint uses the internal value, set by the operator in the current faceplate.
        - The setpoint uses an external value.
    - Setpoint not used for feedback.
    - Constraints used in control.
        - Upper constraint used.
        - Lower constraint used.
        - Deviation constraint used. The constraints are set at the given distance below and above the setpoint, regardless if this setpoint is used for feedback control or not
- Prediction mode, the signal is not controlled, but the value is taken into consideration when predicting the future state of the system.
- The whole Pv signal in off mode, its value will not be taken into account in the prediction and optimization. No control of this signal will be performed.

By other means than through the faceplates, a system engineer at the site of use may set whether setpoint or constraints should be available for control at all. If set to unavailable, they should not appear as an option to the operator. If setpoint and/or constraints are available and used in control, the signal is said to be working in auto mode. When a Pv signal is set in automatic mode but nether setpoint nor constraints are used, it is effectively running in predict mode and this should be clear to the user.

3.9.3 Considerations

Both constraints types can be activated simultaneously on a process variable so the question of which to show on the bar graph arose. A rather quick decision was made, only the narrowest combination should be shown. To show all would get confusing and to just show one type could be misleading.

To show the target setpoint value when it is not the same as the current a thin flashing line was introduced to show the current value in such situations. However this idea was discarded as being to attention stealing. Instead it was decided that a similar but semitransparent arrow as which shows the target setpoint should indicate that "the current setpoint value is different than the target and is currently at this level". If the two values coincide the semitransparent arrow will be hidden behind the other one. There was a problem with making the current setpoint arrow following the current value in a smooth fashion, instead it moved stepwise until the current value reached the target value and the current value arrow and was hidden behind the target value arrow. This turned out to be an effect of the relatively slow sampling time of the controller, the arrow moves each time the

variable is updated. However, this was considered a minor problem not to be addressed in this phase of the project.

The target setpoint arrow mentioned above was implemented using a drag handle. A drag handle is an input item that catches mouse click-and-drag operations. The drag handle keeps track of the distance from the position where the mouse click occurred and the current position of the cursor. When the mouse button is released the drag handle writes a value to a property. To get the graphical effect of the arrow moving along the cursor when it is dragged the transform property of the arrow was set to a move command including the drag handle value property. Another issue was to get the drag handle to write a value corresponding to the arrows graphical position and take into account the constraints and the current scale of the bar. How this was done is shown in detail in Appendix A.

An operator should be able to change the values of constraints before activating them and therefore the input fields for the constraints was set to visible and write enabled unless the constraint are not available for control on that signal.

It was decided that the option to use setpoint or constraints for control is to be represented by checkboxes; this is an intuitive graphical object that will also indicate the current state. A situation where the Pv signal is set to be operating in auto mode but where all the checkboxes are unchecked would result in the signal effectively operating in predict mode. To prevent any confusion about this either the mode indicator must change or the user should be prevented from unchecking the last of the checkboxes.

The question of how to change and indicate the different operating modes was given a lot of thought. Since changing mode is a task that should easily be accessed this should be done with buttons below the faceplate element. To indicate which mode is active, indicator icons are shown above the faceplate element in an area designated for this, see Figure 2.4. A detailed explanation of how this was done, both for the Pv and the Out signals, is given in Appendix B.

3.9.4 Solutions

If a system engineer makes setpoint or constraints unavailable for use in control, the associated components were hidden. This renders the operator unable to run the MPC in a mode that is not intended for the current set up. However if both setpoint and constraints are available for control the operator can choose to use both, either one or none of them. As mentioned above the option to use setpoint for control is presented as a checkbox, this will also indicate if the setpoint feedback is currently in use or not. As for the constraints there are three checkboxes which represent the options to use high constraint, low constraint and/or deviation constraint respectively. If any of these three is checked the signal is considered being controlled with regard to constraints. If only the setpoint checkbox is checked, or only one of the three constraint checkboxes, that last checked checkbox is filled with gray color to indicate that it cannot be unchecked, making it impossible for a user to deactivate all control methods at once. This situation can be seen in Figure 3.11. This was accomplished with changes in the CheckBoxRef_resizable graphical object.

The final design for SignalPv faceplate can be seen in Figure 3.10 and Figure 3.11.

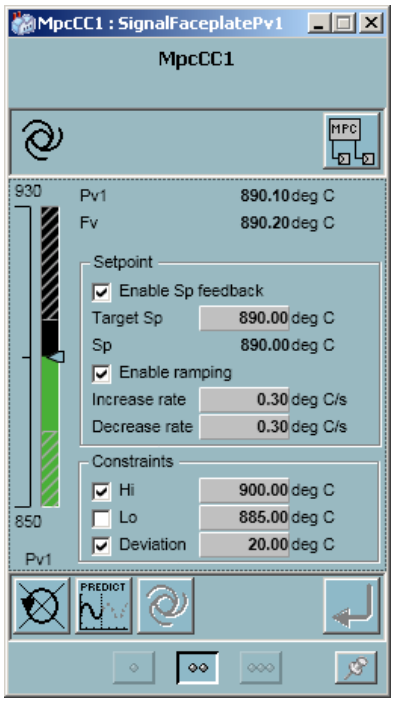

Figure 3.10. The Pv signal faceplate. Only the narrowest combination of constraints is shown in the bar graph.

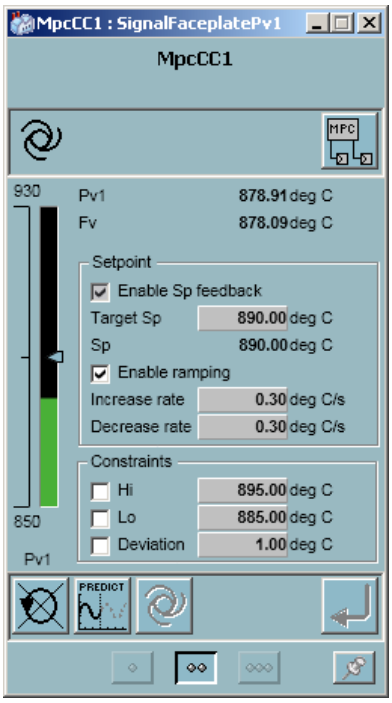

Figure 3.11. The Pv signal faceplate. The "enable Sp feedback" checkbox can not be unchecked since this would change the operating mode of the signal.

3.9.5 Future work

Due to practical reasons the standard graphical object for a checkbox, CheckBoxRef, was not used. Instead a modified version, CheckBoxRef_resizable, was created with CheckBoxRef as a base. The differences are that the modified version can be resized without distorting the text and that the checkbox itself can be disabled without graying the text. This in addition to the normal disabling function. It is possible that this functionality could be achieved without the use of a custom graphical object, but with standard objects alone. Apart from this, if the custom object is to be used, two details to address are that the mouse-over event and the OPC-bad status do not show properly. The mouse-over event is supposed to highlight the object with a thin orange line but when the checkbox object is resized and another object is placed next to it, it does not show as intended. The OPC-bad status is supposed to show a red cross over the checkbox and text but when it is resized it does not show as intended.

In addition, alarm handling, operator note and yet other standard features of faceplates in 800xA were not considered at all and require future attention.

3.10 Out Signal faceplates

3.10.1 Overview

The out signals are the values calculated as control values by the MPC algorithm. Just as for the Pv signal faceplate the out signal faceplates are accessed by clicking the label button in the Control or Reduced faceplate element. This will open a separate window showing the faceplate.

An MPC is often used as a parent object with its out signals used as setpoints for PI or PID controllers which in turn control the actuators. Therefore it is not uncommon that an operator wants to set manual values for one or several of the out signals. However it is important to keep in mind that when setting a specific out signal to manual mode one is impeding on the functionality of the MPC since it reduces the degrees of freedom it can operate in, thus possibly creating the situation where it is impossible to control the process in the demanded direction.

3.10.2 Specifications

The out signals should be viewed both in text and graphically, the current standard is to show a horizontal bar graph, similar to the vertical one for Pv signals. If the signal is in manual mode it should be possible to enter the value both numerically and graphically. As well as rate limitations, the out signal can have constraints in similar manner as a Pv signal, although no analogy to deviation constraints needs to be implemented. An out signal could have an optimal point. This sets the MPC algorithm to trying to minimize the discrepancy from it while still controlling the process accurately, if possible. An optimal point should be indicated in the graphics.

If rate constraints are active, also referred to as ramping, the current out signal value may differ from the target value. Both these values should be shown.

3.10.3 Considerations

It was decided that the optimal point functionality was not to be included in the prototype MPC and hence no support for it needed to be included into the faceplates. The bar graph was modified to be able to take input by click-and-drag action from the user and the gray arrow, that indicated setpoint on a PV, was removed.

Some thought was given to the issue of how to show, in the bar graph, that the target out value differs from the current. It was decided that only the current value was to be shown since it was considered the most important piece of information and if they differ the operator should be alerted by an indicator that the out signal is increasing or decreasing at its maximum rate.

The input fields, where the numerical value of the constraints is entered and displayed, was set to be shown at all times, even if the constraints were inactive. This gives an operator the opportunity to change the constraint value before activating it.

The option to set an out signal in manual or auto mode, a basic and very important option, is presented by two buttons at the bottom of the faceplate, showing the standard icons of manual mode and auto mode. The same icons are used in the indicator field in the upper part of the faceplate to indicate which mode is currently active. The option to isolate the

signal from the MPC prediction, and thereby not using it for control at all, was also considered important and frequently used and was therefore implemented in the same way with buttons and indicators in the faceplate. The current mode may depend not only on what the user has requested in the faceplate but also of what mode the MPC is operating in. If the subsequent connected object is not set to auto mode, the out signal will operate in backtracking mode. It will then attain the value of that object and change as it changes, regardless of what mode requested in the faceplate. How this is indicated is explained in detail in Appendix B.

An item of concern was how to handle the situation with the user switching from manual mode during ramping of the signal. The question was then; should the value stay at its current or continue ramping towards the target value? A similar situation arises when the MPC is turned off for the signal, i.e. the signal is isolated. In the early design, the value would instantly change to become zero, although respecting rate constraints if active. This may seem natural and appropriate but one can argue that zero is also a value and not more suitable than any other random number. The other option was to leave the out signal at its last value before the MPC was turned off. This was thought to be the better option and was implemented since that will probably not make such a big impact on the process state. To make the interface consistent, the issue with a user changing to manual mode while the signal is ramping was resolved in similar way: the signal keeps its current value from the instant before the change, it does not matter what the target value was.

3.10.4 Solutions

Since it is the standard for out signals in faceplates in 800xA to have their bar graphs horizontally, the bar graph was for the out signal was placed horizontally and at the bottom of the faceplate element. The bar graph supports constraints in similar manner as the Pv bar graph although there is no equivalent to a setpoint or deviation constraints. Therefore, there is no setpoint arrow but a user can affect the value directly by clicking the value bar in the bar graph and dragging. For the bar graph to take input in this way, the out signal must be operating in manual mode.

Value constraints and rate constraints are activated by a checkbox and the constraint values are entered in input fields. At the top of the faceplate element, two values are shown: target value and the current value. When the user wants to affect the value of the signal, always the target value is modified. It may not be possible for the current value to achieve the same value because of backtracking, tracking or rate constraints. If the signal is operating in manual mode and rate constraints are active when a big change in the target value is made, the current value will start to change according to its maximum or minimum limit. If the signal mode is changed to auto mode, the target value will instantly be set to the current value, as discussed above.
Figure 3.12 shows the out signal faceplate in auto mode. The target value input field cannot take any input since the MPC determines the value of the signal. In Figure 3.13, the signal is operating in manual mode and the user can set the target value either in the input field or by clicking and dragging the value bar in the bar graph.

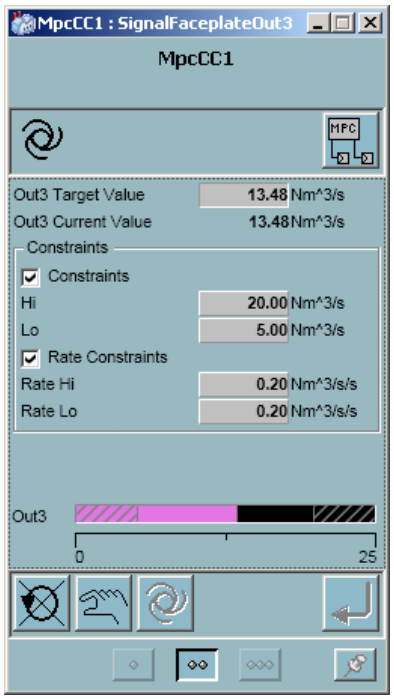

Figure 3.12. An out signal operating in auto mode and therefore the target value is not set by the user but by the MPC.

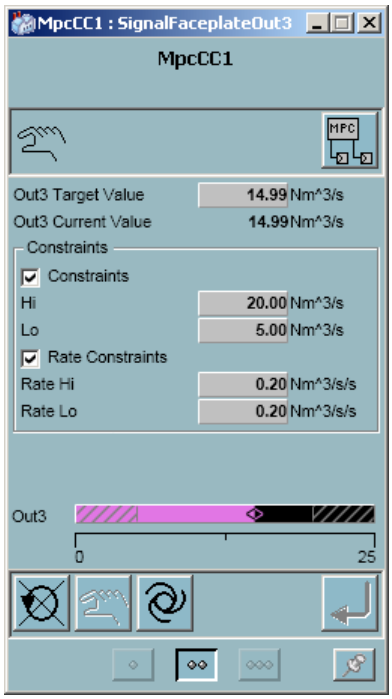

Figure 3.13. An out signal operating in manual mode. The user may change the target value either by the input field or by clicking and dragging in the bar graph.

3.10.5 Future work

The tracking functionality has not been tested at all and will likely need some attention. The option to disable value constraints is currently not supported in the prototype control module and therefore the upper of the two checkboxes is always checked and it cannot be unchecked by the user. On many occasions, the faceplate has been crashing when the value bar in the bar graphs clicked and dragged. This problem is thought to be related to the early versions of the new graphics and will hopefully cease to exist with future upgrades. As can be seen in Figure 3.12 and Figure 3.13 there is pixel space left in the faceplate element. It may prove useful if future development suggest additional components should be added.

As for presenting the unit of the rate constraints, it has been hard-coded into the component to add "/s" at the end of the unit string to denote that it is the change of unit per second that is set. As can be seen in Figure 3.12 and Figure 3.13 the unit itself may end with "/s" and this may not look agreeable, although it is fully correct.

3.11 AV Signal faceplates

3.11.1 Overview

Assisting variables, in short Av, are values used by the MPC to improve the prediction. They don't have setpoint or any other method to be controlled. Assisting variables can be

feedforward variables as described section 2.1.3. They may also be process variables that are not controlled but only used in the prediction.

3.11.2 Specifications

The MPC controller has support for five Av signals. Not all of them may be connected at an application and this is set up by the system engineer. Just as for Pv and out signals it should be very clear to an operator how many of the signals used in the application.

An Av may be turned off which means that it is disregarded from the MPC prediction calculations. Other than this, there are no options for an Av that needs to be set by an operator.

3.11.3 Considerations

It was decided to treat the Av signals in similar manner as the Pv and out signals. Clicking on the signal label in the Control or Reduced faceplate element will open a new faceplate that shows the value of the signal both with numbers and graphically by a bar graph. The option to turn the Av signal off was at first accessed a checkbox in the faceplate element but since there is no equivalent to an auto mode, it was decided that it would instead be by two buttons in the faceplate. An indicator at the top of the faceplate shows which mode is currently active.

3.11.4 Solutions

The Av signal faceplate element can be seen in Figure 3.14. It has a bar graph that is vertically oriented since the Av is considered an input to the MPC. The value is represented in numbers at the top of the faceplate element. The indicators and buttons used in the Av faceplate are descried in detail in Appendix B.

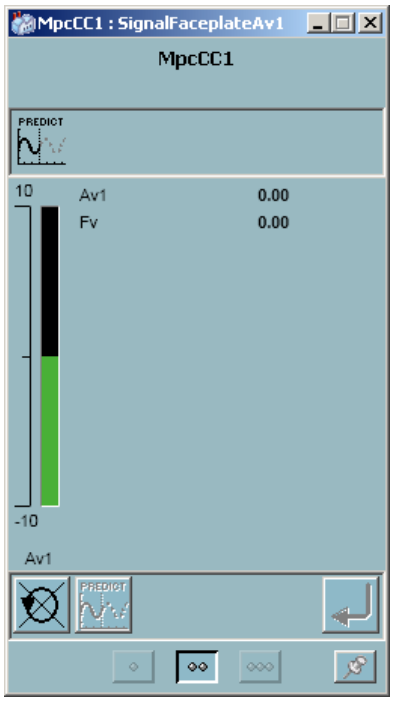

Figure 3.14. The Av Faceplate. It has comparably few components and options.

3.11.5 Future work

At its current function the Av signal faceplate has few components and options and there might not be much to improve. If new functionality is to be added, there is plenty of pixel space available.

4 Conclusion and discussion

4.1 The design work progress

The faceplate design work was not structured the same way as it is described in this report but rather in a fragmented way where minor problems were solved as they arose. When critical problems hindered work progress, the focus was often shifted to another area for a period of time. On several occasions the work was completely halted and could not progress at all due to various software problems. The problems included applications crashing, probably due to the unfinished form of the new graphics system, licensing problems caused by incompatible software versions and on at one occasion waiting for crucial software updates. During the time periods when the design work could be carried out, large parts of this report was written and the simulation of a plant connected to the MPC was set up.

As expected and mentioned in section 1.3, the limited space was a big challenge and often led to decisions of not adding a component even though it would be presenting valuable information. Another challenging part of the design was to make the new design solutions specifically made for the MPC faceplates look and feel like similar to other faceplates in 800xA and the standard. For as much as it was possible, standard graphical object were used, and standard pixel distances were used. A big challenge that was unforeseen was the selection of which parameter to connect to a component. The MPC prototype control module holds several hundred parameters and many of them have similar names. Before the underlying structure was known, realizing what information a parameter holds was very difficult. Also the mode selection and indication proved to be a bigger challenge than expected. Partly because of the fact that the MPC is quite unique in the 800xA system and no similar solutions exists. Partly because there are many combinations of modes and special cases that required attention.

Most of the first planned faceplates were created but several had to be left for future work. These not considered faceplates were: tuning, excite/identification experiment, trim curves for both out and Pv signals. The "tuning" faceplate was meant to be used for loading and changing between different tunings for the MPC. The "excite/identification" experiment would provide a way to start a process identification experiment directly from the faceplates. While the trim curves for out and Pv signals should not only show the history of the process values on chart but also the predicted values of the future.

4.2 The MPC as a control module

As stated in section 1.2 the purpose of the project that the faceplate design is related to, is to examine the possibilities of implementing an MPC controller in 800xA. This might be a very good idea, making an MPC controller much more available for smaller applications. The concept of having an MPC as a control module in 800xA makes the effort of using an MPC as a high level controller much smaller compared to installing a complete new software package. The MPC will, in most cases, not replace the current setup of PLC[4] but

[4] Programmable Logic Controller, a commonly used abbreviation for digital logic controllers often using P, PI or PID algorithms.

will simply provide setpoints for it. Additionally it may indeed prove as an advantage that the MPC algorithm is running on the hardware controller unit as opposed to running on a PC communicating with the hardware controller, because of the increase in system stability and fail-safe possibilities.

4.3 Faceplate design discussion

Some issues of the faceplate design were not limited to a single faceplate element. Decisions were made that would affect several, if not all of the faceplates or faceplate elements. One such issue was deciding a word for describing the situation when a value is higher than the high constraint or lower than the low constraint. As the reader probably has noticed the word "violated" was chosen. This may not be a good choice of word since it might be perceived slightly different to a native English speaker, but it was chosen at the best of the author's knowledge.

As stated in 3.4.3, the main concern when designing the control faceplate element was pixel space, As an overview and starting point when operating the MPC the current layout and design is probably as good as it can get without expanding the surface. Since the indicators are rather small and not part of any existing application, they may not be all that easy to understand at first glance but they are thought to be a very good source of information for a more experienced user.

To the MPC, a Pv signal operating in predict mode is in actual fact working as an Av that is not used as a feedforward variable. Therefore it might be a good idea for future work to replace the Av signals with Pv signals, possibly adding the option to use its value as feedforward data. This could increase the flexibility of the MPC control module without removing any functionality. Since the MPC algorithm used is based on the Predict & Control application, which can support much higher number of process variables than five, this should not implicate a very big work effort. The MPC faceplates would be easily modified to incorporate this change.

As mentioned above, not all faceplates intended, were created created. The ones left for future work were the ones least critical to normal operation of the MPC. Also, some of the functions suggested in those might be a good idea to not include into the faceplate. For example the identification experiment will not usually be conducted by an operator but by a process engineer who might require a more powerful tool anyway. Therefore it is suggested that these more advanced functions are developed not for the faceplates but into separate tool.

5 Glossary

This section lists words and abbreviations that are used frequently throughout the report and that are important to know.

Word	Description
800xA	The name of the ABB control system for which the *MPC* is developed.
Aspect	An object in the *800xA* data structure. Can be thought of as a file in a common file system.
Assisting variable	Short Av. A variable holding a value that is used just to assist the prediction calculation. Can be a *feedforward* variable or a non controlled estimated state in the *plant*.
Constraints	Limits that have a special meaning to the control algorithm. A value may be higher than the upper constraint or less than the low constraint, in which case the constraint is said to be violated.
Control horizon	Denotes how far into the future the control signal at that time is taken into account in the *MPC* algorithm.
Control module	A piece of programming logic made according to certain specifications in *800xA*. It usually has a specific purpose and function and may easily be connected with other objects.
Control signal	The signal that is calculated by the *MPC* to control the *plant*. In the faceplate design referred to as out signal since they come out of the controller. See also *Manipulated variable*
Manipulated variable	The signal that is generated by the controller to control the *plant* In some contexts called *control signal*, but in this faceplate design called out signal.
Cost function	The function used in the *MPC* algorithm to determine the optimal control signal.
Extended view mode	One of three view modes available for faceplates in *800xA*. The extended view mode shows all information to the user. Often holds several view panels and tabs.
Faceplate	An interaction window. Used to communicate information and take input from the user. In *800xA* the faceplate aspect is the framework of the interaction window holding the *faceplate element* that is the actual surface.

Faceplate element	The interaction surface used in a *faceplate* to communicate with a user. Editable with the Graphics Builder application.
Feedforward variable	A variable holding data that is used in the *MPC* algorithm to estimate and predict the state of the *plant*. It is not controlled but only used as information to make more accurate estimations.
Model	The mathematical model of the *plant*. Usually determined through system identification experiments. The model is never completely accurate.
MPC	Model Predictive Control. A control theory using a model of the plant to predict its future behavior. Using this, it calculates an optimal output. The MPC handles *constraints* well due to its use of a *cost function*.
New graphics	Refers to the upcoming graphics standard for ABB process graphics. It is based on Microsoft's WPF. Also called Process Graphics 2.
Plant	The actual physical process that is to be controlled.
PPA	Process Portal A. A frequently used application in ABB control system *800xA*.
Prediction horizon	The number of samples into the future that the *MPC* algorithm calculates predictions of the *plant*.
Process variable	Short Pv. The controlled variable. A physical value in the *plant* that is to be control by some method.
Reduced view mode	One of three view modes available for faceplates in *800xA*. A smaller window which shows nothing but the most critical information.
Setpoint	The value to which it is desirable that the *process variable* is equal or close to.
Standard view mode	One of three view modes available for *faceplates* in *800xA*. Shows more information than the *reduced view mode* but it contains merely a single faceplate element.
Status icons	The icons in the upper part of the *faceplate*. They indicate the status of the current object.
WPF	Windows Presentation Foundation. The latest system developed by Microsoft for showing graphics.

6 References

[1] About Windows Presentation Foundation
http://msdn.microsoft.com/en-us/library/ms752061.aspx, 2008-10-14

[2] "MPC in AC 800M Requirements specification", Lars Mårtensson and Per Erik Modén, document number 9ADB001012-009

[3] PowerPoint presentations "FaceplateMockups" and "SignalFaceplateMockups" by Per Erik Modén

[4] Maciejowski J.M., "Predictive Control with constraints", Prentice Hall, Pearson Education 2002, ISBN 0-201-39823

[5] System 800xA 5.0 SP 1 Engineering Graphics, Document number: 3BSE030335R5011 June 2007.

[6] "Expressions for Graphics DoF", Document number: 3BSE042967, January 2008

[7] "Process Graphics Description of function", Document number: 3BSE043693, June 2008

[8] FaceplateMockups.ppt and SignalFaceplateMockups.ppt, Per-Erik Modén, month year, Document Number

Appendix A

This section describes how the drag handle for the setpoint arrow in the vertical bar graphs was configured. The section can also serve as an example of to configure components in Graphics Builder.

The setpoint arrow and the drag handle

This section has been hidden. It is hidden because the section is not of any academic interest and it lies in the interest of ABB to keep this information within the company.

Appendix B

Faceplate indicators and buttons

This appendix explains the MPC mode handling; it describes in detail how the indicator icons and interaction buttons for all the faceplates were configured.

Often considered a primary and very common task for an operator, is changing the current operating mode of an object. This task includes knowing the current mode, changing the mode and confirming the change. The faceplates in the 800xA system have a standard way to present a way to accomplish this. At the lower part of the faceplate, below the position of the faceplate element, there is a designated button area. A similar area for indicators is located above the faceplate element. See Figure 2.4. To create and configure buttons and indicators the "config view" interface accessed from PPA right-click menu, is used. This interface is described in detail in the manual for 800xA engineering graphics [5].

The mode switching buttons will affect interaction parameters in the control module. Because of limitations in the faceplate "config view" interface a button can only affect a single parameter. Therefore a few interaction parameters were created and used as switches to set others appropriately, their name end with Cmd.

The indicators are set up in groups by their placement, counting from the left. The icons that should be shown in the selected position are added to a numbered list. In the expression field an expressions is entered. The expression returns a number and the corresponding icon becomes visible. The only expression allowed for use is the immediate if, IIF, statement. It is used as: IIF(*statement*, *truepart*, *falsepart*). The *truepart* is returned if the *statement* is true and the falsepart if the statement is false. To get additional results the IIF needs to be nestled. Note that even if the *statement* is true, the *falsepart* is excecuted and evaluated which could result in more than one value to return. If this happens no icon is showed but a dotted square appears instead.

In the tables in this appendix numerous parameters are referred to. Most of them are considered self explanatory otherwise a short explanation is given. The variables are written in shortened form for convenience, see Table B.1 for an explanation. The DesignValid parameter is set true if a design and tuning is successfully loaded and is therefore a prerequisite for the MPC to work as intended.

Table B.1. Translation for the short form of writing parameter names.

Short form	Substitute for
I	InteractionPar
O	Any of InteractionPar.Out1, …, InteractionPar.Out5
P	Any of InteractionPar.PvSp1, …, InteractionPar.PvSp5
A	Any of InteractionPar.Av1, …, InteractionPar.Av5
PO	PvOptions
#	Any of the numbers 1 to 5.
Example: P.EnableSp	InteractionPar.PvSp#.EnableSp (# = 1-5, the Pv in question)

Every signal operates in a mode, as does the MPC which has own sets of modes just like any of the signals. The difference is that the MPC can be seen as a parent object to any signal and its mode affects all of the signals modes. The standard ideology of mode operation is that a restrained operating mode always overrules a more automatic one. If a signal is set to off/isolate mode and the MPC is set in auto mode the signal will still run in off/isolate mode, the same goes for the inverse situation where the signal is set to operate in auto mode but the MPC is turned off. An illustration of the MPC with its connected signals can be seen in
Figure B.1.

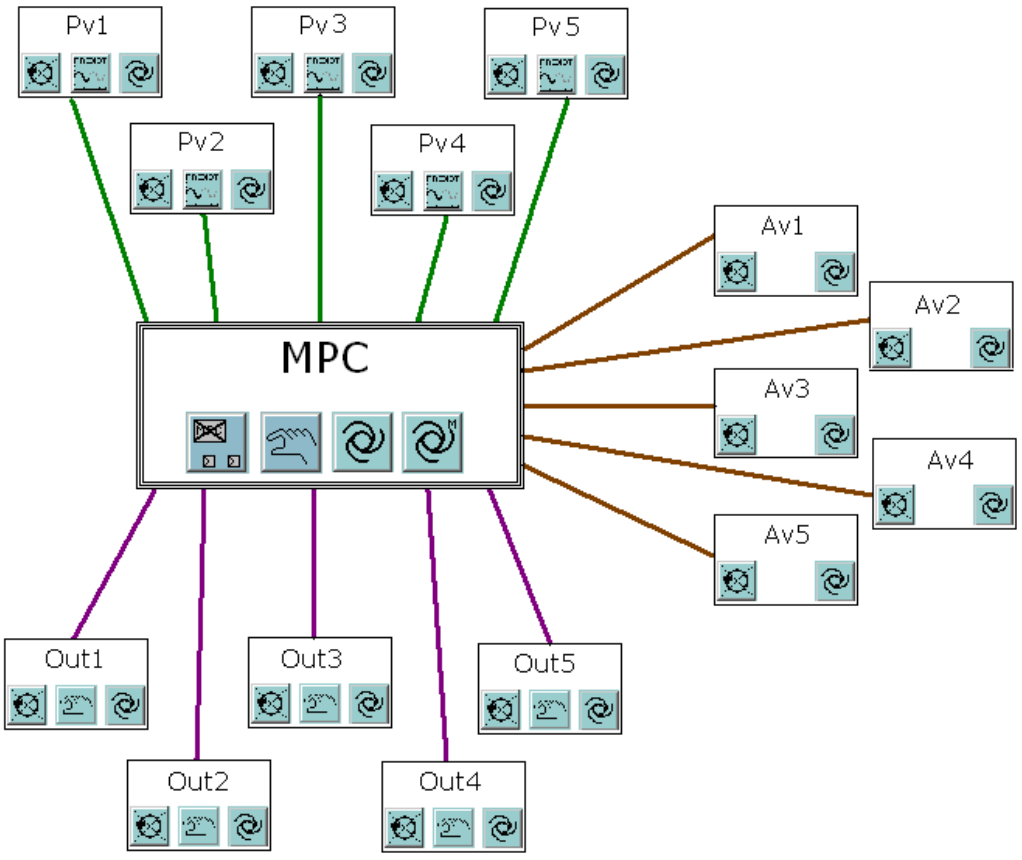

Figure B.1. An illustration of the MPC and its connected signals. The operating mode of the MPC affects every signal.

The rest of this section has been hidden. It is hidden because the section is not of any academic interest and it lies in the interest of ABB to keep this information within the company.

Appendix C

Display of the results

The faceplates were designed as described in chapter 3. In this section all of the faceplates and faceplate elements are displayed. See Table C.1 for figure references.

Table C.1. The figures shown in Appendix C.

Figure number	Description
Figure C.1	**The main faceplate** viewed in reduced view mode.
Figure C.2	**The main faceplate** in standard view mode.
Figure C.3	**The main faceplate** in extended view mode. Showing the Control, Limits and Sample info faceplate elements.
Figure C.4	**The main faceplate** in extended view mode. Showing the Control, Pv Bar Graphs and Out Bar Graphs faceplate elements.
Figure C.5	**The Pv signal faceplate.** The Pv is controlled by setpoint alone.
Figure C.6	**The Pv signal faceplate.** The Pv is controlled by setpoint, the high constraint and deviation constraints
Figure C.7	**The Pv signal faceplate.** The Pv is forced to operate in predict mode by the MPC even though the user has selected auto mode.
Figure C.8	**The Pv signal faceplate.** The Pv is disabled.
Figure C.9	**The Out signal faceplate.** The value is lower than the low limit and this is indicated with yellow in the bar graph.
Figure C.10	**The out signal faceplate.** The out signal is forced to be disabled even though the user has selected the auto mode.
Figure C.11	**The Av signal faceplate** operating in predict mode.
Figure C.12	**The connections faceplate** element as shown in the extended view mode of the main faceplate.

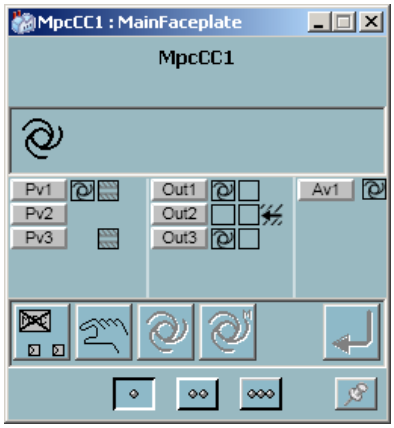

Figure C.1 The main faceplate viewed in reduced view mode.

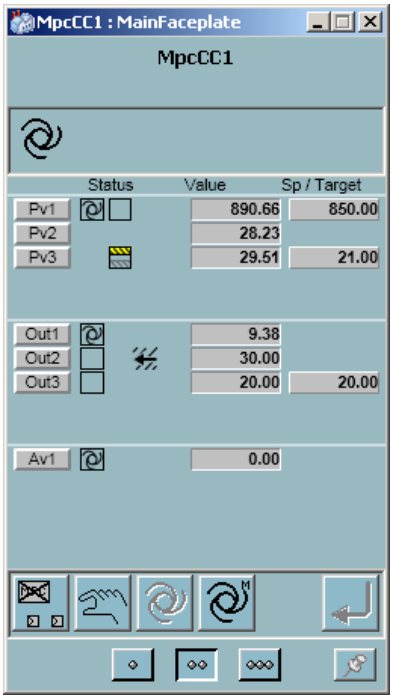

Figure C.2 The main faceplate in standard view mode.

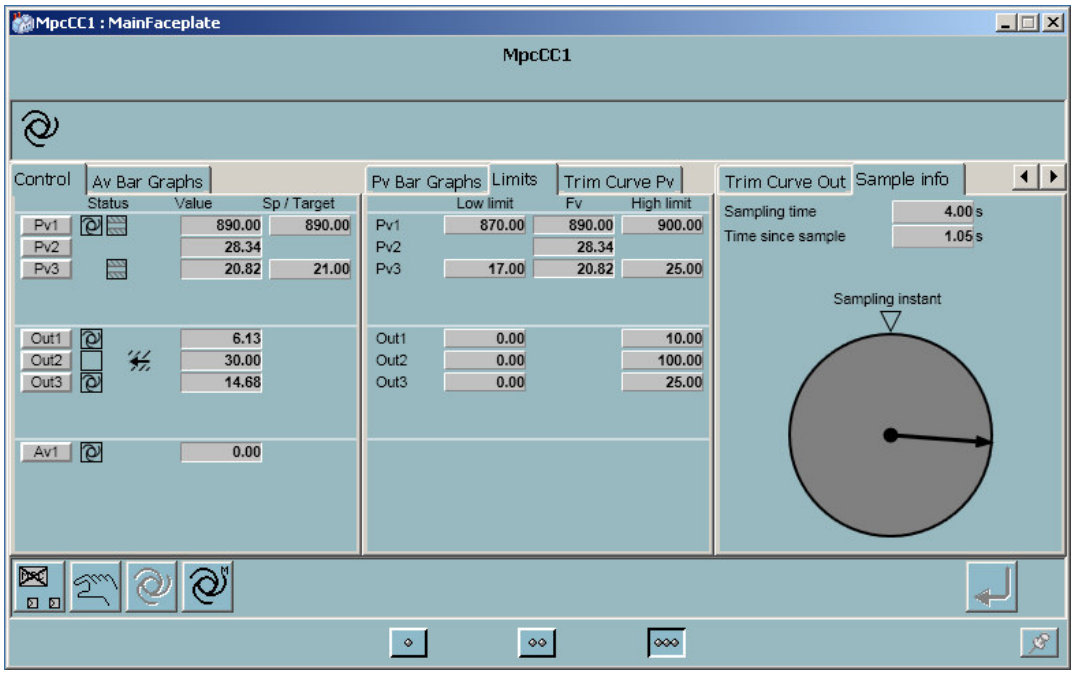

Figure C.3 The main faceplate in extended view mode. Showing the Control, Limits and Sample info faceplate elements.

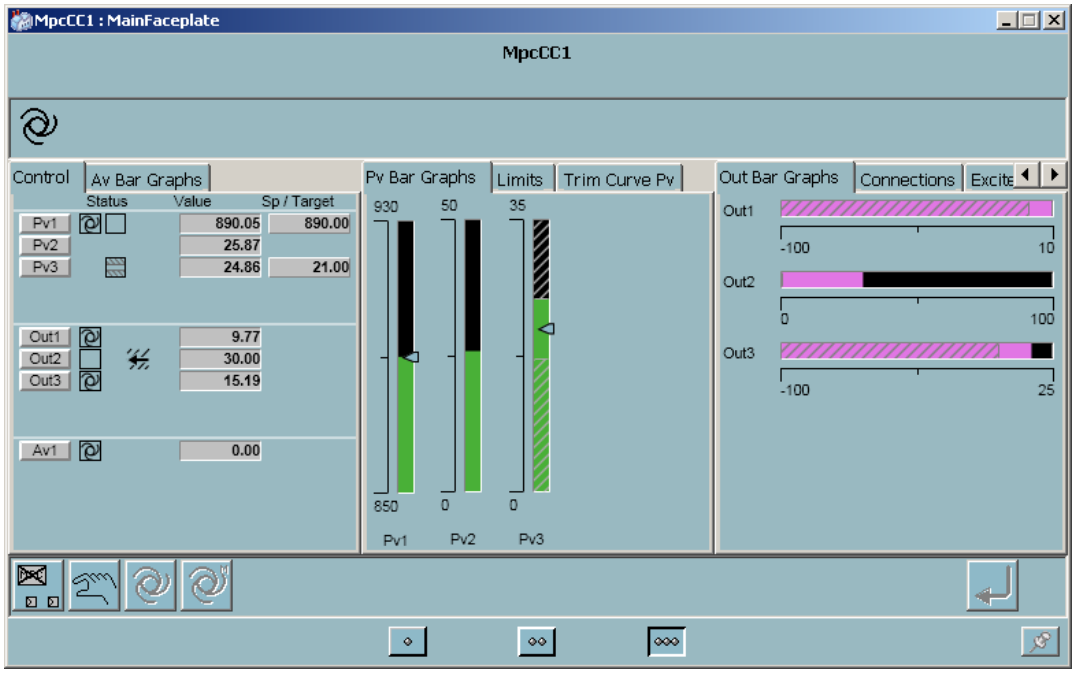

Figure C.4 The main faceplate in extended view mode. Showing the Control, Pv Bar Graphs and Out Bar Graphs faceplate elements.

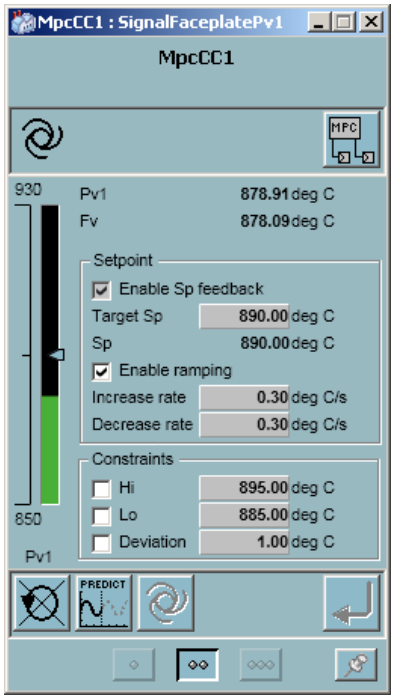

Figure C.5 The Pv signal faceplate. The Pv is controlled by setpoint alone.

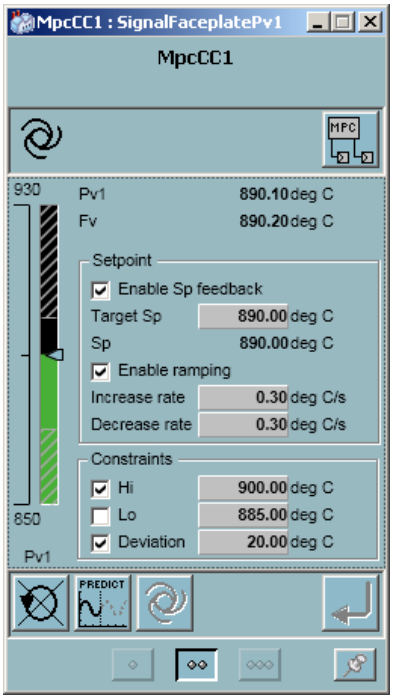

Figure C.6 The Pv signal faceplate. The Pv is controlled by setpoint, the high constraint and deviation constraints

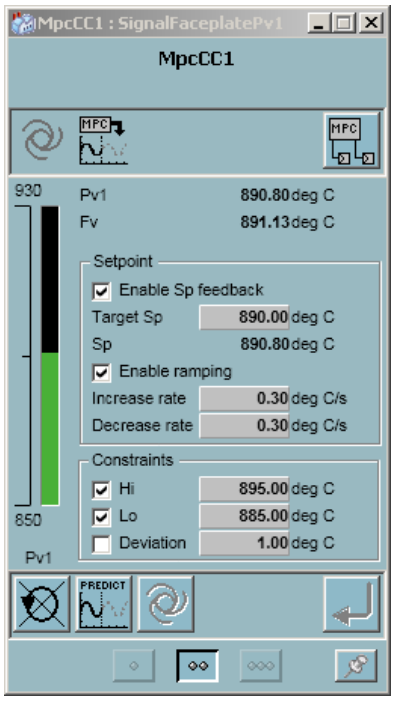

Figure C.7 The Pv signal faceplate. The Pv is forced to operate in predict mode by the MPC even though the user has selected auto mode.

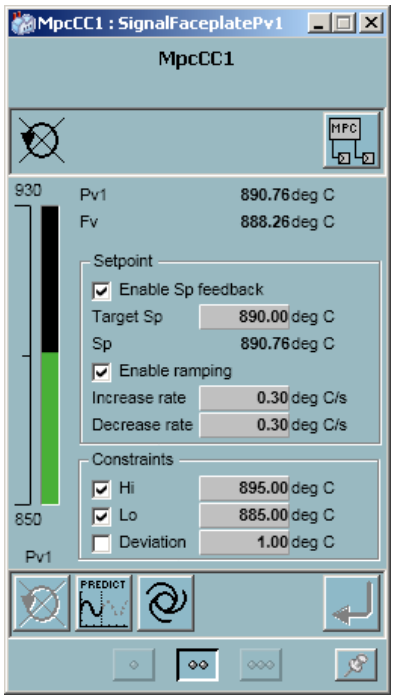

Figure C.8 The Pv signal faceplate. The Pv is disabled.

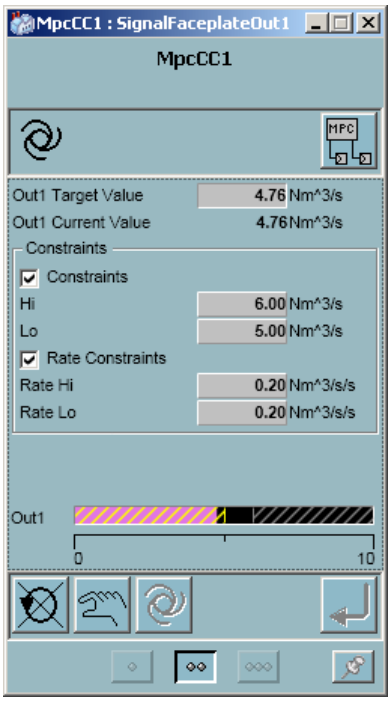

Figure C.9 The Out signal faceplate. The value is lower than the low limit and this is indicated with yellow in the bar graph.

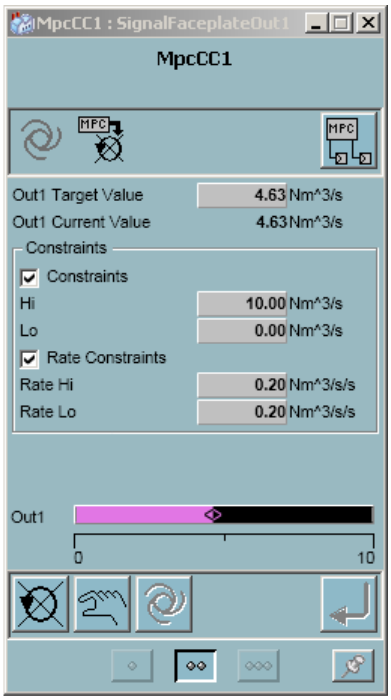

Figure C.10 The out signal faceplate. The out signal is forced to be disabled even though the user has selected the auto mode.

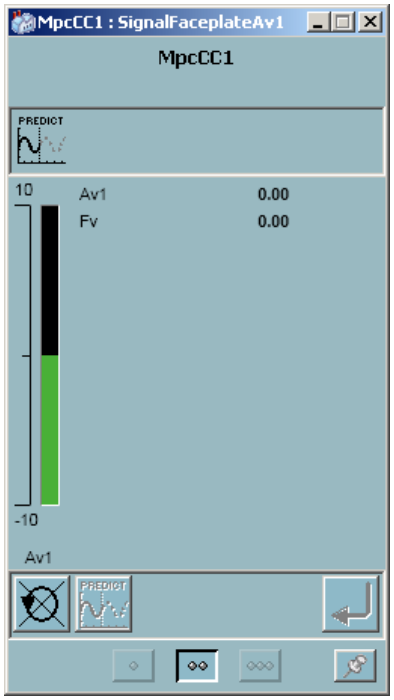

Figure C.11The Av signal faceplate operating in predict mode.

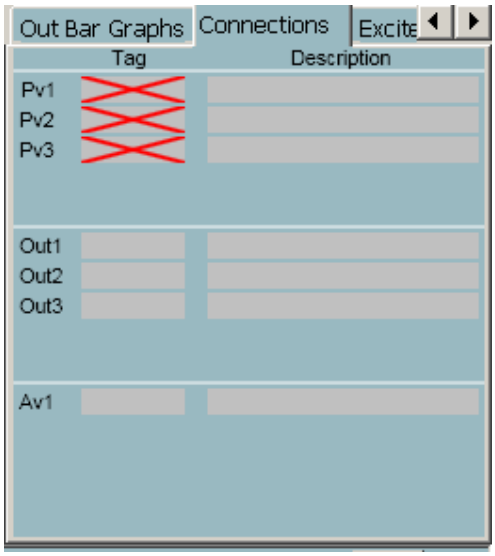

Figure C.12 The connections faceplate element as shown in the extended view mode of the main faceplate.