CODEN: LUTFD2/(TFRT-5428)/1-89/(1990)

An Expert System for ÖSI

Ola Mattsson

Department of Automatic Control
Lund Institute of Technology
December 1990

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
Master Thesis

Date of issue
December 1990

Document Number
CODEN: LUTFD2/(TFRT-5428)/1-89/(1990)

Author(s)
Ola Mattsson

Supervisor
Jan Eric Larsson LTH, Johnny Pettersson Sydkraft

Sponsoring organisation
Sydkraft AB

Title and subtitle
An Expert System for ÖSI

Abstract

This master thesis deals with the design of an expert system for the emergency operational procedures (EOP) of the nuclear power plant of Barsebäck. The project is primarily aimed at serving as a demonstration of the possibilities that lie within the expert system technique but also at developing a tool possible to be used for the training of operators. The thesis treats the structure of the operational procedures and the demands and restrictions this imposes on the implementation. The system is developed with an expert system shell named G2 from Gensym Corp.

Key words
Expert Systems, Knowledge-based Programming, Nuclear Safety, Emergency Operational Procedures

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title

ISBN

Language
English

Number of pages
89

Recipient's notes

Security classification

Preface

During the last few years there has been a strong interest within the industry for knowledge-based systems in various applications such as: expert control, on-line monitoring, and planning. This project, which deals with the design of an expert system for the Emergency Operational Procedures (EOP) of the nuclear power plant at Barsebäck, is the result of an aim to gain knowledge of such techniques. The work has been performed during May - December, 1990 at the Department of Automatic Control, Lund Institute of Technology in cooperation with Sydkraft AB.

Acknowledgements

I would like to thank my supervisor at the Department of Automatic Control, Lund Institute, Jan Eric Larsson for being such a strong support and for his inspiring guidance into the concept of knowledge-based programming which at the start was a whole new field for me. This gratitude also extends to Karl-Erik Årzen at the same department for sharing his insights in G2. I also want to thank my supervisor at Sydkraft AB, Johnny Pettersson who has given given me the opportunity to perform this interesting masters thesis and for his deep and not least patient support. Finally, I would like to thank Kenneth Zander of Sydkraft, the man behind the design of the EOP and the true expert in this field, for his views and opinions and for supplying me with additional material.

Ola Mattsson

Lund December 1990

Contents

1

Introduction

Main purpose

The objective of this master thesis is to translate the ÖSI-instructions (see chapter 3) used at the nuclear power plant of Barsebäck into an expert system. Presently, the use of such a system would not be possible in a real situation due to problems of guaranteeing totally error free performance. So rather than this, the implementation should serve as a means to demonstrate the expert system technique as a whole and also as a step in gaining knowledge about equipment that is likely to be used in a near future. Furthermore the system might find some use as a tool in the training of operators for ÖSI. The expert system shell which will be used in this task is G2 from Gensym corp.

The instructions of ÖSI are represented in the form of flow charts being by nature deterministic, containing no uncertainty, in the sense that the action to be performed next depends only on the result of the previous action and nothing else. We might even go so far as to claim that the ÖSI-instructions, at least in some respect, already represent an expert system, guiding an operator, through the use of the total knowledge gained by the experience of previous operators, contained in a tree-like structure. Bearing this in mind, we understand that the implementation will not demonstrate G2 or for that matter, the expert system technique itself, at its full power, if we by this mean concepts such as the testing of hypothesises or reasoning about uncertainties. Rather the emphasis should fall within finding a suitable graphical representation of the charts, displaying and storing the reasoning in an easy-to-follow animated fashion, and also within designing a man-to-machine communication with the system which, in a logical and compact form presents the operator with any requested information or knowledge about the present state of the process or retrieves information concerning actions previously performed.

2

Expert systems

This chapter will give a brief introduction to the concept of expert systems concentrating on systems using rule-based knowledge representation.

Introduction

Knowledge-based systems or expert systems is an area of AI that has expanded rapidly over the last few years. It is not very easy to give a definition that covers the term expert system but as a characterisation the following might serve: An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution. As opposed to conventional programming that relies on an algorithmic behaviour, expert system in particular apply to problems that lack such analytic solutions due to their being in some way too complex or ill-structured. The expert system instead tries to imitate the reasoning strategy and knowledge representation of a human expert. Another unique property of expert systems is that the knowledge of the system is incorporated explicitly in a separate part of the program, called the knowledge-base, being both readable and easy to modify due to it being built incrementally, where as in every conventional high-level language the knowledge is expressed in the ordinary program statements. Different expert systems use different methods of encoding the facts and relationships that constitute the knowledge of the system. Such approaches to use are: semantic networks, frames, logical expressions, rules, and object-attribute-value triplets. Of these only the last two will be examined in this chapter.

O-A-V triplets

Objects may be used to represent parts of the knowledge of a system in many ways: physical, conceptual, or abstract entities. The characteristic properties of an object is represented by its attributes which may obtain values through manipulations performed on them by other parts of the data base.

Objects are often structured into a hierarchy of classes and subclasses where the subclasses inherit the properties of the superior classes as well as they might include their own specific properties.

3

G2

This chapter contains an overview of G2, the expert system shell used in this project, concentrating mainly on the facilities actually used in the implementation.

Technical description

The main parts of G2 are: the knowledge-base, a real-time inference engine, a procedure language, a simulator, the development environment, the operator interface, and optional interfaces to external on-line data servers.

Classes and objects

The above-all class in G2 is the item or to put it in other words: everything in G2 including rules, objects, buttons, text boxes, graphs is an item! The items are ordered into a hierarchy and do all have graphical representations through which they can be manipulated in such ways as: cloning, changing size or colour, displaying its attribute table and so on, by the user.

The major item of G2, apart from the rules themselves of course, is the object. The object is the item over which the user has the largest control. When defining an object the user specifies the properties of the object: i.e., its attributes and creates a graphical representation of the object using an icon editor. The user also places the objects into a hierarchy of his own choice in which single inheritance of attributes applies. The objects themselves can be used to represent almost any concept of an application. They can represent the physical entities of an application, carrying attributes which match the properties of the real ones and having graphical representations which at least in some respect resembles the real objects.

On the other hand objects might be used to represent abstract entities having no graphical representation whatsoever.

The values of an attribute may be of the following kinds:

constants,

variables,

parameters,

lists,

other objects.

Constants may be numbers, symbolic values, logical values, or text strings. Unlike, for instance variables, constants can only receive new values by the user explicitly changing them in the attribute table of the object.

Variables may receive new values during run-time and are defined from four basic predefined classes:

quantitative variables, i.e., real-valued variables,

symbolic variables,

logical variables,

text variables.

With each variable comes an attribute subtable which contains a set default attributes, such as: the current value of the variable, the source of the variable, e.g., the inference engine, the simulator, or an external data server, the validity interval of the variable, e.g., how long the value should remain valid, etc. All things said about variables also apply to parameters with the difference being that parameters always have a current value.

Lists may contain arbitrary values belonging to a predefined class or objects defined by the user. Basic operations on lists are, as one might expect, the ability to insert or remove items on either side of the list and referring to items in ways such as:"**the second to last item in list-1**", etc.

An object also has the option of containing other objects as attributes. By doing so the object being an attribute can have no iconic representation. To give an example: having represented a car as an object with attributes such as colour, brand, etc., we might choose to represent the wheels as object-attributes of the car having their own attributes such as radius, etc.

Objects can be static, i.e., they are created explicitly by the developer,or transient, i.e., they are created (or deleted) dynamically during run-time.This in combination with G2 containing actions for moving, rotating, and changing the colour of an object renders the user rather great opportunities to create animations.

Connections

Connections form a way of representing relations between objects, perhaps mainly physical ones such as wires and pipes, but occasionally also abstract ones. Connections have graphical representations, attributes, are defined in a connection hierarchy similar to objects and may be defined as having a direction or not.

A connection may be attached anywhere onto an object or to a pre-defined port on the object which may have a port-name. Using connections we obtain a convenient way of referring to objects in rules and so on, for instance an expression such as: "**the water-tank connected to water-tank-1**". Unlike the case with objects there is no way in G2 to create connections during run-time. For this purpose there exists another predefined class: relations. Relations can only be created during run-time and may not have any attributes or hierachial structure. Since these are not used in the actual implementation I will not penetrate this further.

The rules

In G2 a rule is represented by a rule icon where the icon is the textual representation of the rule.

The rules of G2 are of five types:

Initial rules

If rules

When rules

Unconditional rules

Whenever rules

The rules of G2 refer to objects, connections, and their attributes in natural language style syntax and like other G2 items have attribute tables in which to specify properties such as: scan-time, search strategy, under which circumstances the rule should be evaluated (back or forward chaining), focal categories, focal classes, focal objects, etc.

"Initial rules" are, as one might guess, used to set up the state of the system and may look like: **initially conclude that every water-tank is empty**, where empty is a symbolic value of an attribute of the class water-tank.

"If rules" are fired through forward or backward chaining. As an example we might consider the following slightly naive G2-rule: **if weather is raining and any person is outdoors then conclude that the person is wet** (1), where "weather" is a symbolic variable able to obtain the values raining or not-raining and "person" is an object definition with variable attributes matching the values "outdoors" and "wet". Suppose that "weather" is raining and another rule is fired which concludes that **John is outdoors**, where "John is an instance of the class "person". In that case (1) will fire for the item"John" through forward chaining and the conclusion will be: **John is wet** i.e. the attribute of John which is able to have the value "wet" will receive this value.

Another possibility is that for some reason the expression **weather is raining** becomes true, in that case the inference engine will forward chain once again and scan through all members of the class "person" changing the adequate attribute of each one being "outdoors" to "wet".

Yet another possibility is that G2, evaluating another rule needs to find out whether a person is wet or not, thus activating (1) through backward chaining.

"When rules" may not be invoked through backward or forward chaining, instead they are invoked repeatedly through a scan interval specified by the developer. A typical "when rule" may look like: **when tank-1 is empty then inform the operator that "Tank-1 is empty!".** This rule will be evaluated in let us say 20 second intervals if so specified even if tank-1 should become empty" in between".

"Unconditional rules" are simply equivalent to "if rules" with the left hand condition always being true. Example: **unconditionally conclude that every tank is empty.** A rule of this format may only be invoked by a scan interval or by an invoke action in another rule, which is something we touch upon later.

"Whenever rules" rules allow asynchronous rule firing on events such that: a variable receives/fails to receive a value, or an object is moved by G2/the user. "Whenever rules" may not be fired through forward or backward chaining and of course not by a scan interval.

As we have seen there are several reasons for a rule to be fired, three of them are already familiar: forward and backward chaining and the fact that a rule is an initial rule.

There are however more opportunities in G2:

invoke: this command tells G2 to invoke all rules of a particular category. For instance: **invoke safety rules** tells G2 to invoke all rules that have safety as a focal category.

focus: this command tells G2 to focus on a particular object, referred to as the focal object. In order to do so G2 invokes all rules that have the object as a focal object or one of the objects superior classes as a focal class. Consider the following rules: **initially focus on tank-1** and **if any tank is empty then inform the operator that "A tank is empty"** with the latter having **tank** as a focal class of which **tank-1** is an instance. When G2 focuses on **tank-1** in invokes the second rule just for this tank and not for every tank in the application.

There are two further ways to invoke rules in G2: by "wakeup" and by activating an object, but since they are not used in the application, they will not be described here.

Procedures

The "tricky" part in programming using rules compared to conventional programming is to make sure that a given set of actions are performed in the right order when so needed.

For this purpose procedures have been introduced in the recent versions of G2. The procedures of G2 are rather Pascal-like in their programming style and are allowed to have attributes, just like any other G2 item. The allowed procedure statements include all the rule actions, assignment of values to local variables, **if-then-else** statements, **case** statements, **repeat** statements, **exit if** statements to exit loops, and **go to** statements. It is also possible to temporarily halt a procedure using a **wait** statement. A **wait** statement causes G2 to stop processing the procedure until either a specified time has elapsed or a condition is met. Procedures are executed by G2's procedure interpreter. The procedure interpreter cannot be interrupted by other G2 processing, i.e., the inference engine or the simulator.

Experiences of G2

Working with G2 has indeed been a both pleasant and instructive introduction to the concept of knowledge based programming. For my application G2 has proven to be an in every sense ideal tool. The more obvious advantages of G2 would include items such as:

Generic rules: Making it easy to add or subtract new items, when needed without having to change the background structure.

Connections: Giving the possibility to create a structure in which objects are referred to through physical and visible connections.

Graphic based development environment: Giving great possibilities to create, copy and move parts of the knowledge base through simple mouse interaction.

The user interface: Through the use of buttons and workspaces it is easy to design a suitable way for an intended user to communicate with the system.

From my point of view it proves to be rather difficult to find any real disadvantages of G2. But none the less a very mild form of criticism would be to say that G2, at least in the later versions, contains too much, i.e., to many ways of referring to objects through a variety of rule categories, making it virtually impossible to be well acquainted with every aspect of it. This sometimes makes it very tempting to start with the design of a system without first analysing the problem properly in order to find the simplest and perhaps most appropriate solution.

4

The structure of ÖSI and criteria for its conversion into an expert system

This chapter will give a description of the ÖSI-plans and in connection with that, demands and constraints that should be imposed upon the conversion of the plans into an expert system.

Description of ÖSI

ÖSI stands for (in Swedish): "De övergripande störningsinstruktionerna", which in english litterature normally translates into something like: "Emergency Operational Procedures (EOP)".These plans contain the instructions for how the operators at the nuclear power station should cope with the different sorts of disturbances that might occur.

ÖSI is divided into eight categories of disturbances:

REAKTIVITET
(reactivity)

ONORMAL VATTENNIVÅ I REAKTORTANKEN
(abnormal water level in the reactor-tank)

ONORMALT TRYCK I REAKTORTANKEN
(abnormal pressure in the reactor-tank)

ONORMALT TRYCK I REAKTORINNESLUTNINGEN
(abnormal pressure in the reactor-containment)

ONORMAL TEMPERATUR I
KONDENSATIONSBASSÄNGEN
(abnormal temperature in the condensation-pool)

ONORMALT LÅG VATTENNIVÅ I
KONDENSATIONSBASSÄNGEN
(abnormally low water-level in the condensation-pool)

ONORMALT HÖG VATTENNIVÅ I
KONDENSATIONSBASSÄNGEN
(abnormally high water-level in the condensation-pool)

DIFFUST LÄCKAGE
(diffuse leakage)

Each one of those is represented in the form of a flow chart, consisting of one up to three A4-pages (see appendix B).
The nodes upon the flow charts fall into three categories:

questions (labelled: KONTROLL) concerning the present state of the reactor environment, such as "Is the water level in the reactor tank sinking?" which are always to be answered simply by "yes" or "no" by the operators or

actions (labelled: ÅTGÄRD) to be performed by the operator such as: "Activate FILTRA !" or simply pieces of useful

information (labelled: INFO) such as: "The reactor is probably not shut down!".

To each of the eight ÖSI-plans there exists an entrance value, i.e, a condition under which the activities of the particular plan apply. For instance to plan nr. 4 (irregular temperature in the condensation pool) the condition is: "The temperature of the condensation pool is more than 28° C." In the case of some of the plans this gives the operator a possibility to enter the plan at a node different from the actual starting node, that is, if the operator already knows that the temperature of the condensation pool is more than, let us say 150° C, he does not have to answer a number of questions whose sole purpose would be to find out that this is true. This possibility does however not apply to all of the plans. The way in which the operator uses the ÖSI-plans is of course rather straight forward, answering the questions posed, performing the actions supplied while marking his route with a pen. To his aid there is to each page an opposite page providing information on parameters to that plan in the form of diagrams, curves, sketches, and so on.

Each of the nodes also contains information in coded form telling the operator where in the control room to retrieve the information required or where to perform the necessary action. Furthermore to each plan as well as to each node there exists, if so needed, additional more detailed explanatory information which is represented in separate documents and not on the plans themselves.

Demands on the implementation.

From our knowledge of the structure of the ÖSI-plans we can at this stage make some reflections on how the implementation should be done. As we have seen the plans simply pose questions and suggest actions in a totally well-defined and deterministic manner, i.e. it does not leave any of the reasoning to the operator. As a consequence of that, the implementation as an expert system needs not contain the knowledge, confined in the plans, as a de facto knowledge base and the rules will not have to perform any reasoning from the state of parameters of the real process. Rather the system should be based on rules responding to instructions from the operator, i.e., mainly based on forward chaining, displaying an animated graphical version of the paper version and providing the operator with requested information in an interactive, self-explaining, and user friendly manner.

The minimum set of demands that should be imposed on the implementation is, in my opinion, the following:

It should display a neat and fairly realistic version of the paper plans marking the question or action which is under consideration at the moment by changing the colour of the node and also marking nodes already dealt with, thus making the operator "feel at home" and making him able to follow the reasoning. When or if necessary the plan should be moved by the system since all of the plans are too large to be contained within the boundaries of the screen and having the text upon the nodes visible at the same time.

The system should provide the operator with a large version of the text of the node being considered at the moment and of course at the same time provide him with a means of answering the question or confirming that the action has been successfully performed.

The operator should have the option to obtain further explanatory information regarding the question or action displayed. The operator should also be able to get a computer version of the pages opposite to the pages containing the plans.

The system should contain a facility to keep history on each plan, enabling the operator to replay all the previous steps on each plan in the same animated manner as otherwise. While moving back or forth at his own choice he should be given information as to at what time these nodes were visited and what the answer was in case of a question, still having a large version of the text of the node displayed to him and still being able to retrieve further information.

When choosing a plan the operator should be provided with the entrance values of that particular plan and asked to confirm as to whether it applies or not. If there, as I have mentioned earlier exists a possibility to choose between different "levels of entrance", he should be able to do so.

And last, but by no means not least, since there is a distinct possibility that more than one plan applies at the same time, the operator should have the ability to switch between different plans in whatever manner he pleases and while his doing so the system should keep track for him, making sure that he returns at the same position where he left.

How these conditions are met and implemented into G2 will be examined in the next chapter.

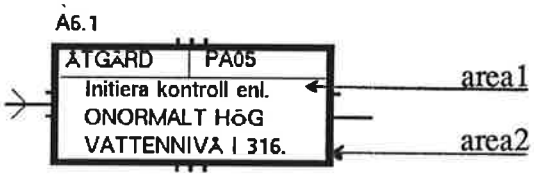
5

Description of the implementation of ÖSI

This chapter will the describe the over-all structure of the expert system and in particular closely examine the object definitions and rules used in the implementation.

Class definitions

The fundamental object of the system is an item which I, in a stroke of vivid imagination, chose to call a **box**. The icon of a **box** is a rather naturalistic copy of the real alternative and action boxes of the ÖSI-plans (**picture 5.1**).



picture 5.1

The icon consists of two layers: area1 and area2.

> Area1 has the ability to change from white to green and does so when the box is reached through a previous action or question thus indicating that the question or the action of the box is the one currently to be considered.
>
> Area2 changes from white to red indicating that the question of the box has been answered or that the action has been undertaken.

The text posing the question or describing the action is simply placed above the boxes in the form of G2 text boxes and has no real connection with the boxes (in the form of being an attribute or otherwise). Each box has a subworkspace which contains a large version of the text upon it. During run-time this subworkspace will be displayed at bottom centre of the screen, the instant its box is under consideration.

The attributes of a **box** are the following:

state: a symbolic variable with values active or non-active

process-state: a symbolic variable with values in-process or not-in-process

type: a symbolic variable with values alternative, action or end

The **state** attribute of a box being active indicates that the question or the action of the box is the one presently under consideration on that particular plan. Since the user can shift from one plan to another without finishing the latest one, due to circumstances, there might be several boxes active at the same time, at the most eight (the total number of plans).
A box that is not active is non-active.

The **process-state** attribute of a box being in-process indicates that the box is on the plan currently being worked upon, which means that all boxes on that plan is in-process and all other not-in-process.

An important observation to make is that, although there might be several boxes active and certainly are several boxes in-process at the same time, there is always exactly one both in-process and active.
This gives us a handy way of referring to this, the most important box, in rules etc. for instance through the expression: "**if any box is active and the box is in-process...**"

The **type** attribute indicates whether the box poses a question, suggests an action or is a terminal node of the plan.

Each box has six named out-ports, representing the real connections of the ÖSI-plans, two named: **yes-out**, two named: **no-out**, and two named: **uncond-out**.
To each of these there exists a corresponding in-port. The reason for having doublets of every port is just practical, making it easier to imitate the connections of the real plans.

Corresponding to the ports we have three connection classes: **yes-conn**, **no-conn** and **uncond-conn**.
These classes have no attributes and each one is made up of three black stripes, i.e., they are in fact identical. The reason for this is once again practical, giving the opportunity to use different colours during the construction of the plans to avoid mix-ups.

At this stage we can already guess that the basic rules for graphically moving through the "knowledge trees" should read something like: "**if answer is yes and any box B is active and B is in-process then conclude that the box connected at the yes-out of B is active...**" where answer is a symbolic variable which receives values "yes", "no" or, "check".
This is indeed quite simple and not very far from the truth.

The second object definition we describe is the **dot**.

Dots are used for moving the plans in order to keep the active box visible because, unfortunately, as every plan originally consists of up to three A4-pages it is not possible for the screen to contain the whole plan and keep the text readable at the same time. The mechanism for this will be described more thoroughly later. The icon of a dot is simple enough: it is represented merely by a geometrical point.

Every plan contains exactly one dot which makes a total of eight dots, labelled: **central-point-a, central-point-b, ..., central-point-h**.
Since the dots are the only objects that are unique on each plan and also carry names, it is quite convenient to use the dots for carrying and storing the necessary information of the whole plan through their attributes.
The attributes of a dot are the following:

process-state: a symbolic variable with values in-process or not-in-process

entrance-name: a symbolic constant

step-list: a list of boxes

time-list: a list of time-keepers

The **process-state** attribute indicates whether the dot is on the plan currently being worked with or not, in total analogy with the process-state attribute of a box.

The **entrance-name** attribute is simply the name of one of the entrance-boxes that applies to that particular plan (see the definition of an entrance-box). This attribute is necessary since the entrance-boxes are located on separate workspaces without names.

The **step-list** and **time-list** of each dot are G2-lists used for storing the history of each plan.

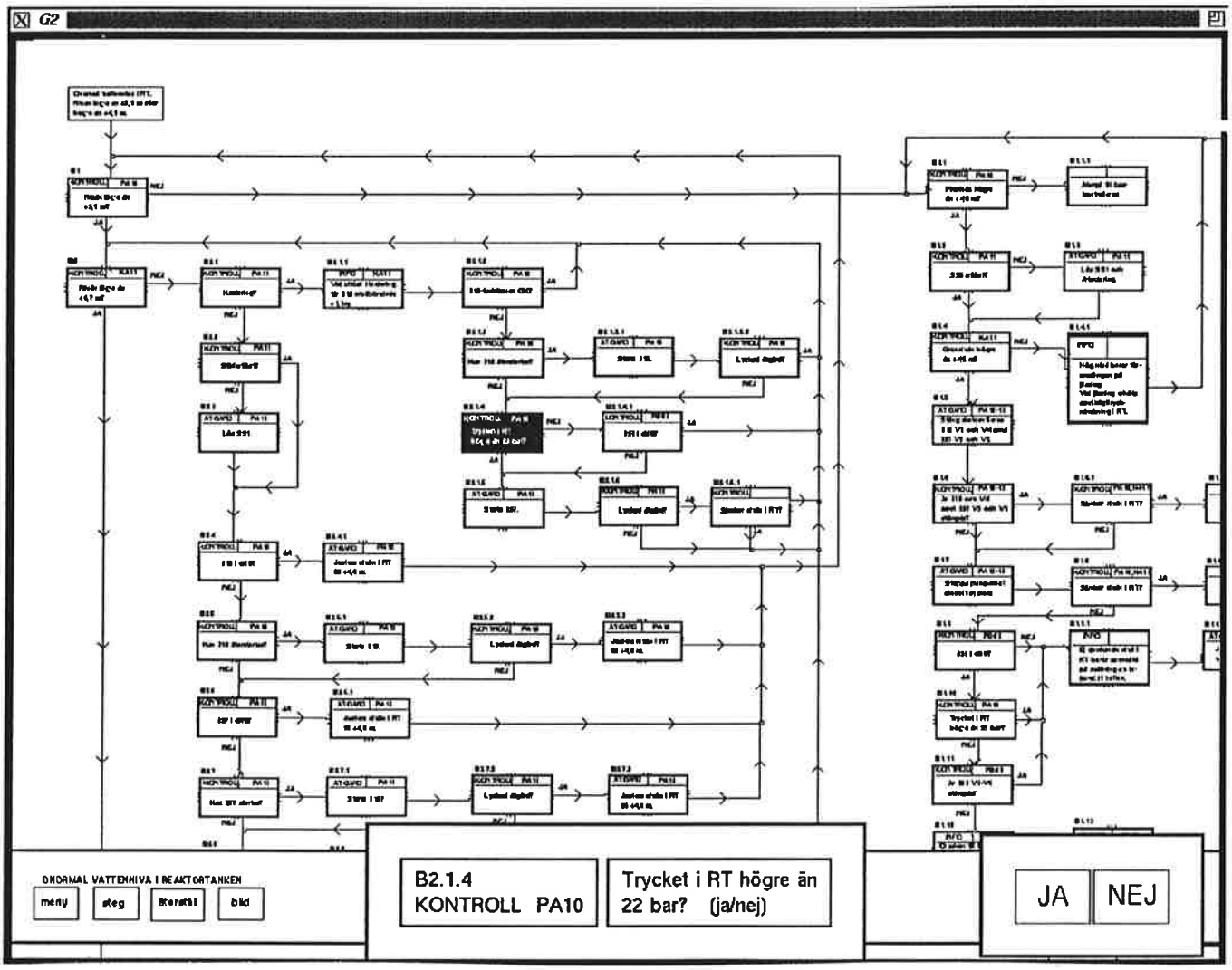
Each dot, just as every box has a subworkspace which in this case is designed like and serves as a "control board" for the plan. During run-time, this subworkspace will be displayed at the lower part of the screen whenever a particular plan is entered (see **picture 5.2**).
These boards each contain four buttons labelled:

meny (menu): displays a menu containing buttons for each of the plans enabling the operator to shift to a new one.

återställ (reset): resets the plan to same state it was before it was worked on, ridding it of all memory of previous steps.

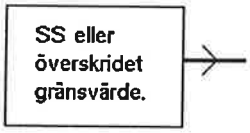
bild (picture): displays a computer version of the opposite page in the paper version of ÖSI.

steg (step): sets the system to a mode in which it is possible to reproduce the history of the plan.



picture 5.2

The third object definition we describe is the **entrance-box (picture 5.3).**



picture 5.3

These items are iconical representations of the different levels of entrance mentioned in chapter 4. When choosing a plan which has not been entered before, the system will display a workspace containing entrance-boxes labelled by the entrance values that apply to this plan and by clicking on an entrance-box the operator will be able to choose where to start.

Each instance of this class only has one attribute: **box-name**, which is the name of the box it refers to (only boxes that correspond to a level of entrance have names). To this class I have defined a user-menu-choice which is a facility in G2 that allows the user to define an action that should be performed on the object when clicking on it (the object). In our case this action reads: **conclude that the box named by the box-name of the item is active and hide the workspace of the item**, where **item** refers to the entrance-box.

The fourth object definition is an item called: **add-inf** (additional information). The icon is, as in the case of a **dot** merely a geometrical point. Members of this class carry no attributes, instead their sole purpose is to hold a subworkspace containing the explanatory text associated with every question/action. An instance of this class is placed on the subworkspace of each box. When the operator clicks on the text of the question/action proposed to him the subworkspace of the **add-inf** containing the explanatory text is displayed. This is again achieved through a **user-menu-choise** for **free-text.** The action performed when clicking on a **free-text** is the following: **show the subworkspace of the add-inf nearest to the item.** This facility is only available for a small fraction of the questions/actions since implementing it fully would be more of a job for a full time secretary.

The fifth and last class definition described is the **time-keeper**. This class has no iconic representation and no permanent instances. It carries three attributes: **proc-hour**, **proc-minute**, and **proc-answer**. During run-time instances of this class are created transiently and inserted into the **time-list** of the **dot** of the of each plan. Through actions of the rules base their attributes are supplied with information on when each box was dealt with and what the answer was.

These five object definitions are really the only ones in the system apart from two or three of which the icons are merely used as pictures holding no attributes. Neglecting these, we will instead describe a number of additional item definitions necessary to understand the next part of this documentation.

Answer is a symbolical variable which is used to retrieve the response from the operator on the questions posed by the system. **Answer** gets the symbolic values: **yes**, **no**, or **check** where **check** confirms that the operator has undertaken the action proposed by system. The operator makes his response by clicking on buttons labelled "JA", "NEJ", or "OK" located at two different workspaces named: **option1** and **option2 (picture 5.2)**, which pop up during run-time. Through the rule base the system guarantees that in the case of an alternative box, the operator is given only a yes/no option and in the case of an action box only the check option.

Show-step is a logical variable, being false indicating that the system is in the normal operative mode in which it responds to the answers and actions of the operator as described earlier. The situation of **show-step** being true represents a functional mode in which the previous historical events of plan may be reproduced by the operator. The variable is used in the left hand side as a condition separating the rules that apply to each of these two modes. The variable is initially false and becomes true only by the operator pressing the **steg**-button.

The next item is a list named **track** which due to actions of the rule base always contains the box, considered by the operator at the moment, as its sole element. This may seem strange, considering what has been said earlier, since the proper way to address this box should be through its unique property of being both **active** and **in-process**. This is still true for most of the rules but in the case of a rule with a left hand side like: **if answer is true and any box B is active and B is in-process...** , which normally will be triggered by **answer** obtaining a value and not by a box becoming either **active** or **in-process**, this would be rather slow since G2 would scan this rule for each and every instance of the class **box**. Therefore **track** in some cases gives a fast and convenient way of reference.

Before we drown in any more definitions I would like to summarise what we already know by presenting the rules I consider being the fundamental ones, describe them and explain in what order and for what reasons they are invoked and thereby, hopefully, give an insight as to how the system works in the operative mode.

The basic rules

The rules we will consider, in this section, are the following:

(1)

if any box B is active and B is in-process then
in order
remove the first box from track and
insert B at the beginning of the box list track

The first of these rules: (1), ensures us that the box in **track** always is the box currently being considered. This rule could be fired in two ways, either by a box that is in-process becoming active by the operator responding to a question/action or by the operator switching to a plan where there is an active box thus making it in-process.

(2)

if any box B is active and B is in-process then
invoke moving rules

Rule (2) fires as the same time as the previous rule and invokes a rule category **moving rules.** These rules decide whether or not the plan has to be moved in order to keep the current box visible and in that case how much (**moving rules** are described in the next section).

(3)

if any box B is active then
change the area1 icon-color of B to green

(4)

if any box B is non-active then
change the area1 icon-color of B to white

These two rules (3) and (4) are simple enough: when the operator has responded to a question/action from the system, the colour of the new box that comes into consideration changes to green thus marking it, while the previous box changes back to its original white colour.

Note that these two rules for marking boxes apply in both modes.

(5)

if any box B is active and show-step is false then
insert B at end of the box list the step-list of the dot
upon the workspace of B

Rule (5) is used for keeping history and it simply states that if a box becomes active in the operative mode (**show-step** is false), i.e., as a consequence of rule (7), this box is saved in the **step-list** of the dot on that particular plan (**the dot upon the workspace of B**) to be used in the previous step mode.

(6)

if any box B is active and show-step is false then
in order
create a time-keeper TK and
insert TK at the end of the item list the time-list of
the dot upon the workspace of B

Rule (6) fires at the same time and for the same reason as the previous rule. It creates a transient **time-keeper** and adds this to the **time-list** of the dot on the plan.

This transient **time-keeper** is used for preserving the time the box was dealt with and the answer. This might seem a bit strange since one might think that we could use the boxes in **step-list** for this purpose, provided they be supplied with the proper attributes. We can not however and the reason for this is that the same box can appear at several places in the step-lists and assigning a **proc-hour, proc-minute**, or **proc-answer** to one of these would change these attributes of the box at other places in the list.

(7)

if answer is yes then
in order
conclude that answer has no value and
conclude that the box in track is non-active and
change the area2 icon-color of the box in track to red and
conclude that the proc-answer of the last time-keeper in
the time-list of the dot upon the workspace of the first
box in track = "JA" and
conclude that the proc-hour of the last time-keeper in the
time-list of the dot upon the workspace of the box in
track = the current hour and
conclude that the proc-min of the last time-keeper in the
time-list of the dot upon the workspace of the box in
track = the current minute and
conclude that the box connected at the yes-out of the box
in track is active

Rule (7) is the response from the system to a yes-answer. It exists in two more almost identical versions corresponding to the no- or check-answers.

If we examine the actions one by one, what it does is the following: It concludes that **answer** is no longer **yes** which is crucial because otherwise this rule would not respond to a second yes-answer from the operator since **answer** would already be **yes**. It changes the **area2** colour of the current box to red, thus marking it as having been "visited" and concludes that it is no longer active. It saves the answer of the operator and the time in the last **time-keeper** in the **time-list** of the dot upon that plan. Finally it concludes that the box at the **yes-out** of the box in **track** is now active.

(8)

whenever the process-state of any dot D receives a value and when the process-state is in-process and not(there exists a box B such that B is active and B is in-process) then
show the workspace of the entrance-box named by the entrance-name of D with its bottom center at the bottom center of the screen

Rule (8) might be somewhat difficult to interpret, but in short what it says is the following: if the operator has chosen a plan (**any dot D is in-process**) and there does not exist a box both active and in-process, i.e., the plan has not been worked on before, then G2 should display the workspace containing the corresponding entrance-values for that plan (**the workspace of the entrance-box named by the entrance-name of D**).

(9)

if any box B is active and B is in-process and B is alternative and show-step is false then
show option1

Rule (9) fires when the operator has responded to a question/action and thereby made a box active or when he has switched to a plan on which there exists an active box, thereby making it in-process. The action of the rule is to show a workspace named **option1** which contains two buttons labelled **JA** and **NEJ** of which the action is to conclude that **answer** is yes respectively no.

This is in the case of the box being **alternative** which is a left hand side condition. There two more similar rules for the case of the box being **action** or **end**. The first of these has: **show option2**, as the right hand side, where this workspace of course contains a button for concluding that **answer** is **check**. In the case of the box being **end** the response of the system is to hide both **option1** and **option2** and to display a message telling the operator to choose a new plan.

(10)

for any box B if there exists a box connected at the yes-out of B then
conclude that B is alternative

Rule (10) concludes that if a box has got another box connected at the **yes-out** of it then the type of the box is **alternative**. There are to more versions of this rule, one stating that if a box has another box connected at its **uncond-out** then its type is of **action**, and one stating that if a box has no other box connected at its out-ports then it is of type **end**. These three rules are invoked through backward chaining when other rules refer to the type of a box in their left hand conditions, such as the previous rule (9). As a matter of fact this is the only situation throughout the whole system where backward chaining actually occurs, which is quite in line with what has been pointed out earlier about the system being mainly data driven, responding to the actions of the operator through forward chaining.

For the simplicity of the run time example that will follow we may consider one more rule which in fact does not exist:

(11)

if any box B is active and B is in-process then
show the subworkspace of B at the bottom center of the screen

This rule (11) is quite simple. It merely states that when a box comes under consideration either as a consequence of rule (7) or by the operator switching plans G2 should display the subworkspace of this box at the bottom of the screen i.e. the large version of the question/action. The reason that this rule does not exist is, although it might seem logical enough, is that it would display the subworkspace at random time and in the case of the plan having to be moved it might even end up behind the plan. Because of this, I have found it easier to have this rule embedded as actions in rules and procedures that move the plan.

Having these eleven rules in mind it should be quite easy to understand the basics of how the system works in a short run time example.

We consider a situation where the operator is working on a plan and decides to shift to a plan which has not been previously worked on. To shift plans the operator first presses the menu button on the control board of the plan he is currently working with. The action performed by this button is simply: **show menu with its bottom center at the center of the screen.** Located on this workspace, now displayed, are eight buttons each one referring to one of the eight plans of ÖSI.

Let us assume the operator chooses the first plan (reactivity). The action performed when pressing this button is:

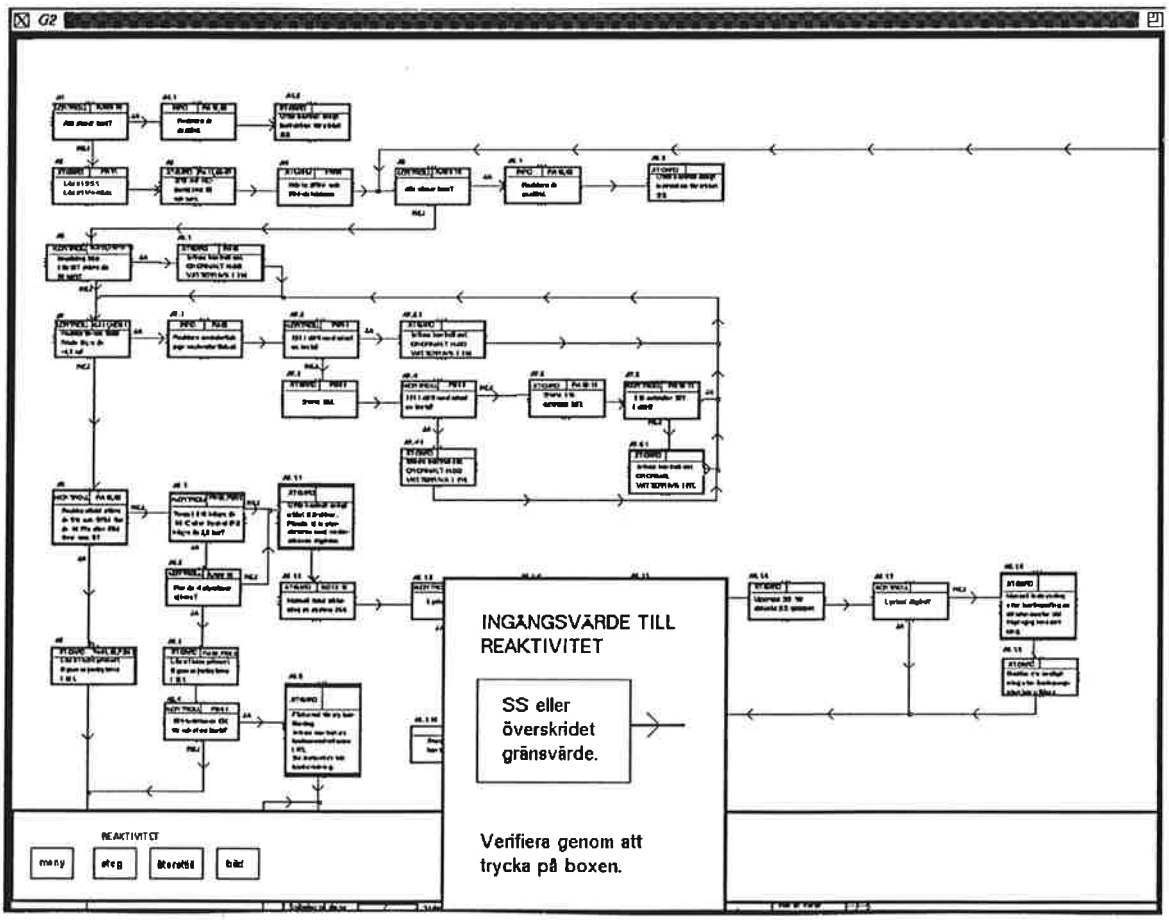
in order
conclude that every box is not-in-process and
conclude that every dot is not-in-process and
show central-point-a 100 units above the center of the screen and
conclude that every box upon the workspace of central-point-a is in-process and
conclude that central-point-a is in-process and
show the subworkspace of central-point-a

The conclude actions of this button assures us that the boxes and the dot of the first plan are **in-process** and that none other are.

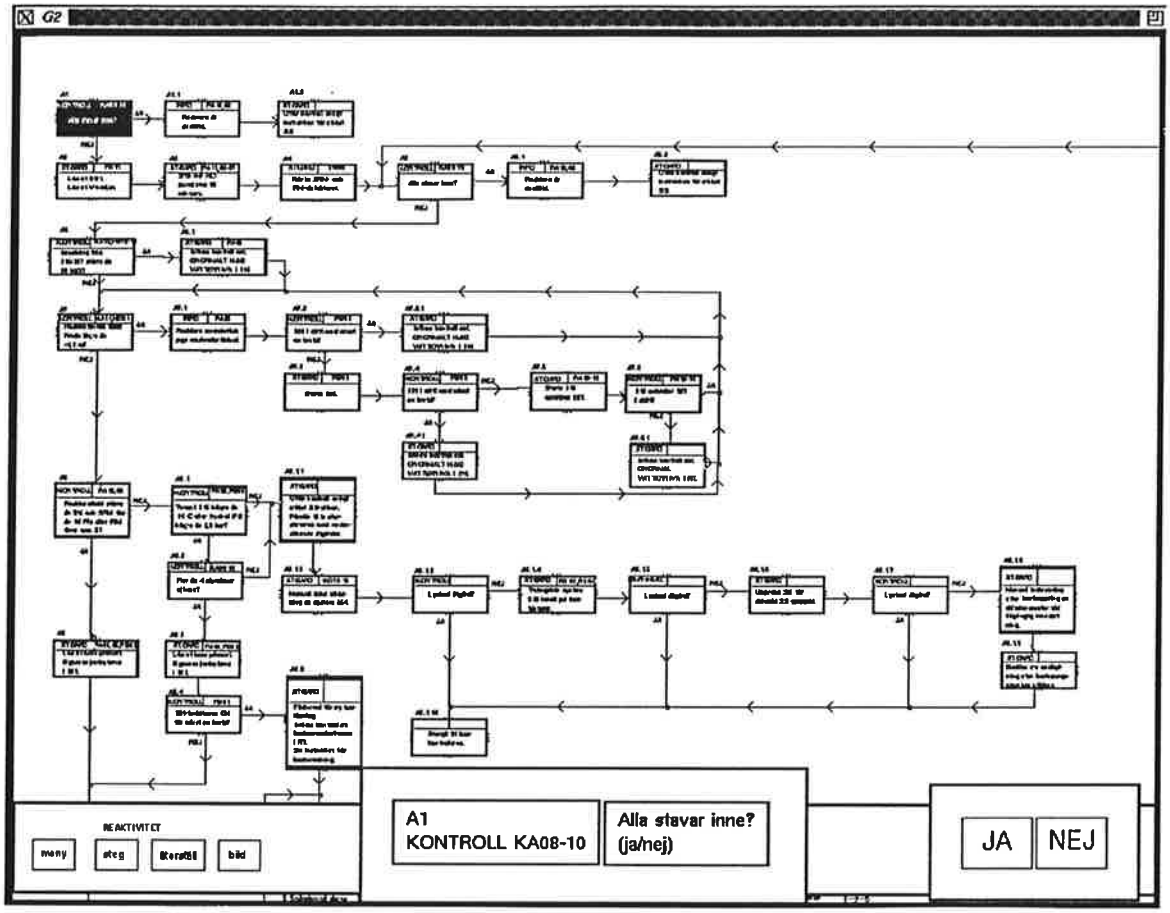
The button also displays the plan (**show central-point-a**) and thereafter displays the control board of the plan (**...the subworkspace of central-point-a**).The actions of the seven other buttons are identical except that they of course refer to other **dots** (**central-point-b** and so on). The fact that **central-point-a** becomes **in-process** and that there are no boxes both **active** and **in-process** causes rule (8) to fire, displaying the workspace of the **entrance-box** named by **central-point-a** which is this case contains only one level of entrance (**picture 5.4**). When clicking on this entrance-box the following action is performed, as we have seen earlier: **conclude that the box named by the box-name of the item is active and hide the workspace of the item.** Or to put it in other words one, previously determined box on the plan, becomes **active** and the workspace of the **entrance-box** disappears. The fact that a box which is already **in-process** becomes **active** causes the following things to happen. Rule (1) fires placing the box as the single element in **track**. Rule (2) fires and invokes the rule category **moving rules**. Rule (3) fires changing the colour of the box to green. Rules (5) and (6) fire, adding the box to the **step-list** of **central-point-a** and creating a transient **time-keeper** which is placed in the **time-list** of the same dot for future use. If we assume that the active box is of type **alternative**, rule (9) will also fire, backward chaining to rule (10), and after that displaying the workspace named **option1** thus presenting the operator with a yes/no alternative. Finally the non-existing rule (11) will fire, displaying the question of the box (**picture 5.5**). At this stage the operator will preferably try to answer the question by pressing either the **JA** or **NEJ** button. If we assume that the answer is **yes** rule (7) will fire.

Through this, the box will become **non-active** while its **area2** icon-colour changes to red thereby marking it as a member of the group of boxes already having been dealt with at least once, the answer and the time of the answer will be stored in the **step-** and **time-lists** of **central-point-a**. Finally a new box located at the **yes-out** of the previous box will become active. The previous box becoming non-active will cause rule (4) to fire, changing its colour back to white. Apart from this, the fact that a new box has become active will start the chain of rule firing all over again.

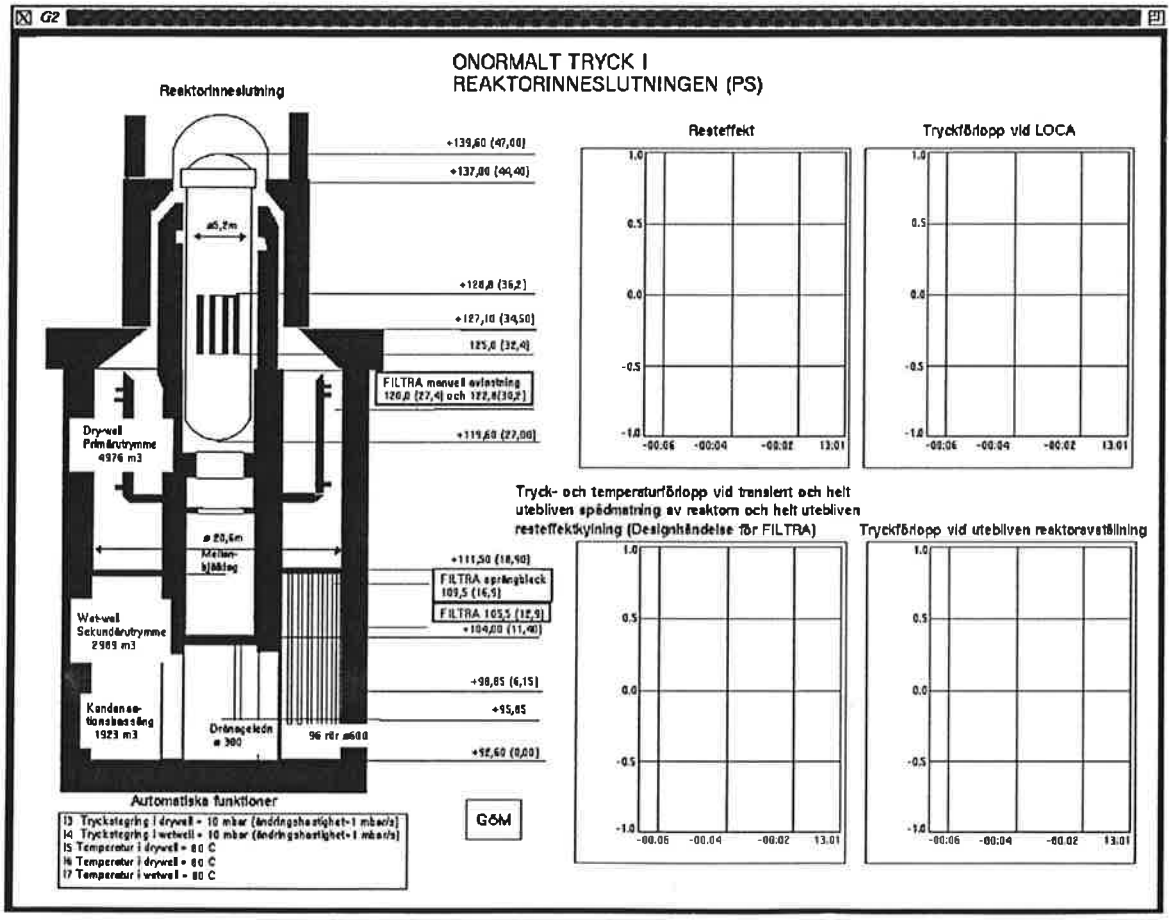
Let us now assume that the operator decides to change back to the plan he was working with earlier. When pressing the button for that plan on the menu, the operator will cause all boxes on that plan to become **in-process**. Since there exists a box that is **active** on this plan this box will now be placed in **track** while all the relevant rules will fire presenting him the with the question/action for this box and with an option to respond to it. Apart from the capability to shift plans and responding to question/actions presented to him the operator, all the time, while working with a plan, has four more options: to press the **återställ**-button which resets the plan by concluding that all boxes on it are non-active and emptying the history keeping lists of the plan, to press the button labelled **bild** which shows the opposite page of the paper version (**picture 5.6**), to press the **steg** button which enables the operator to replay the history of the plan or to click on the text of the question/action to retrieve more information if necessary (**picture 5.7**).



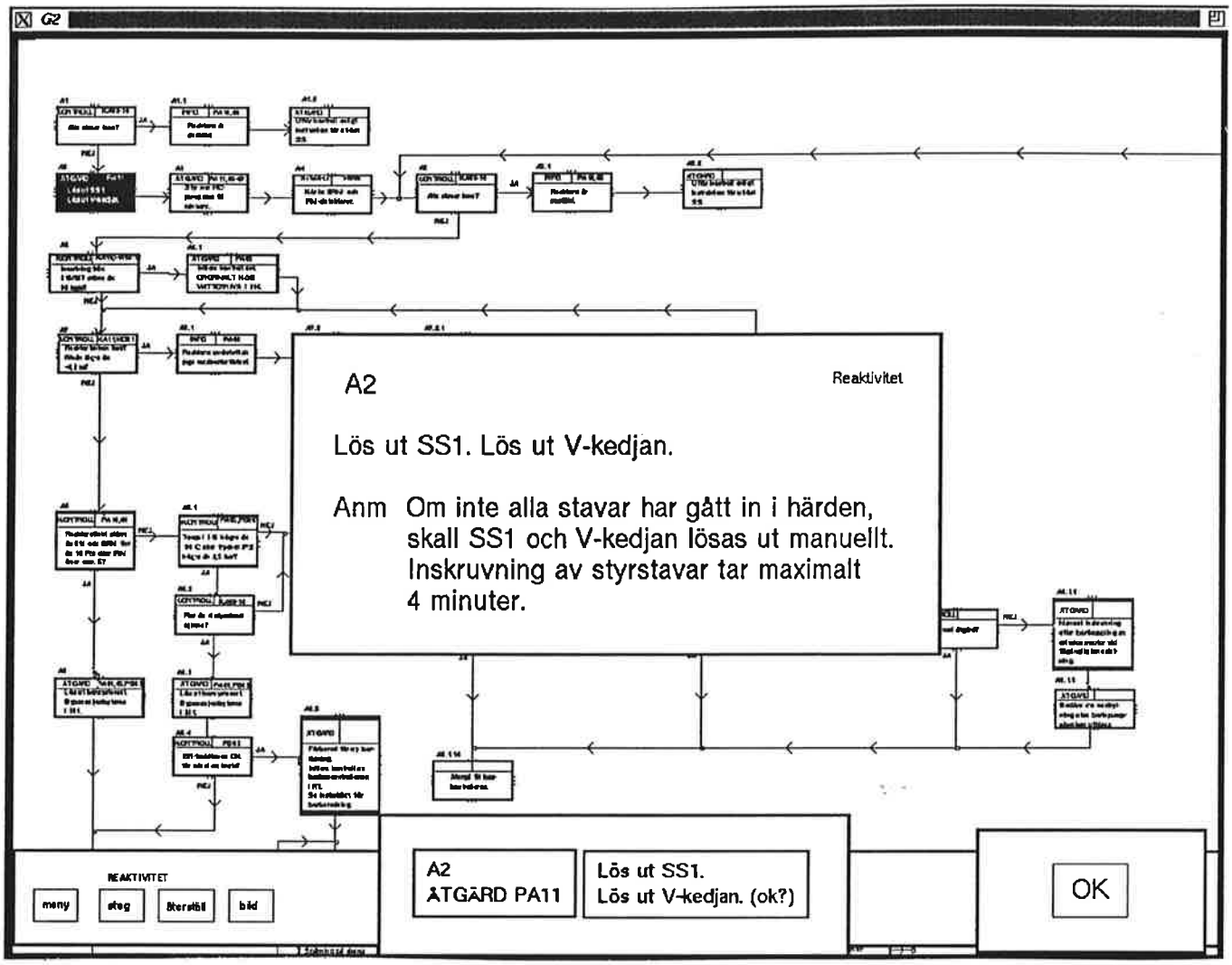
picture 5.4



picture 5.5



picture 5.6


G2
A2
Reaktivitet
Lös ut SS1. Lös ut V-kedjan.
Anm Om inte alla stavar har gått in i härden,
skall SS1 och V-kedjan lösas ut manuellt.
Inskruvning av styrstavar tar maximalt
4 minuter.
REAKTIVITET
meny
steg
återställ
bild
A2
ÅTGÄRD PA11
Lös ut SS1.
Lös ut V-kedjan. (ok?)
OK

picture 5.7

Moving the plans

As I have pointed out several times earlier, one problem of the implementation is the fact that the plans are simply too large to be displayed within the borders of the screen. There two ways of solving this. One is to change pages as in the paper version and the other is to move the plans when the box currently under consideration disappears out of sight thus making it reappear. In the implementation I have chosen the latter possibility. The reason for this is that this solution makes it easier for the operator to orientate himself since he will all the time be able to see the neighbouring boxes that have been previously visited, being marked by their colour. The mechanism for moving a plan is indeed quite simple: we continuously keep track of the **x** and **y** distances between the **dot** (central point) of the plan and the box currently under consideration. When either of these becomes larger than a couple of pre-defined values we move the dot, in some pre-defined manner, through a **move** action and after that through a **show** action, we make the **dot** reappear at a given position of the screen. This will give the impression of the plan having moved. However the plan will now cover all other workspaces so we must keep in mind to make these reappear in a correct order.

To implement this idea I use the following items:

x-dist and **y-dist:** quantitative variables for storing the coordinate distances mentioned above,

move-plan: a g2-procedure that handles the actual moving and reappearing of other workspaces (see appendix A for a full description),

and three rules:

if any dot D is in-process then
conclude that x-dist = (the icon-x-position of the box in track - the icon-x-position of D) and
conclude that y-dist = (the icon-y-position of the box in track - the icon-y-position of D)

whenever x-dist receives a value and when max(abs(x-dist/1500), abs(y-dist/1100)) < 1 then
show the subworkspace of the box in track with its bottom center at the bottom center of the screen

whenever x-dist receives a value and
when any dot D is in-process
and max(abs(x-dist/1500),abs(y-dist/1100)) >=1 then
start move-plan (x-dist, y-dist, D, show-step).

The first one of these is of category **moving** and is invoked by rule (2) whenever a new box comes under consideration. The reason for invoking this rule rather than constructing it as a rule that would forward chain itself when a new box becomes active is that we need to be certain that rule (1) has completed otherwise the reference **the box in track** might not be updated. The action of the rule is to provide **x-dist** and **y-dist** with their proper values.

The second rule states that if neither **x-dist** is larger than 1500 units nor **y-dist** is larger than 1100 units, we need not move the plan and the only action is to display the subworkspace containing the large version of the current question/action. It would have been nicer to be able to construct this rule with a left hand side simply as: **if max(abs(x-dist/1500 ... < 1,** rather than as a **whenever rule** but unfortunately G2 does not forward chain to expressions of this complexity. The exact numbers: 1500 and 1100, used as constraints are simply found by trial and error and possess no deeper meaning.

The last one of the rules handles the case where either **x-dist** or **y-dist** exceed their boundary values (1500,1100), i.e., the plan has to be moved and does so by starting **move-plan** (see **appendix A**). This procedure contains actions that move the **central-point** of the plan by distances given through formulas I have found appropriate (once again by trial and error and after that shows the **central-point** 100 units above the center of the screen. Having done this, **move-plan** uses the rest of its arguments to decide which workspaces that need to be shown again and does so.

Reproducing history of plans

While working on a particular plan the operator, all of the time, has a possibility to shift from the normal mode in which he responses to the propositions of the system to a mode in which the earlier steps and actions are replayed to him. While doing this the operator is still presented with the plan in the same animated fashion as otherwise and still presented with the large version of the text of the box he is examining at the moment. The main difference is of course that he is no longer able to respond to these. Instead through the clicking on buttons he moves back or forward through the previous route of the plan.

In the history replaying mode most of the rules described earlier still apply. The exceptions are (5), (6), and (9) which have **show-step** being false as a left hand condition. This means that when in this mode, for what should be obvious reasons, we no longer store the boxes in lists when becoming active and that the operator is not presented with the possibility to respond. To differ further between the two modes the colour of the control board and the questions are altered.

The items and rules designed to apply only in the historical mode are listed and described below.

duplicate-list: a G2 procedure that produces a transient copy of the list given as argument (see **appendix A**).

empty-list: a standard procedure for emptying the list given as an argument.

duplist-a and **duplist-b**: G2-lists containing duplicates of the **time-** and **step-list of** the **central-point** of the plan.

list-number: a quantitative variable that keeps track of the position while moving back and forth in the lists above.

step-mess: a message definition. An instance of this class, named **step-inf**, is placed upon the control board and used for displaying the time and answers mentioned above.

(15)

if show-step is true and any dot D is in-process then
in order
start duplicate-list (the step-list of D, duplist-a) and
start duplicate-list (the time-list of D, duplist-b) and
conclude that the first box in the step-list of D is active
and show D 100 units above the center of the screen
and show prev-step-area and
conclude that list-number = 1

This rule (15), which is used for initiating the history showing mode is of category **step** and not invocable through forward chaining.

(16)

if any dot D is in-process then
in order
conclude that the first box in duplist-a is non-active and
start empty-list (duplist-a) and
start empty-list (duplist-b) and
show the subworkspace of D and
conclude that show-step is false and
conclude that the last box in step-list of D is active and
remove the last box from the step-list of D and
remove the last time-keeper from the time-list of D

This rule (16) of category **get-back** is used for exiting from the historical mode. Nor this one is invocable through forward chaining.

(17)

unconditionally
in order
insert the first time-keeper in duplist-b at the end of the
time-keeper list duplist-b and
remove the first time-keeper from duplist-b and
insert the first box in duplist-a at the end of the
box list duplist-a and
remove the first box from duplist-a and
conclude that the last box in duplist-a is non-active and
conclude that the first box in duplist-a is active

This rule updates **duplist-a** and **duplist-b,** i.e., it single steps forward through the copies of the lists keeping the history. This rule is of category **forward** and has a corresponding rule of category **backward** which performs the opposite, i.e., single steps backwards.

(18)

if any box B is active and B is alternative and show-step
then
change the text of step-inf to: 'Svaret var: (the
proc-answer of the first time-keeper in duplist-b),
klockan var (the proc-hour of the first time-keeper in
duplist-b):(the proc-minute of the first time-keeper in
duplist-b)'

This final rule has no category and is used for changing the text of the message telling the operator the time and the answer. It exists in two more versions for the case of the box being **action** or **end.**

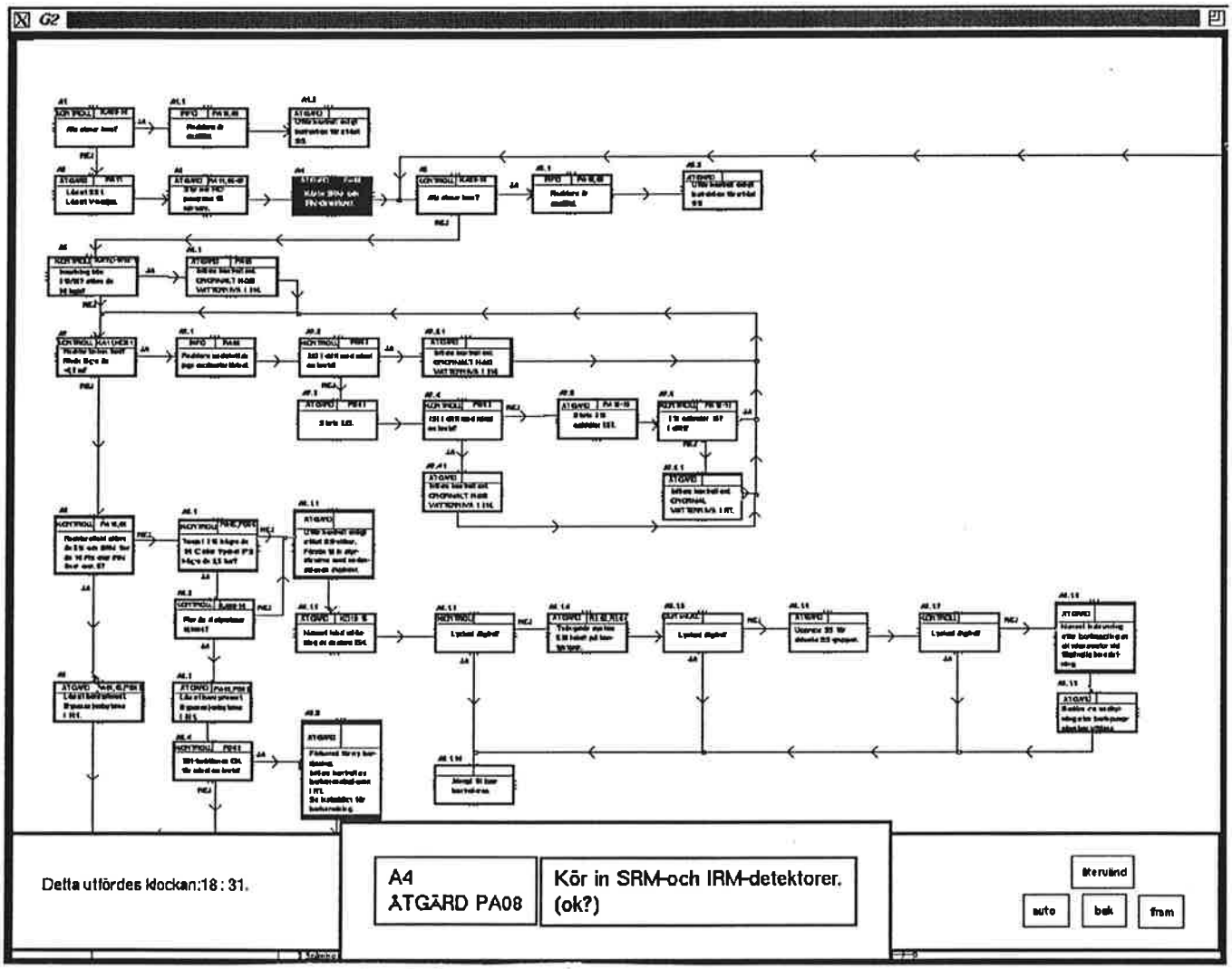
Unlike in the operative mode where the subworkspaces of the central points serve as control boards, one for each plan, in the history mode we have one single control board for all plans. This board (see picture **5.8**) contains the message described above and four buttons for operating. These are labelled:

fram (forward): activates the next box in the replay, i.e. takes a step forward.

bak (backward): one step backwards.

auto: single steps forward automatically in three second intervals.

återvänd (return): returns the system to its operative mode.



picture 5.8

To explain how these rules cooperate to perform their task I will once again do so by giving a run time example. We consider the situation where the operator is working with a plan (let us assume REAKTIVITET) and decides to shift to the history replaying mode. In order to do so he first of all clicks on the **steg** button on the control board. The action performed when doing so is:

when the number of elements in the step-list of
central-point-a > 1 then
conclude that show-step is true and
conclude that the box in track is non-active and
invoke step rules

The condition, the number of elements in the list exceeding one, assures us that the plan has indeed been worked on. The actions after that conclude that **show-step** is true. The first box in track is no longer active since this is normally not the first of the boxes performed earlier and we invoke **step rules** (rule (15)). Note that this rule is not forward chained to when **show-step** becomes true since this variable's ability to do so has been turned off to avoid undesired forward chaining to other rules with this variable in their left hand condition.

When rule (15) is invoked its actions are to make copies of the history keeping lists of the **central-point-a**, conclude that the first box in the step-list of **central-point-a** is active since this is the first box of the plan that was dealt with, display the control board (**prev-step-area**), and to set **list-number** to one since we are at the beginning of the lists. As a consequence of a box becoming active several of the rules described earlier will fire, placing the box in **track**, changing its colour, invoke **moving** rules, displaying its text and so on.

Rule (18) firing as a consequence of the first box in the **step-list** of the plan becoming active, will in the case of the box being **alternative**, yield a message displaying the time and answer of the box at the left of the control board. At this stage the operator will probably press the **fram** button of which the action is:

when list-number < the number of elements in duplist-a
then in order
conclude that list-number = list-number + 1 and
invoke forward rules

This means that as long as we have not reached the end of the previous steps we update **list-number** and invoke **forward** rules. The action of the **forward** rule (17) is to update **duplist-a** (which contains a copy of the **step-list)** and **duplist-b** (the **time-list**) and to conclude that the box currently active is non-active and that the box after this one is active. At each of the steps backwards or forwards rules (18) or a version of it is invoked through forward chaining. The **bak** button and backward rules represent merely the opposite of what has just been described and are quite similar (see appendix).When the operator chooses to re-enter the operative mode he presses the **återvänd** button the action of this is to hide the **prev-step-area** and to invoke **get-back** rules.

This rule (16) concludes that the first box in **duplist-a** is no longer active, i.e., the last box that was examined in the historical mode. It empties **duplist-a** and **duplist-b** by calling the procedure **empty-list**. It displays the control board for the operative mode (the subworkspace of D) and sets **show-step** to false. It concludes that the last box in the **step-list** of **central-point-a** is active which is the last box that was active in the operational mode and finally it removes the last items of the two lists of **central-point-a** which is necessary because otherwise these would appear twice in a row since rule (5) and (6) have fired once again as a result of the last box in **step-list** becoming active while **show-step** is now false again. Apart from this other necessary rules have fired making the screen the same as when we left the operative mode.

Miscellany

Apart from the rules described in the previous sections and their corresponding versions, the rule base indeed contains a very small number of additional ones. The major part of these are of course **initial** ones and are all covered in appendix A and need no further explanation. The only ones I would like to comment on are the following:

if any box B is active and not(show-step) then
change the background-color of every free-text
upon the subworkspace of B to grey and
change the border-color of every free-text
upon the subworkspace of B to yellow

This rule exists in a corresponding form for the case of **show-step** being true in which two colours are black and wheat instead of grey and yellow. These two rules assign different colours to the text upon the large version of the questions/actions making it easier for the operator to differ between the operative and historical mode.

if any box B is non-active then
hide the subworkspace of B

if any dot D is not-in-process then
hide the subworkspace of D and
hide the workspace of D

These rules serve as the "cleaning-ladies" of the system discarding workspaces as they come out of consideration.

One further facility, which has been introduced through a **user-menu-choice** is the possibility for the operator to retrieve all available information of each box simply by clicking on it. This works even if the box is not active at the time and the result is the same as clicking on the text upon the subworkspace of the box.

Summary

The major ambitions when creating the rule base, apart from obvious ones such as giving the system a correct behaviour, have been mainly two.

The first one is to make the system sufficiently fast in spite of the fact that it contains a large number of objects to be manipulated with. This is obtained mainly by avoiding, throughout the rule base, left hand conditions that would have to go through every instance of the class **box** and when so needed instead use the reference **track**.

The second aim has been to avoid referring specifically to any plan or any named item upon them, i.e., to make the rule base independent of the plans themselves. This makes it quite easy for any presumed user to introduce additional plans without having to make any changes whatsoever. To do so he simply designs a new plan, places a named **dot** on it, creates a control board and workspace containing the **entrance-boxes,** preferably using the cloning and operate on area facilities and, finally places a **button** referring to the **dot** of the plan on the menu.

A further interesting remark would be the fact that the rule base could relatively easy (at least in principle) be modified for on-line purposes. To do this, one would need to associate with every box an activatable workspace.
This workspace would in the case of an alternative contain rules of the following format: **initially if <condition> then conclude that answer is yes and <further actions>**, where the parameters in the <condition> part would receive values from external data servers. In the case of actions to be performed the workspace would contain a procedure capable of handling these. The rôle for the operator in this version would be to answer questions that are too "fuzzy" for the system to evaluate and to confirm each measure performed by the system.

6

Concluding remarks

The main purpose if this masters thesis has been to implement the ÖSI-plans as an expert system, using G2 as a shell. In order to achieve this, the approach has been to fully preserve and benefit by the flow chart structure already present in the original version; making use of the excellent facilities in G2 for constructing a system of this kind, representing the nodes by identical G2-objects carrying suitable attributes and the transition between nodes by physical G2-connections. Hereby the knowledge base of the system will lie within the structure itself, rather than being represented separately in the form of facts about the real process.

In this, indeed quite simple approach, the task of the rule base is to administer the graphical representation and the communication with the user.

The more intuitive and traditional expert system approach in which the system, during a consultation phase, retrieves information from the user, makes the proper inferences and then provides the user with a list of accurate measures to perform would be virtually impossible to implement. The reason for this is once again the specific structure of ÖSI in which the questions posed often refer to the result of a previously performed action. This, in connection with the lack of any global knowledge that would apply at any time or to any situation, would not make it possible to break the system down to any rule base of a reasonable size. Instead it would certainly require at least one rule for every node, making it impossible to guarantee a correct behaviour in every situation. As one further complication with this method we could add the fact that G2 being constructed for real-time applications has no facility for stopping the backward chaining and prompting the user for the value of a variable or parameter.

If we choose to add up the reasoning of the former lines the conclusion would be that a straight-forward implementation along the lines of the constructed one, concentrating mainly on the graphical representation of the system and creating a user environment containing all necessary information in a compact and easy-retrievable form, proves to be the simplest, safest and perhaps only reasonable solution. The expert system shell G2, with its object-orientation and graphic based environment containing: windows and mouse-interaction, has proven to be an nearly ideal tool for this application.

References

Harmon and King. (1985): Expert Systems, John Wiley & Sons.

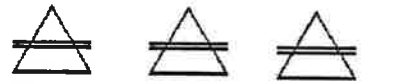
Knowledge-Based Real-Time Control Systems: IT4 Project: Phase 1 (1990): Department of Automatic Control, Lund Institute of Technology, Lund, Sweden.

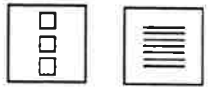
Moore, R.L, H. Rosenof, and G. Stanley (1990): "Process control using a real time expert system," Proc. 11th IFAC World Congress, Vol 7, Tallinn, Estonia, pp. 234-239.

Appendix A

The Rule Base

basic definitions





initially rules

initially conclude that every dot is not-in-process

initially conclude that every box is non-active

initially conclude that show-step is false

initially for any add-inf unconditionally change the background-color of the subworkspace of the add-inf to wheat

initially insert a-box at the beginning of the box list track

rules for checking the connections between boxes

for any box B if there exists a box connected at the yes-out of B and not (there exists a box connected at the no-out of B) then
show the subworkspace of B and
inform the operator that 'Dessa boxar är felaktigt kopplade.'

for any box B if there exists a box connected at the no-out of B and not (there exists a box connected at the yes-out of B) then
show the subworkspace of B and
inform the operator that 'Dessa boxar är felaktigt kopplade.'

for any box B if there exists a box connected at the no-out of B and there exists a box connected at the uncond-out of B then
show the subworkspace of B and
inform the operator that 'Dessa boxar är felaktigt kopplade.'

for any box B if there exists a box connected at the yes-out of B and there exists a box connected at the uncond-out of B then
show the subworkspace of B and
inform the operator that 'Dessa boxar är felaktigt kopplade.'

rules for handling the choise of colours

if any box B is active and not (show-step) then
change the background-color of every free-text upon the subworkspace of B to gray and
change the border-color of every free-text upon the subworkspace of B to yellow

if any box B is active and show-step then
change the background-color of every free-text upon the subworkspace of B to wheat and
change the border-color of every free-text upon the subworkspace of B to black

when remainder (the current second, 2) = 0 and not (show-step) and the box in track is active then
change the area2 icon-color of the box in track to black

when remainder (the current second, 2) /= 0 and not (show-step) and the box in track is active then
change the area2 icon-color of the box in track to red

the handling of a YES-answer

if answer is yes then
in order
conclude that answer has no value and
conclude that the box in track is non-active and
change the area2 icon-color of the first box in track to red and
conclude that the proc-answer of the last time-keeper in the time-list of the dot upon the workspace of the box in track = 'JA' and
conclude that the proc-hour of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current hour with expiration 1 week and
conclude that the proc-minute of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current minute with expiration 1 week and
conclude that the box connected at the yes-out of the box in track is active

the handling of a NO-answer

if answer is no then
in order
conclude that answer has no value and
conclude that the box in track is non-active and
change the area2 icon-color of the first box in track to red and
conclude that the proc-answer of the last time-keeper in the time-list of the dot upon the workspace of the box in track = 'NEJ' and
conclude that the proc-hour of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current hour with expiration 1 week and
conclude that the proc-minute of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current minute with expiration 1 week and
conclude that the box connected at the no-out of the box in track is active

the handling of an OK-answer

if answer is check then
in order
conclude that answer has no value and
conclude that the box in track is non-active and
change the area2 icon-color of the first box in track to red and
conclude that the proc-hour of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current hour with expiration 1 week and
conclude that the proc-minute of the last time-keeper in the time-list of the dot upon the workspace of the box in track = the current minute with expiration 1 week and
conclude that the box connected at the uncond-out of the box in track is active

the updating of "track"

if any box B is in-process and B is active then
in order
remove the first box from track and
insert B at the beginning of the box list track

if any box B is non-active then hide the subworkspace of B

if any box B is active and B is in-process then
invoke moving rules

rules for updating the history keeping lists

if any box B is active and show-step is false then
insert B at the end of the box list the steplist of the dot upon the workspace of B

if any box B is active and show-step is false then
in order
create a time-keeper TK and
insert TK at the end of the item list the time-list of the dot upon the workspace of B

colour-marking rules

if any box B is active then
change the area1 icon-color of B to green

if any box B is non-active then
change the area1 icon-color of the box to white

initiating the historical mode

if show-step is true and any dot S is in-process then
in order
start duplicate-list (the time-list of S, duplist-b) and
start duplicate-list (the steplist of S, duplist-a) and change the area2 icon-color of the last box in the steplist of S to lime-green and
conclude that the first box in the steplist of S is active and
show S 100 units above the center of the screen and
show prev-step-area and
conclude that list-number = 1

ending the historical mode

if any dot S is in-process then
in order
change the area2 icon-color of the last box in the steplist of S to black and
start empty-list (duplist-a) and
start empty-list (duplist-b) and
show the subworkspace of S and
conclude that every box upon the workspace of S is non-active and
conclude that show-step is false and
conclude that the last box in the steplist of S is active and
remove the last box from the steplist of S and
remove the last time-keeper from the time-list of S

step forward

unconditionally
in order
insert the first time-keeper in duplist-b at the end of the time-keeper list duplist-b and
remove the first time-keeper from duplist-b and
insert the first box in duplist-a at the end of the box list duplist-a and
remove the first box from duplist-a and
conclude that the last box in duplist-a is non-active and
conclude that the first box in duplist-a is active

step backwards

unconditionally
in order
insert the last time-keeper in duplist-b at the beginning of the time-keeper list duplist-b and
remove the last time-keeper from duplist-b and
insert the last box in duplist-a at the beginning of the box list duplist-a and
remove the last box from duplist-a and
conclude that the second box in duplist-a is non-active and
conclude that the first box in duplist-a is active

rules for updating the messages displayed to the operator in the historical mode

if any box B is active and B is alternative and show-step
then change the text of step-inf to 'Svaret var: [the proc-answer of the first time-keeper in duplist-b], klockan var: [the proc-hour of the first time-keeper in duplist-b] : [the proc-minute of the first time-keeper in duplist-b].'

if any box B is active and B is action and show-step
then change the text of step-inf to 'Detta utfördes klockan:[the proc-hour of the first time-keeper in duplist-b] : [the proc-minute of the first time-keeper in duplist-b].'

if any box is active and show-step and list-number = the number of elements in duplist-a then change the text of box-inf to 'Detta är sista boxen'

if any box is active and show-step and list-number = 1 then change the text of box-inf to 'Detta är första boxen'

if any box B is active and B is end and show-step
then change the text of step-inf to ''

if any box is active and show-step and list-number /= the number of elements in duplist-a and list-number /= 1 then change the text of box-inf to ''

automatically forward

when show-step and list-number < the number of elements in duplist-a and auto then in order conclude that list-number = list-number + 1 and invoke forward rules


X-DIST


Y-DIST


MOVE-PLAN

rules for the moving of plans

if any dot S is in-process then
conclude that x-dist = (the icon-x-position of the box in track - the icon-x-position of S) and
conclude that y-dist = (the icon-y-position of the box in track - the icon-y-position of S)

whenever x-dist receives a value and when any dot S is in-process and max(abs (x-dist/1500) , abs (y-dist/1100)) >= 1 then start move-plan (x-dist,y-dist, S ,show-step)

whenever x-dist receives a value and when max(abs(x-dist/1500),abs(y-dist/1100))<1 then
show the subworkspace of the first box in track with its bottom center at the bottom center of the screen

the updating of a message when a plan is terminated

if any box B is active and B is in-process and B is end and show-step is false then change the text of the step-mess upon the subworkspace of the dot nearest to B to 'Välj en annan plan genom att trycka på meny.'

if any box B is non-active and B is end then change the text of the step-mess upon the subworkspace of the dot nearest to B to ' '

displays a workspace containing the entrance-boxes for the plan

whenever the process-state of any dot S receives a value and when the process-state is in-process and not (there exists a box B such that (B is active and B is in-process)) then
show the workspace of the entrance-box named by the entrance-name of S with its bottom center at the bottom center of the screen

if any dot S is not-in-process then hide the subworkspace of S and hide the workspace of S

rules for displaying the operator with a way of answering

if any box B is active and B is altemative and B is in-process and show-step is false then
show option1

if any box B is active and B is action and B is in-process and show-step is false then show option2

decides the type for a box

for any box B if there exists a box connected at the yes-out of B then conclude that B is alternative

for any box B if there exists a box connected at the uncond-out of B then conclude that B is action

for any box B if not (there exists a box connected at the yes-out of B) and not (there exists a box connected at the uncond-out of B) then conclude that B is end

Procedures

CLICK-BUTTON

animation of button-clicking

CLICK-BUTTON, a procedure	
Notes	OK
User restrictions	none
Tracing and breakpoints	default
Default procedure priority	6

```
click-button (b: class general-button)
begin
change the center icon-color of b to black;
change the center icon-color of b to wheat
end
```

EMPTY-LIST

empties a list given as an argument

EMPTY-LIST, a procedure	
Notes	OK
User restrictions	none
Tracing and breakpoints	default
Default procedure priority	6

```
empty-list (org-list: class item) ctn: quantity;
   begin for ctn = 1 to the number of elements
   in org-list do
remove the first box from org-list;
end
end
```

DUPLICATE-LIST

copies a list given as an argument

DUPLICATE-LIST, a procedure	
Notes	OK
User restrictions	none
Tracing and breakpoints	default
Default procedure priority	6

```
duplicate-list (org-list: class item, dup-list: class
   item)
 alt: symbol;
cnt: quantity; begin  for cnt = 1 to the number of
   elements in org-list do
insert the first item in org-list at the end of the item
   list dup-list;
remove the first item from org-list;
insert the last item in dup-list at the end of the item
   list org-list ;
end;
remove the last item from dup-list;
end
```

MOVE-PLAN

handles the moving of a plan

MOVE-PLAN, a procedure	
Notes	OK
User restrictions	none
Tracing and breakpoints	default
Default procedure priority	6

```
move-plan (dist1: quantity,dist2: quantity, c-p:
   class spot, s-step: truth-value ) t: symbol;
begin
if abs(dist1)> 1500 then move c-p by (round
   (dist1) - 1000*dist1/abs(dist1) ,0);
if abs (dist2) >1100 then move c-p by
   (0,dist2*0.8);
show c-p 100 units above the center of the
   screen;
if s-step is false then
show the subworkspace of c-p;
collect data t = the type of the box in track
   end ;
if t is alternative and s-step is false then show
   option1 ;
if t is action and s-step is false then show
   option2;
if s-step is true then show prev-step-area;
show the subworkspace of the box in track
   with its bottom center at the bottom center
   of the screen end
```

Button-definitions and their corresponding actions defined through user-menu-choises

GENERAL-BUTTON

ANSWER-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	answer-button
Condition	true
Action	in order hide the workspace of the item and conclude that answer = the value of the value-of-answer of the item
Action priority	2

CHOISE-BUTTON

selecting a plan

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	choise-button
Condition	not (the dot named by the dot-name of the item is in-process)
Action	in order change the arrow icon-color of every choise-button to salmon and change the arrow icon-color of the item to transparent and conclude that every box is not-in-process and conclude that every dot is not-in-process and show the dot named by the dot-name of the item 100 units above the center of the screen and conclude that every box upon the workspace of the dot named by the dot-name of the item is in-process and hide the workspace of the item and show the subworkspace of the dot named by the dot-name of the item and conclude that the dot named by the dot-name of the item is in-process
Action priority	2

RESET-BUTTON

resetting a plan

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	reset-button
Condition	true
Action	in order start click-button (the item) and change the area2 icon-color of every box upon the workspace of the dot named by the dot-name of the item to black and conclude that every box upon the workspace of the dot named by the dot-name of the item is non-active and show the workspace of the entrance-box named by the entrance-name of the dot named by the dot-name of the item and start empty-list (the steplist of the dot named by the dot-name of the item) and start empty-list (the time-list of the dot named by the dot-name of the item) and hide option1 and hide option2
Action priority	2

STEP-BUTTON

entering of the historical mode

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	step-button
Condition	the number of elements in the steplist of the dot named by the dot-name of the item > 1
Action	in order start click-button (the item) and conclude that show-step is true and conclude that the first box in track is non-active and invoke step rules
Action priority	2

Button-definitions (cont.)

HIDE-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	hide-button
Condition	true
Action	hide the workspace of the item
Action priority	2

MENU-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	menu-button
Condition	true
Action	in order start click-button (the item) and show menu with its bottom center at the bottom center of the screen
Action priority	2

STOP-BUTTON

ending the historical mode

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	stop-button
Condition	true
Action	in order conclude that auto is false and hide the subworkspace of the first box in duplist-a and hide the workspace of the item and invoke get-back rules
Action priority	2

GO-TO-SUB-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	go-to-sub-button
Condition	true
Action	show the subworkspace of the item
Action priority	2

FORWARD-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	forward-button
Condition	list-number < the number of elements in duplist-a
Action	in order conclude that list-number = list-number + 1 and conclude that auto is false and invoke forward rules
Action priority	2

BACKWARD-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	backward-button
Condition	list-number > 1
Action	in order conclude that list-number = list-number - 1 and conclude that auto is false and invoke back rules
Action priority	2

Further button-definitions and user-menu-choices for other items

AUTO-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	auto-button
Condition	true
Action	conclude that auto is true
Action priority	2

RESTART-BUTTON

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	restart-button
Condition	true
Action	conclude that every box is non-active and conclude that every box is not-in-process and conclude that every spot is not-in-process and start empty-list (the steplist of every spot) and start empty-list (the time-list of every spot) and change the area2 icon-color of every box to black and hide menu and hide the workspace of every entrance-box and hide option1 and hide option2 and change the arrow icon-color of every choise-button to salmon and hide the workspace of the item
Action priority	2

ENTRANCE-BOX

an user-menu-choice	
Notes	OK
User restrictions	none
Label	push
Applicable class	entrance-box
Condition	true
Action	conclude that the box named by the box-name of the item is active and hide the workspace of the item
Action priority	2

user--menu-choice for borderless-free-text

an user-menu-choice	
Notes	OK
User restrictions	none
Label	borderless-text-info
Applicable class	borderless-free-text
Condition	true
Action	show the subworkspace of every add-inf upon the subworkspace of the box nearest to the item with its center at the center of the screen
Action priority	2

BOX

an user-menu-choice	
Notes	OK
User restrictions	none
Label	get-inf
Applicable class	box
Condition	true
Action	show the subworkspace of every add-inf upon the subworkspace of the item with its center at the center of the screen
Action priority	2

user-menu-choice for free-text

an user-menu-choice	
Notes	OK
User restrictions	none
Label	text-inf
Applicable class	free-text
Condition	true
Action	show the subworkspace of the add-inf nearest to the item with its center at the center of the screen
Action priority	2

user-menu-choice for kb-workspace

an user-menu-choice	
Notes	OK
User restrictions	none
Label	hide-space
Applicable class	kb-workspace
Condition	there exists an add-inf such that (the subworkspace of the add-inf is the same object as the item)
Action	hide the item
Action priority	2

Appendix B

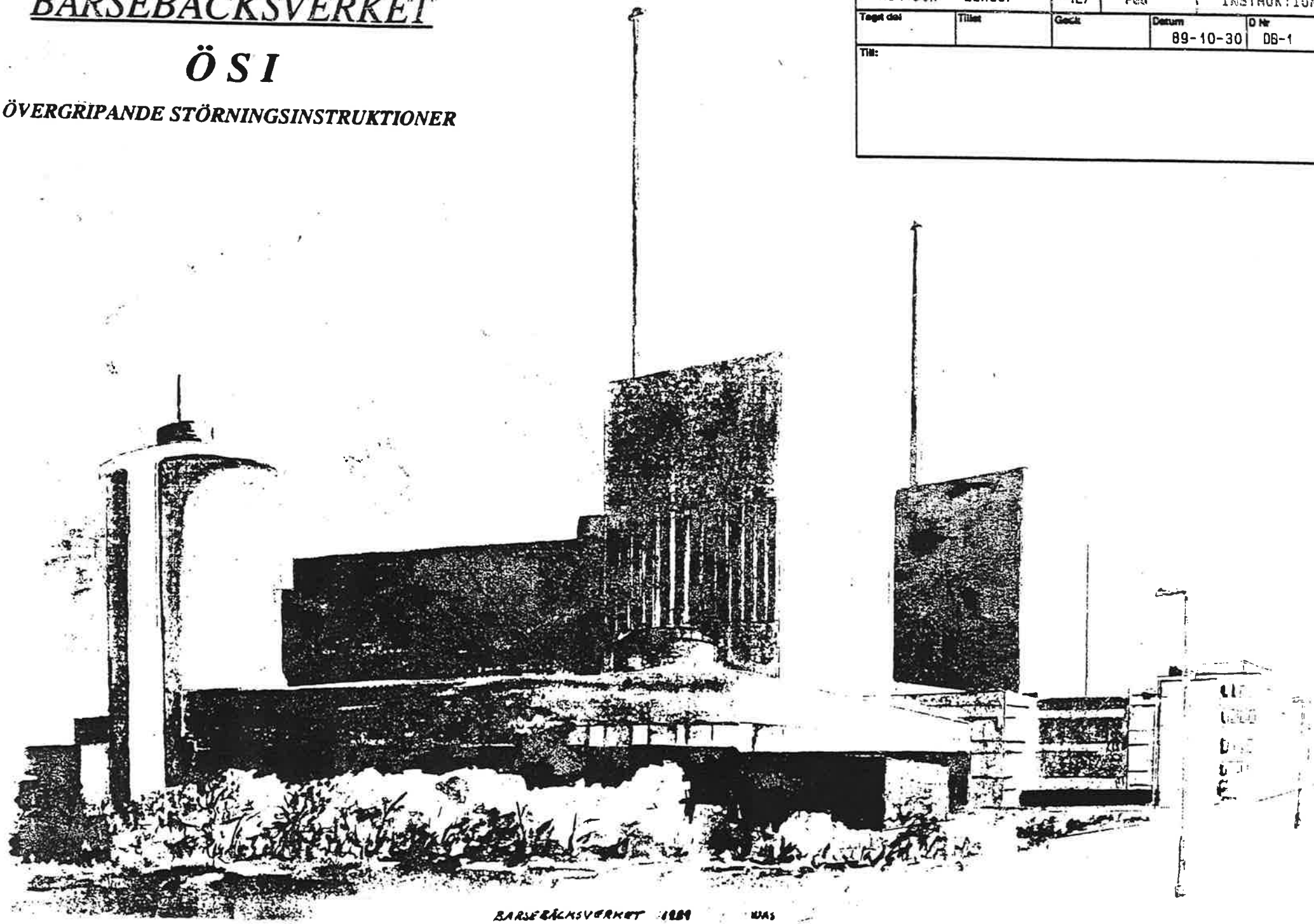
The ÖSI-Plans

BARSEBÄCKSVERKET

ÖSI

ÖVERGRIPANDE STÖRNINGSINSTRUKTIONER

Från		Tel	Utskr	Dokumentnummer
Kenneth Zander		127	Ped	INSTRUKTION
Taget dat	**Tillat**	**Godk**	**Datum**	**D Nr**
			89-10-30	DB-1 Utg1
Till:				



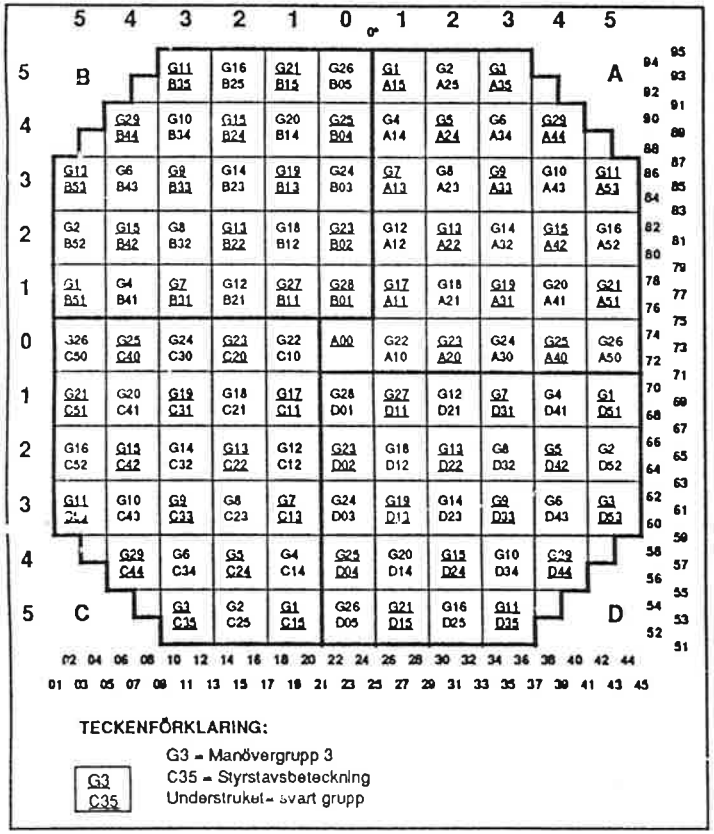
BARSEBÄCKSVERKET 1989 WAS

INGÅNGSVÄRDEN TILL DE ÖVERGRIPANDE STÖRNINGSINSTRUKTIONERNA (ÖSI)

REAKTIVITET	ONORMAL VATTENNIVÅ I REAKTOR-TANKEN (RT)	ONORMALT TRYCK I REAKTOR-TANKEN (RT)	ONORMALT TRYCK I REAKTOR-INNESLUTNINGEN (PS)	ONORMAL TEMPERATUR I KONDENSATIONS-BASSÄNGEN (316)	ONORMAL LÅG VATTENNIVÅ I KONDENSATIONS-BASSÄNGEN (316)	ONORMAL HÖG VATTENNIVÅ I KONDENSATIONS-BASSÄNGEN (316)	DIFFUST LÄCKAGE
SS eller överskridet gränsvärde.	Vattennivån i RT lägre än 2,9 m eller högre än 4,8 m.	Trycket högre än 74 bar eller temperatur-sänkningen större än 60°C.	Trycket i (PS) högre än 1,5 bar.	Temp i kondensations bassängen större än 28°C.	Vattennivån i kondensations-bassängen 6,0 m eller lägre.	Vattennivån i kondensations-bassängen högre än + 6,35 m.	Förhöjd aktivitet i PS.
Alla stavar inne?	Vattennivå större än 4,8m?	Trycket högre än 74 bar?	Tryck i PS 1,5 bar - 4,5 bar?	Temperaturen 28 °C - 55 °C?	Vattennivån 4.3m - 6,0m ?	Vattennivån 6,35m - 7,75m ?	Förhöjd aktivitet i PS?
Reaktoreffekt större än 5%?	Vattennivå 0,7m - 2,9 m?	Temperatur-sänkningen större än 60°C/h?	Tryck i PS 4,5 bar - 6,5 bar?	Temperaturen 55 °C - 95 °C?	Vattennivån 3,0m - 4,3m ?	Vattennivån 7,75m - 11m ?	
Fler än 4 styr-stavar ej inne?	Vattennivå -1,8m - 0,7 m?		Tryck i PS större än 6,5 bar?	Temperaturen 95 °C - 150 °C?	Vattennivån lägre än 3,0 m?	Vattennivån 11m - 12,9m ?	
	Vattennivå lägre än - 1,8m?			Temperaturen högre än 150 °C?		Vattennivån 12,9m - 15,5m ?	
						Vattennivån 15,5m - 26,0m ?	
						Vattennivån högre än 26,0m ?	

REAKTIVITET

Härdbild



Drivdon i resp. SS-grupp

SS - grupp (PA 10) :	01	02	03	04	05	06	07	08	09
SS - ventil (KD 12 - 16):	V401	V402	V403	V404	V405	V406	V407	V408	V409
Drivdon:	A40 B02 B44 C40 C44 D02 D44	A12 A50 B34 C30 C34 D12	A20 A24 B22 C22 C24	A30 A34 B12 C12 C50 D34	A00 A42 B04 B42 C42 D04 D42	A10 A14 A52 B32 C32 D14 D52	A22 B24 C20 C24 D22	A32 B14 B52 C10 C14 C52 D32	A44 B13 B51 C15 C33 D11 D53
SS - grupp (PA 10) :	10	11	12	13	14	15	16	17	
SS - ventil (KD 12 - 16):	V410	V411	V412	V413	V414	V415	V416	V417	
Drivdon:	A21 B05 B21 C43 D03 D43	A11 A15 A53 B53 C31 D13 D51	A43 B03 B43 C21 D05 D21	A13 A51 B31 B35 C13 D33	A25 A41 B01 B25 B41 C23 D23	A31 A35 B33 C11 C35 D35	A23 B23 C25 C41 D01 D25 D41	A33 B11 B15 C33 D15 D31	

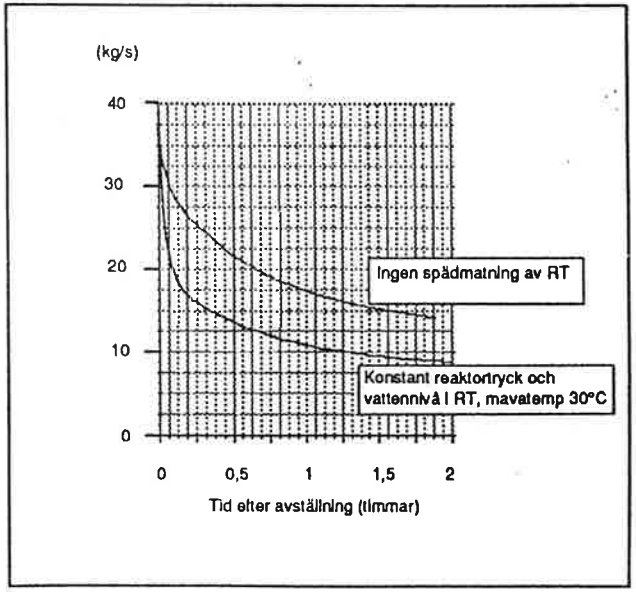
Automatiska SS-villkor

SS2	Alla stavar ej inne (för återställning)
SS3	Omkopplare ej i SS
SS4	Vattennivån i reaktortanken mindre än 2,9 meter (L2)
SS5	Vattennivån i reaktortanken högre än 4,8 meter (H2)
SS6	Trycket i reaktortanken större än 74 bar (H1)
SS7	Hög pulsfrekvens SRM vid låg effekt
SS8	Extra hög effekt IRM vid låg PRM eller reaktortryck mindre än 60 bar.
SS9	Filtrerat flow/flux
SS10	Ofiltrerat flow/flux
SS11	Dumpförbud till turbin p g a avvikande oljetryck
SS13	Utlöst A-, Y- eller I-isolering
SS14,15	Hög aktivitet i ångledning
SS16	Hög konduktivitet
SS17	Högt tryck i kondensorn
SS18	Högt tryck i MÖH

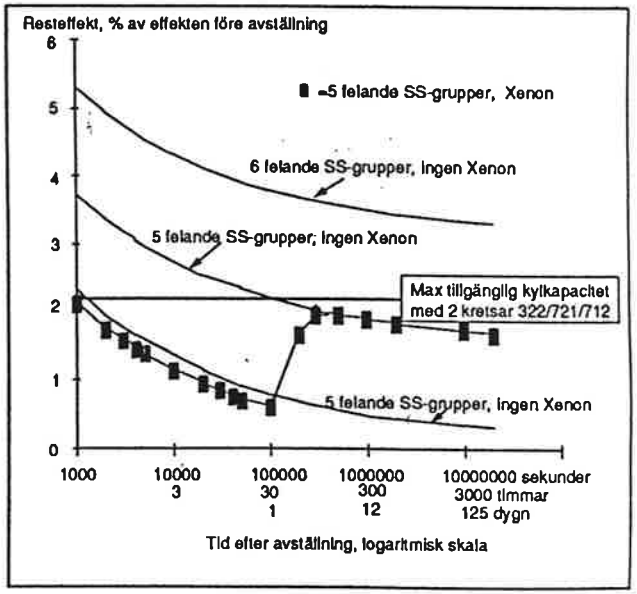
Temperaturförloppet i kondensationsbassängen vid utebliven reaktoravställning redovisas i ONORMAL TEMPERATUR I 316.

Tryckförloppet i reaktorinneslutningen vid utebliven reaktoravställning redovisas i ONORMALT TRYCK I PS.

314-flöde vid normal avställning

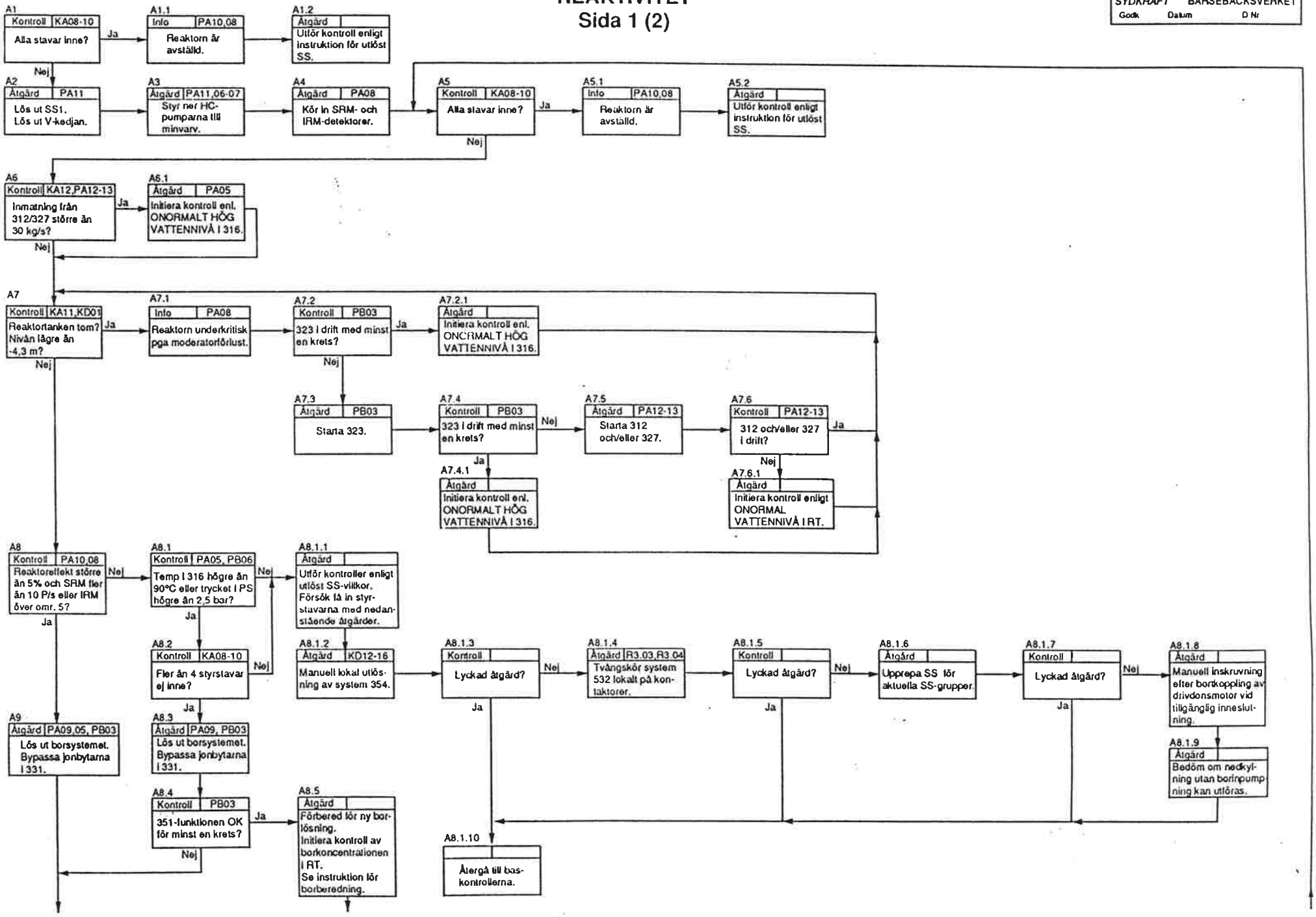


Resteffekt s f a antal utevblivna SS-grupper



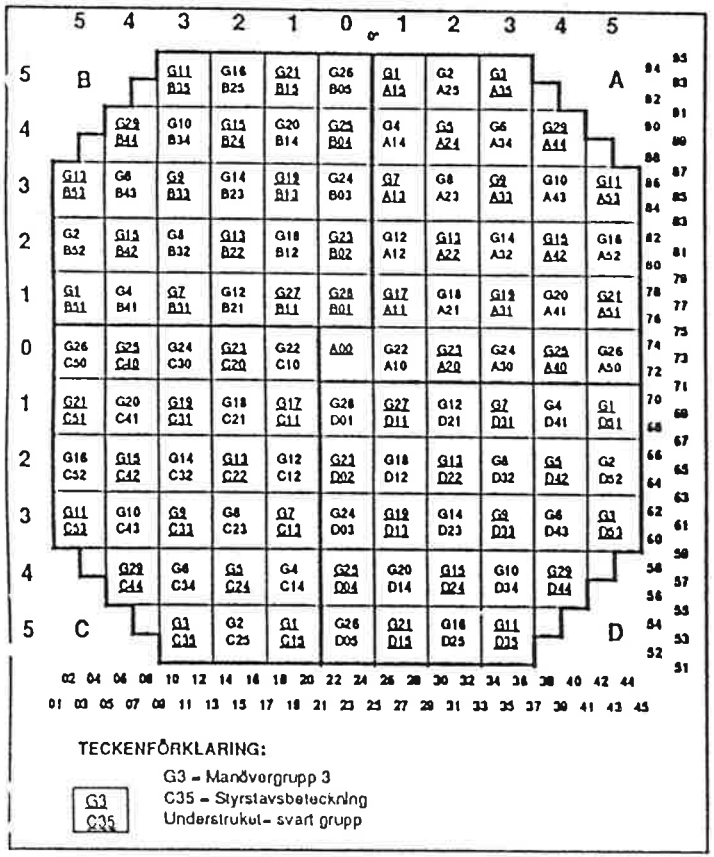
REAKTIVITET
Sida 1 (2)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr



REAKTIVITET

Härdbild



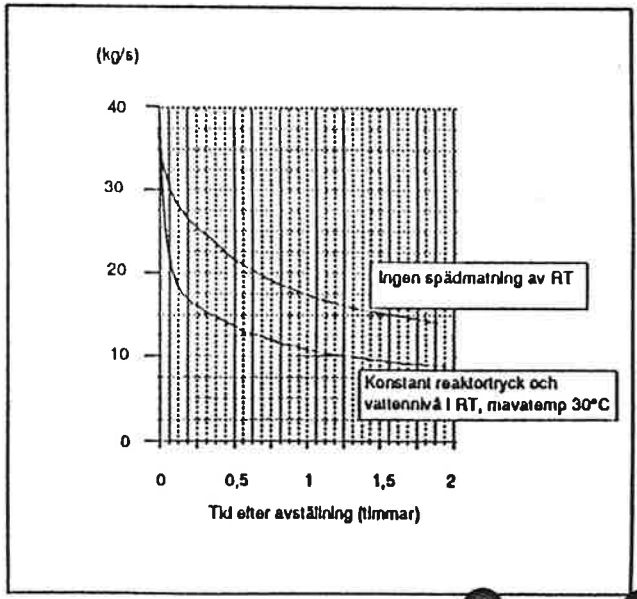
Drivdon i resp. SS-grupp

SS - grupp (PA 10):	01	02	03	04	05	06	07	08	09
SS - ventil (KD 12 - 16):	V401	V402	V403	V404	V405	V406	V407	V408	V409
Drivdon:	A40	A12	A28	A38	A00	A18	A22	A32	A44
	B02	A50	A24	A34	A42	A14	B24	B14	B13
	B44	B34	B22	B12	B04	A52	C20	B52	B51
	C40	C30	C22	C12	B42	B32	C24	C18	C15
	C44	C34	C24	C50	C42	C32	D22	C14	C53
	D02	D12		D34	D04	D14		C52	D11
	D44				D42	D52		D32	D53
SS - grupp (PA 10):	10	11	12	13	14	15	16	17	
SS - ventil (KD 12 - 16):	V410	V411	V412	V413	V414	V415	V416	V417	
Drivdon:	A21	A11	A43	A13	A25	A31	A23	A33	
	B05	A15	B03	A51	A41	A35	B23	B11	
	B21	A53	B43	B31	B01	B33	C25	B15	
	C43	B53	C21	B35	B25	C11	C41	C33	
	D03	C31	D05	C13	B41	C35	D01	D15	
	D43	D13	D21	D33	C23	C51	D25	D31	
		D51			D23	D35	D41		

Automatiska SS-villkor

SS2	Alla stavar ej inne (för återställning)
SS3	Omkopplare ej i SS
SS4	Vattennivån i reaktortanken mindre än 2,9 meter (L2)
SS5	Vattennivån i reaktortanken högre än 4,8 meter (H2)
SS6	Trycket i reaktortanken större än 74 bar (H1)
SS7	Hög pulsfrekvens SRM vid låg effekt
SS8	Extra hög effekt IRM vid låg PRM eller reaktortryck mindre än 60 bar.
SS9	Filtrerat flow/flux
SS10	Ofiltrerat flow/flux
SS11	Dumpförbud till turbin p g a avvikande oljetryck
SS13	Utlöst A-, Y- eller I-isolering
SS14,15	Hög aktivitet i återledning
SS16	Hög konduktivitet
SS17	Högt tryck i kondensorn .
SS18	Högt tryck i MÖH

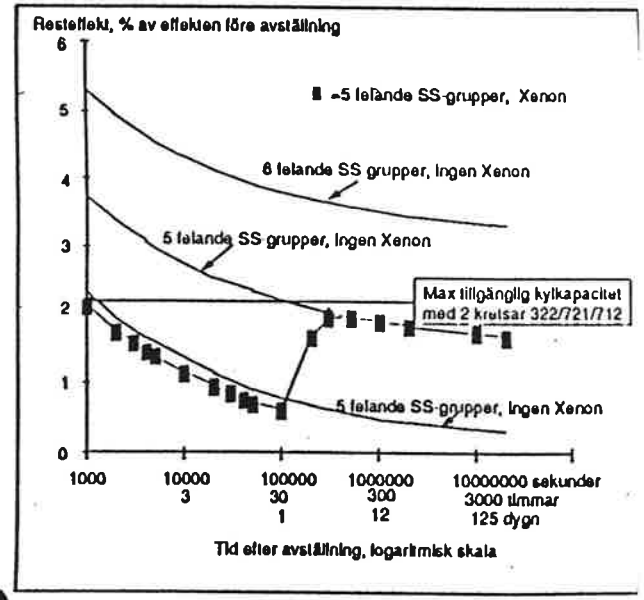
314-flöde vid normal avställning



Temperaturförloppet i kondensationsbassängen vid utebliven reaktoravställning redovisas i ONORMAL TEMPERATUR I 316.

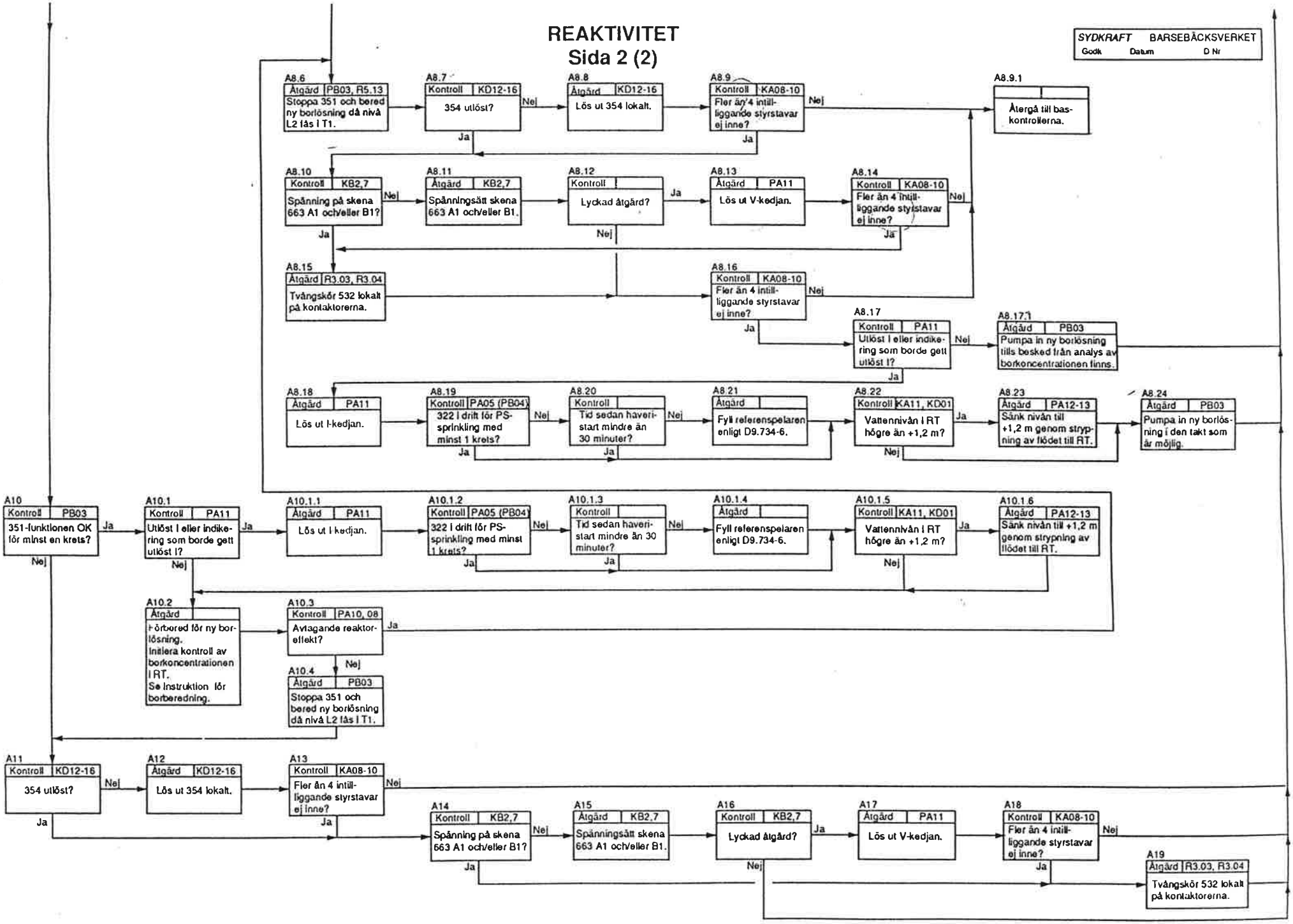
Tryckförloppet i reaktorinneslutningen vid utebliven reaktoravställning redovisas i ONORMALT TRYCK I PS.

Resteffekt s f a antal utevblivna SS-grupper



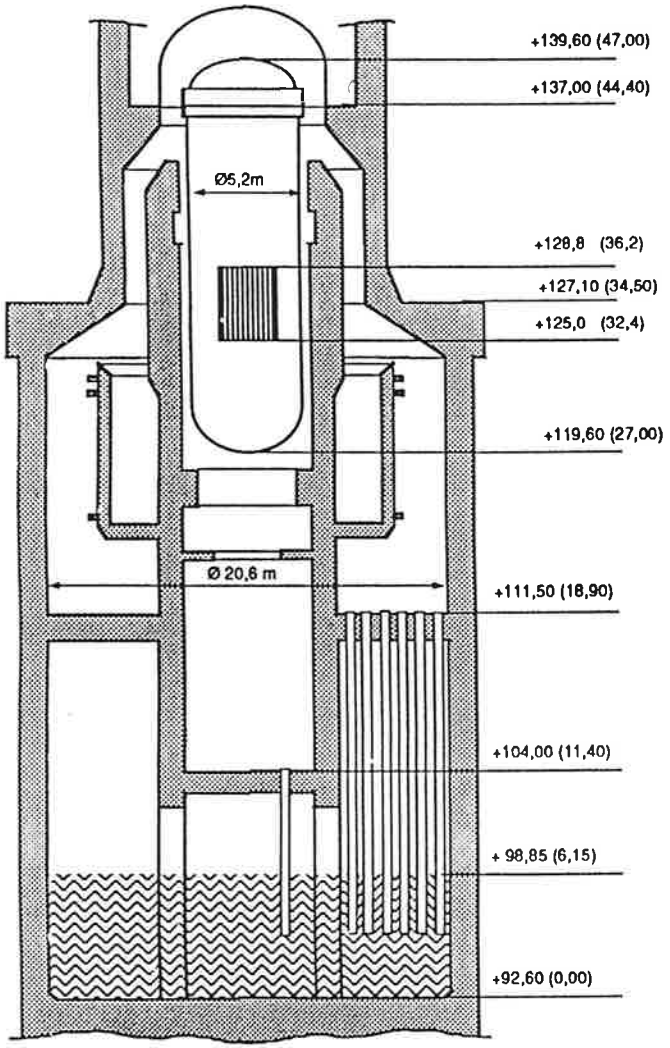
REAKTIVITET
Sida 2 (2)

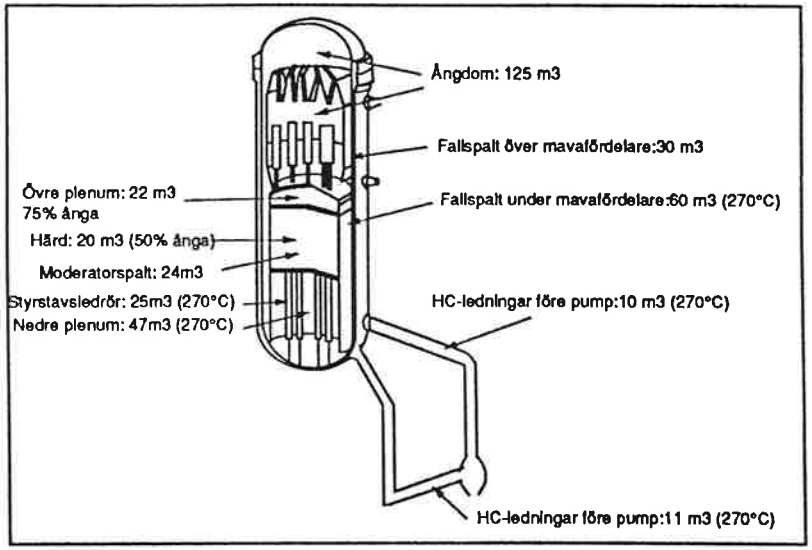
SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr



ONORMAL VATTENNIVÅ I REAKTORTANKEN (RT)

Reaktorinneslutning





Automatiska funktioner vid onormal vattennivå I RTsamt utlöst I-isolering

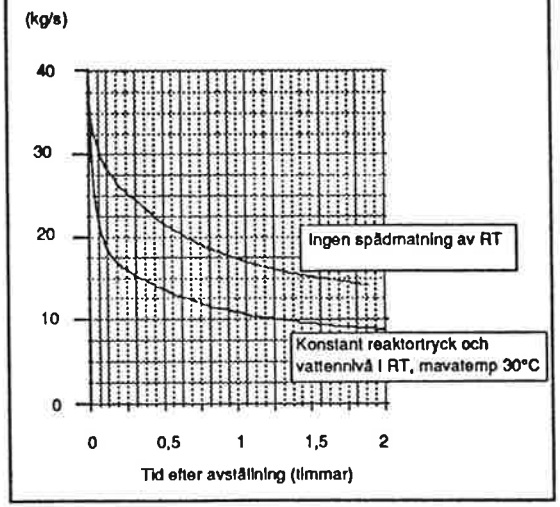
H2 (+4,74)	SS5
L2 (+2,84)	SS4
L3 (+0,64)	Y20 och TB1
L4 (-1,86)	TB2
I-isolering	Nivåbörvärdet sänks till +1,3

Tankstutsarnas placering

Ångstutsarna 6 st + 6,31
Mavastutsarna 4 st +1,84

Tryckförloppet vid utlöst tvångsblådning (TB) redovisas I ONORMALT TRYCK I REAKTORTANKEN.

314-flöde vid normal avställning



Rumsvakternas placering vid olika I-, Y-, A- och M-villkor

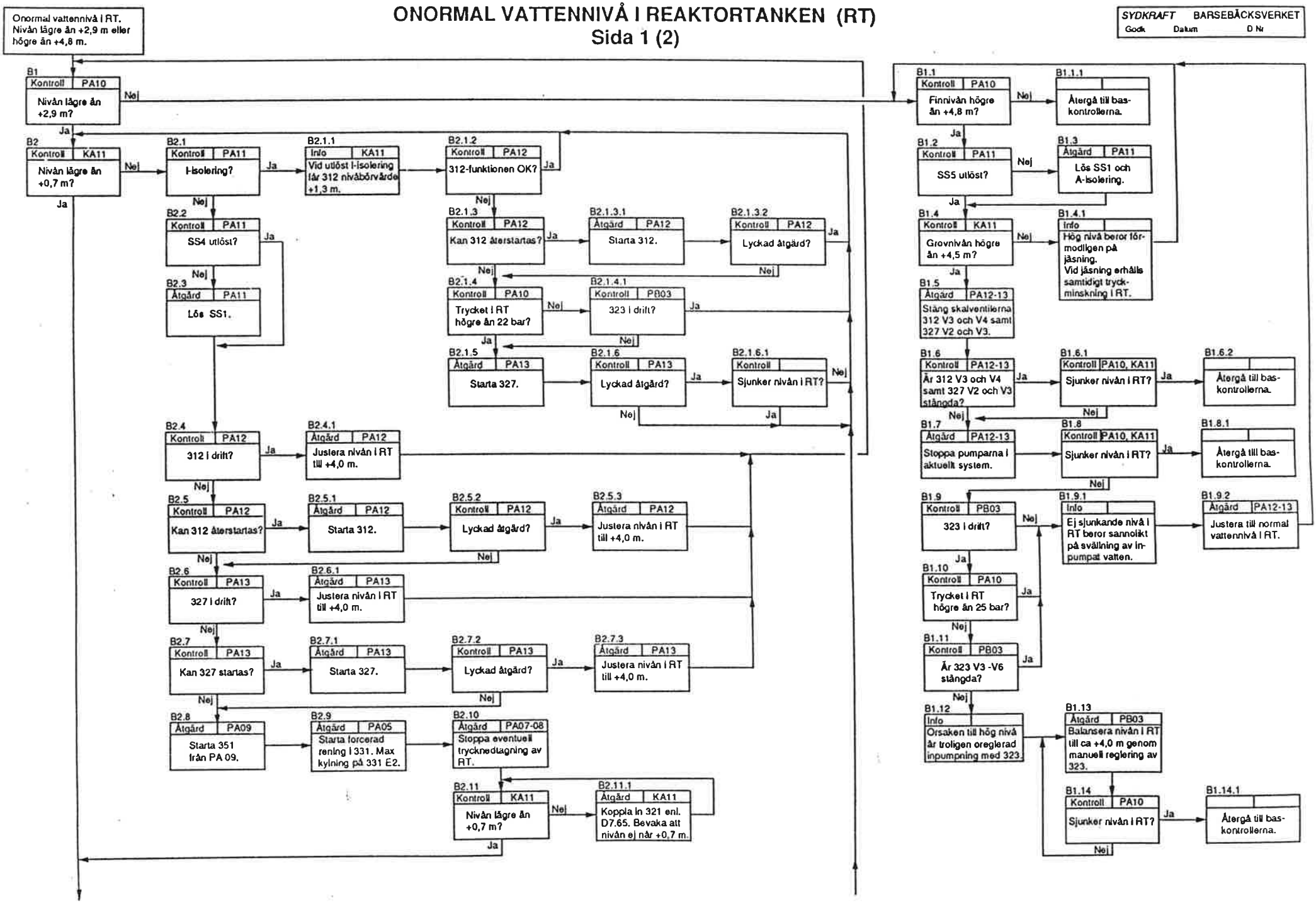
I3	546 K101 - K103	R4.60	311, 313	
I4	546 K105 - K107	R1.60	311, 313	
I5	546 K501 - K503	R4.60	311, 313	
I6	546 K505 - K507	R4.60	311, 313	
I7	546 K509 - K511	R1.60	311, 313	
Y3	546 K109 - K111	R1.31	321	
Y4	546 K113 - K115	R1.32	321	
Y5	546 K117 - K119	R4.31	321	
Y6	546 K121 - K123	R6.13	354	
Y7	546 K125 - K127	R6.31	331, 354	
Y8	546 K129 - K131	R1.40	321, 327, 331, 354	
Y9	546 K133 - K135	R1.57	321, 327, 331, 354	
Y10	546 K137 - K139	R6.19	331	
Y11	546 K513 - K515	R1.31	321	
Y12	546 K517 - K519	R1.32	321	
Y13	546 K521 - K523	R4.53	331	
Y14	546 K525 - K527	R5.31	331	
Y15	546 K529 - K531	R6.13	354	
Y16	546 K533 - K535	R6.31	331, 354	
Y17	546 K537 - K539	R1.40	321, 327, 331, 354	
Y18	546 K541 - K543	R1.57	321, 327, 331, 354	
Y20	211 K406(L3), K408(L3), K411(L3)			
Y21	546 K401 - K403	R1.31	321	
Y22	546 K405 - K407	R1.32	321	
Y23	546 K409 - K411	R1.47	321, 327, 331, 354	
Y25	546 K417 - K419	R4.31	321 (331)	
Y26	546 K421 - K423	R5.31	331	
Y29	546 K441 - K443	T9405	327	
Y30	546 K445 - K447	T9608	327	
Y32	546 K461 - K463	T9905	327	
Y35	546 K853 - K855	T0318	327	
Y36	546 K905 - K907	R1.47	321, 327,331, 354	

A3	546 K141 - K143	T9401	311	
A4	546 K145 - K147	T9912	311	
A5	546 K149 - K151	T9921	312	
A6	546 K153 - K155	T9922	312	
A7	546 K545 - K547	R7.17	311	
A8	546 K549 - K551	R2.17	311	
A9	546 K553 - K555	T0308	311	
A10	546 K557 - K559	T1404	311	
A11	546 K561 - K563	T9921	312	
A12	546 K565 - K567	T9922	312	
A13	546 K161 - K163	T1404	311	
A14	Utöst kanal I M-kedjan			
M3	546 K449 - K451	T9801	462	
M4	546 K453 - K455	T9802	462	
M5	546 K437 - K439	T9404	462	
M6	546 K489 - K491	T9915	462	
M7	546 K473 - K475	T9912	451, 462	
M8	546 K465 - K467	T9910	462	
M9	546 K825 - K827	T0003	462	
M10	546 K477 - K479	T9912	451, 462	
M11	546 K481 - K483	T9912	451, 462	
M12	546 K485 - K487	T9914	462	
M13	546 K909 - K911	T9926	462	
M14	546 K805 - K807	T9919	332, 462	
M15	546 K857 - K859	T0402	332	
M16	546 K869 - K871	T0705	332	
M17	546 K873 - K875	T0706	332	
M18	546 K877 - K879	T0707	332	
M19	546 K881 - K883	T0708	332	
M20	546 K885 - K887	T0709	332	
M21	546 K889 - K891	T0710	332	
M22	546 K817 - K819	T9923	462	
M23	546 K821 - K823	T9924	462	
M24	546 K901 - K903	T1401	312	
M25	546 K921 - K923	T1401	312	
M26	546 K917 - K919	T1401	312	
M27	546 K913 - K915	T1402	312	
M28	546 K861 - K863	T0601	453, 462	
M29	546 K809 - K811	T9921	312	
M30	546 K813 - K815	T9922	312	
M31	546 K425 - K427	R6.16	312	
M32	546 K429 - K431	R6.18	312	

ONORMAL VATTENNIVÅ I REAKTORTANKEN (RT)
Sida 1 (2)

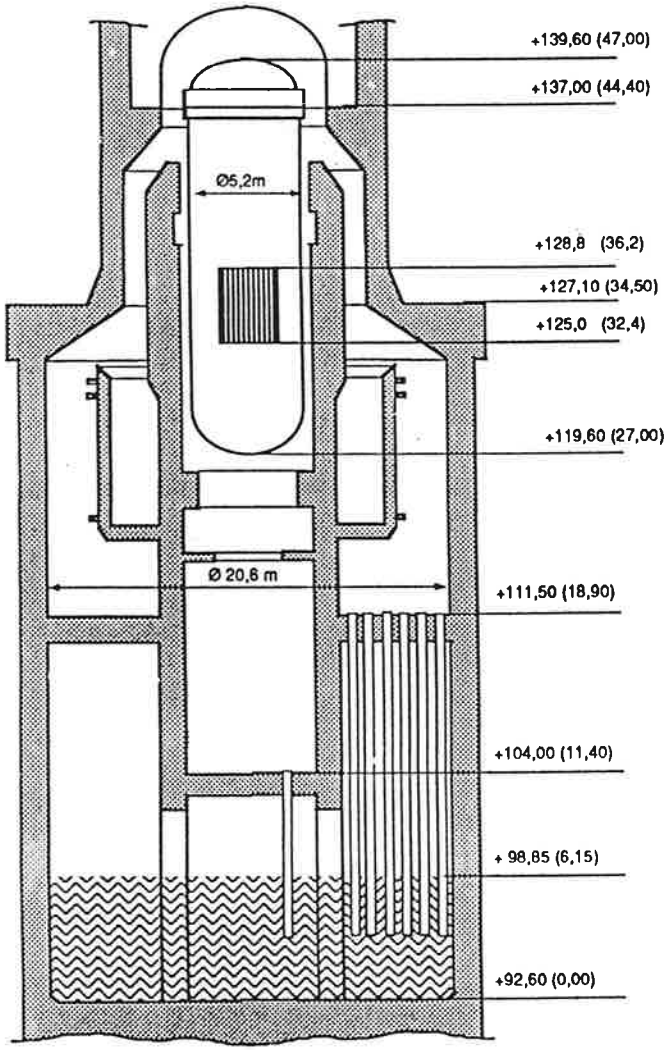
SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr

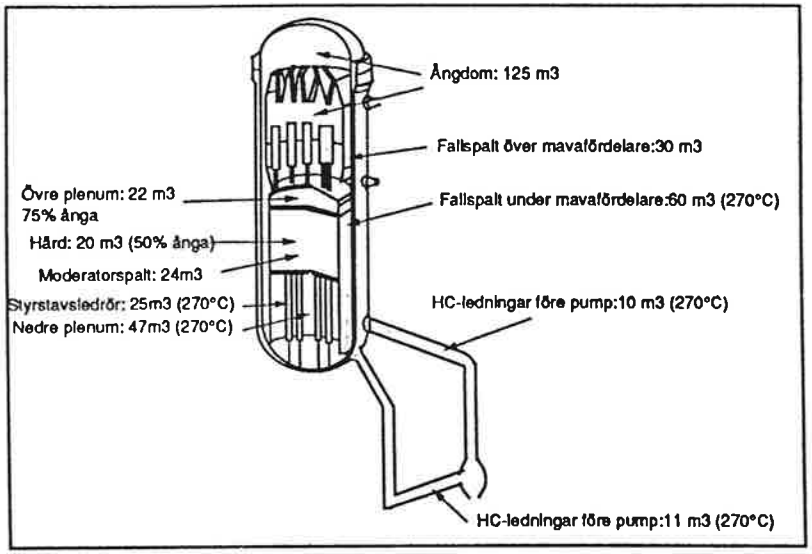
Onormal vattennivå i RT. Nivån lägre än +2,9 m eller högre än +4,8 m.



ONORMAL VATTENNIVÅ I REAKTORTANKEN (RT)

Reaktorinneslutning





Automatiska funktioner vid onormal vattennivå i RTsamt utlöst I-isolering

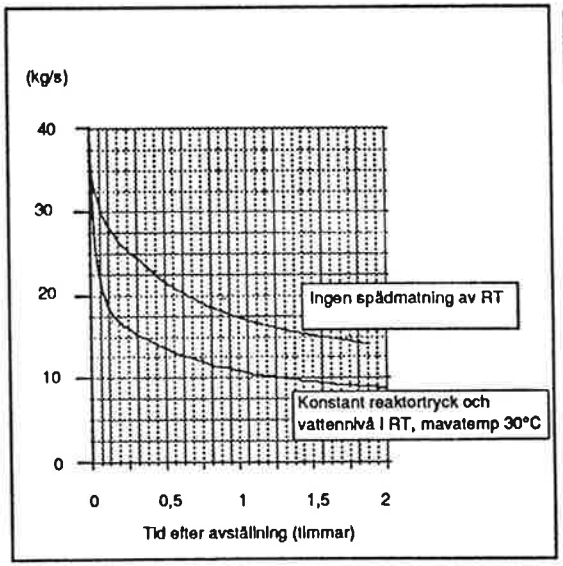
H2 (+4,74)	SS5
L2 (+2,84)	SS4
L3 (+0,64)	Y20 och TB1
L4 (-1,86)	TB2
I-isolering	Nivåbörvärdet sänks till +1,3

Tankstutsarnas placering

Ångstutsarna 6 st + 6,31
Mavastutsarna 4 st +1,84

Tryckförloppet vid utlöst tvångsblådning (TB) redovisas I ONORMALT TRYCK I REAKTORTANKEN.

314-flöde vid normal avställning



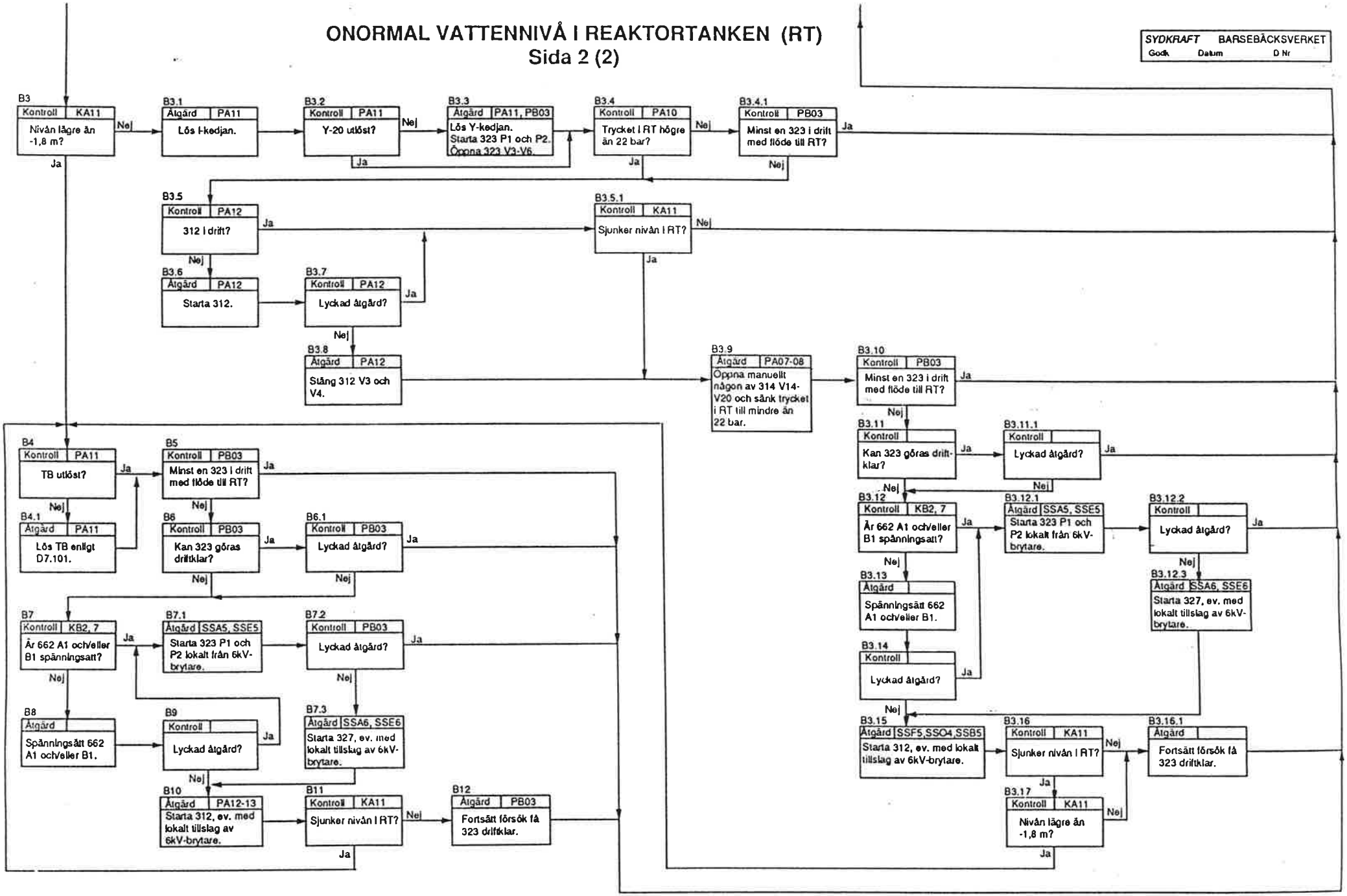
Rumsvakternas placering vid olika I-, Y-, A- och M-villkor

I3	546 K101 - K103	R4.60	311, 313
I4	546 K105 - K107	R1.60	311, 313
I5	546 K501 - K503	R4.60	311, 313
I6	546 K505 - K507	R4.60	311, 313
I7	546 K509 - K511	R1.60	311, 313
Y3	546 K109 - K111	R1.31	321
Y4	546 K113 - K115	R1.32	321
Y5	546 K117 - K119	R4.31	321
Y6	546 K121 - K123	R6.13	354
Y7	546 K125 - K127	R6.31	331, 354
Y8	546 K129 - K131	R1.40	321, 327, 331, 354
Y9	546 K133 - K135	R1.57	321, 327, 331, 354
Y10	546 K137 - K139	R8.19	331
Y11	546 K513 - K515	R1.31	321
Y12	546 K517 - K519	R1.32	321
Y13	546 K521 - K523	R4.53	331
Y14	546 K525 - K527	R5.31	331
Y15	546 K529 - K531	R6.13	354
Y16	546 K533 - K535	R6.31	331, 354
Y17	546 K537 - K539	R1.40	321, 327, 331, 354
Y18	546 K541 - K543	R1.57	321, 327, 331, 354
Y20	211 K406(L3), K408(L3), K411(L3)		
Y21	546 K401 - K403	R1.31	321
Y22	546 K405 - K407	R1.32	321
Y23	546 K409 - K411	R1.47	321, 327, 331, 354
Y25	546 K417 - K419	R4.31	321 (331)
Y26	546 K421 - K423	R5.31	331
Y29	546 K441 - K443	T9405	327
Y30	546 K445 - K447	T9608	327
Y32	546 K461 - K463	T9905	327
Y35	546 K853 - K855	T0318	327
Y36	546 K905 - K907	R1.47	321, 327,331, 354

A3	546 K141 - K143	T9401	311
A4	546 K145 - K147	T9912	311
A5	546 K149 - K151	T9921	312
A6	546 K153 - K155	T9922	312
A7	546 K545 - K547	R7.17	311
A8	546 K549 - K551	R2.17	311
A9	546 K553 - K555	T0308	311
A10	546 K557 - K559	T1404	311
A11	546 K561 - K563	T9921	312
A12	546 K565 - K567	T9922	312
A13	546 K161 - K163	T1404	311
-A14	Utöst kanal I M-kedjan		
M3	546 K449 - K451	T9801	462
M4	546 K453 - K455	T9802	462
M5	546 K437 - K439	T9404	462
M6	546 K489 - K491	T9915	462
M7	546 K473 - K475	T9912	451, 462
M8	546 K465 - K467	T9910	462
M9	546 K825 - K827	T0003	462
M10	546 K477 - K479	T9912	451, 462
M11	546 K481 - K483	T9912	451, 462
M12	546 K485—K487	T9914	462
M13	546 K909 - K911	T9926	462
M14	546 K805 - K807	T9919	332, 462
M15	546 K857 - K859	T0402	332
M16	546 K869 - K871	T0705	332
M17	546 K873 - K875	T0706	332
M18	546 K877 - K879	T0707	332
M19	546 K881 - K883	T0708	332
M20	546 K885 - K887	T0709	332
M21	546 K889 - K891	T0710	332
M22	546 K817 - K819	T9923	462
M23	546 K821 - K823	T9924	462
M24	546 K901 - K903	T1401	312
M25	546 K921 - K923	T1401	312
M26	546 K917 - K919	T1401	312
M27	546 K913 - K915	T1402	312
M28	546 K861 - K863	T0601	453, 462
M29	546 K809 - K811	T9921	312
M30	546 K813 - K815	T9922	312
M31	546 K425 - K427	R6.16	312
M32	546 K429 - K431	R6.16	312

ONORMAL VATTENNIVÅ I REAKTORTANKEN (RT)
Sida 2 (2)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr



ONORMALT TRYCK I REAKTORTANKEN (RT)

Elstyrd 314-blåsning:

Elstyrd 314 blåsning sker på följande signaler
TS*D
A- eller I-isolering
Y20 Vattennivå L3 (0,7 m) i reaktortanken
SS5 Vattennivå H2 (4,8 m) i reaktortanken
SS6 Trycket i reaktortanken större än H1 (74 bar)
SS14 Hög aktivitet i ångledningar
SS15 Hög aktivitet i ångledningar
SS18 Högt tryck i MÖH

Antalet 314-ventiler som aktiveras beror av reaktoreffektenenligt följande:

Effekt	V50	V51	V14	V15	V16	V17	V18	V19	V20
≤ 5%	X	X							
≤30%	X	X						X	X
≤60%	X	X			X		X	X	X
>60%	X	X	X	X	X	X	X	X	X

Efter 6 sekunder avbryts öppningssignalen varvid ventilerna stänger då följande tryck i reaktorn underskrids

Stängningstryck (bar)	Ventil
76	V14, V15 och V17
73	V16 och V18
71	V19 och V20

Ovanstående ventiler erhåller ny stängningsimpuls vid ca 67 bar.
Om någon av ventilerna förblir öppen erhålls larm vid ca 60 bar.

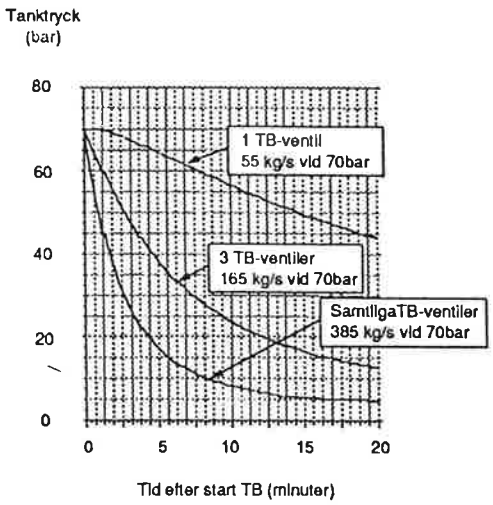
Tryckstyrd 314-blåsning:

Tryckstyrd 314-blåsning sker vid ca 82 bar och 85 bar i reaktortanken. Vid 82 bar öppnar basängblåsningsventilernaV14 - V20 och vid 85 bar öppnar direktblåsningsventilerna V1 - V13. Ventilerna stänger då reaktortrycket sjunkit ca 6 % underrespektive ventils öppningstryck.

Tvångsblåsning:

Automatisk tvångsblåsning erhålls då L3 och L4 föreligger i reaktortanken. Vid tvångsblåsning öppnar samtliga bassängblåsningsventiler.

Tryckförlopp vid tvångs-blåsning (TB).



Tvångsblåsning har antagits ske omedelbart i samband med avställning ingen spädmatning antas ske av reaktorn.
Detta innebär att den verkliga tryckstörkningen vid TB blir snabbare än vad som redovisas i ovanstående figur.

Rumsvakternas placering vid olika I-, Y-, A- och M-villkor

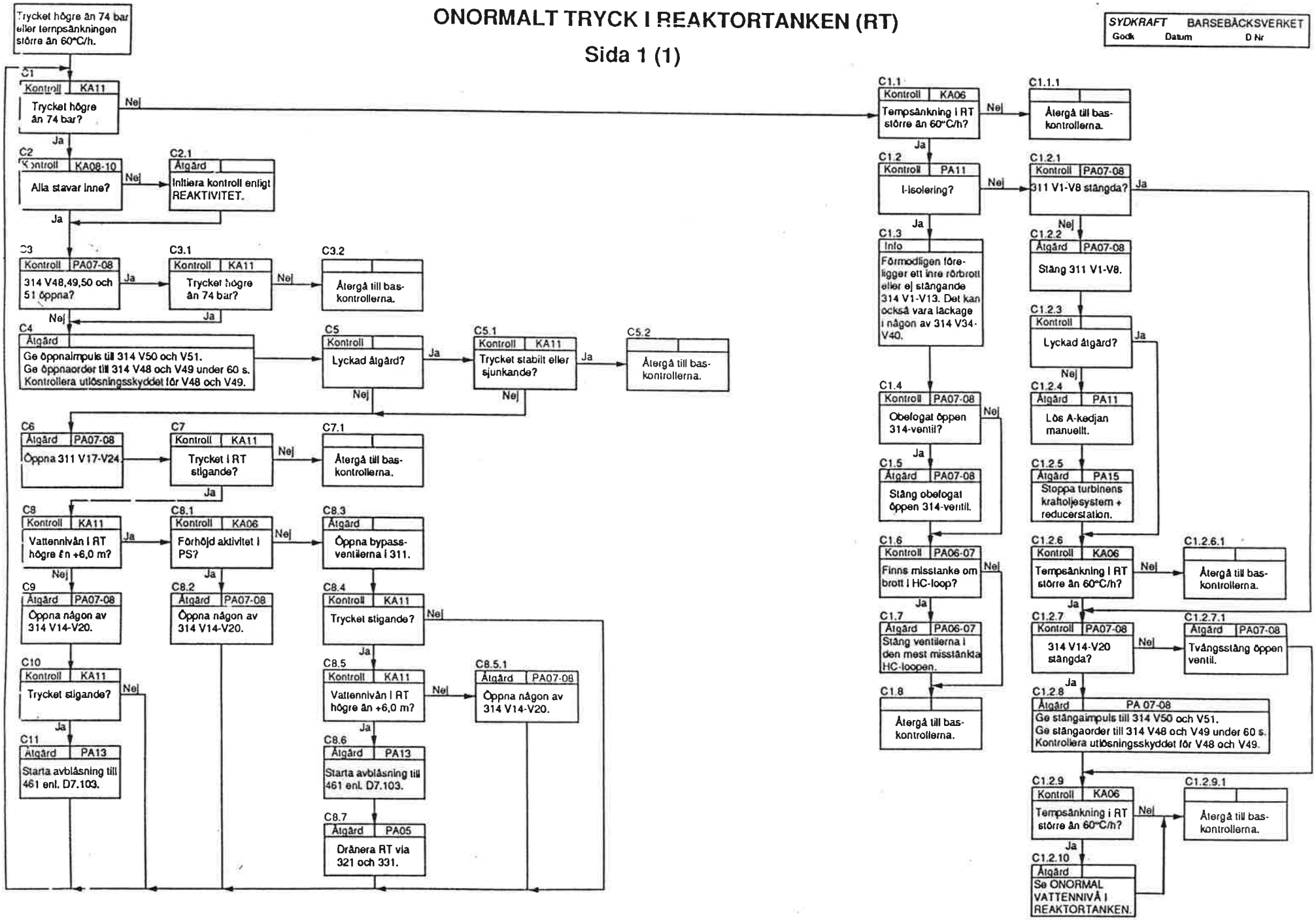
I3	546 K101 - K103	R4.60	311, 313
I4	546 K105 - K107	R1.60	311, 313
I5	546 K501 - K503	R4.60	311, 313
I6	546 K505 - K507	R4.60	311, 313
I7	546 K509 - K511	R1.60	311, 313
Y3	546 K109 - K111	R1.31	321
Y4	546 K113 - K115	R1.32	321
Y5	546 K117 - K119	R4.31	321
Y6	546 K121 - K123	R6.13	354
Y7	546 K125 - K127	R6.31	331, 354
Y8	546 K129 - K131	R1.40	321, 327, 331, 354
Y9	546 K133 - K135	R1.57	321, 327, 331, 354
Y10	546 K137 - K139	R8.19	331
Y11	546 K513 - K515	R1.31	321
Y12	546 K517 - K519	R1.32	321
Y13	546 K521 - K523	R4.53	331
Y14	546 K525 - K527	R5.31	331
Y15	546 K529 - K531	R6.13	354
Y16	546 K533 - K535	R6.31	331, 354
Y17	546 K537 - K539	R1.40	321, 327, 331, 354
Y18	546 K541 - K543	R1.57	321, 327, 331, 354
Y20	211 K406(L3), K408(L3), K411(L3)		
Y21	546 K401 - K403	R1.31	321
Y22	546 K405 - K407	R1.32	321
Y23	546 K409 - K411	R1.47	321, 327, 331, 354
Y25	546 K417 - K419	R4.31	321 (331)
Y26	546 K421 - K423	R5.31	331
Y29	546 K441 - K443	T9405	327
Y30	546 K445 - K447	T9608	327
Y32	546 K461 - K463	T9905	327
Y35	546 K853 - K855	T0316	327
Y36	546 K905 - K907	R1.47	321, 327,331, 354

A3	546 K141 - K143	T9401	311
A4	546 K145 - K147	T9912	311
A5	546 K149 - K151	T9921	312
A6	546 K153 - K155	T9922	312
A7	546 K545 - K547	R7.17	311
A8	546 K549 - K551	R2.17	311
A9	546 K553 - K555	T0308	311
A10	546 K557 - K559	T1404	311
A11	546 K561 - K563	T9921	312
A12	546 K565 - K567	T9922	312
A13	546 K161 - K163	T1404	311
A14	Utlöst kanal I M-kedjan		
M3	546 K449 - K451	T9801	462
M4	546 K453 - K455	T9802	462
M5	546 K437 - K439	T9404	462
M6	546 K489 - K491	T9915	462
M7	546 K473 - K475	T9912	451, 462
M8	546 K465 - K467	T9910	462
M9	546 K825 - K827	T0003	462
M10	546 K477 - K479	T9912	451, 462
M11	546 K481 - K483	T9912	451, 462
M12	546 K485 - K487	T9914	462
M13	546 K909 - K911	T9926	462
M14	546 K805 - K807	T9919	332, 462
M15	546 K857 - K859	T0402	332
M16	546 K869 - K871	T0705	332
M17	546 K873 - K875	T0706	332
M18	546 K877 - K879	T0707	332
M19	546 K881 - K883	T0708	332
M20	546 K885 - K887	T0709	332
M21	546 K889 - K891	T0710	332
M22	546 K817 - K819	T9923	462
M23	546 K821 - K823	T9924	462
M24	546 K901 - K903	T1401	312
M25	546 K921 - K923	T1401	312
M26	546 K917 - K919	T1401	312
M27	546 K913 - K915	T1402	312
M28	546 K861 - K863	T0601	453, 462
M29	546 K809 - K811	T9921	312
M30	546 K813 - K815	T9922	312
M31	546 K425 - K427	R6.16	312
M32	546 K429 - K431	R6.18	312

ONORMALT TRYCK I REAKTORTANKEN (RT)

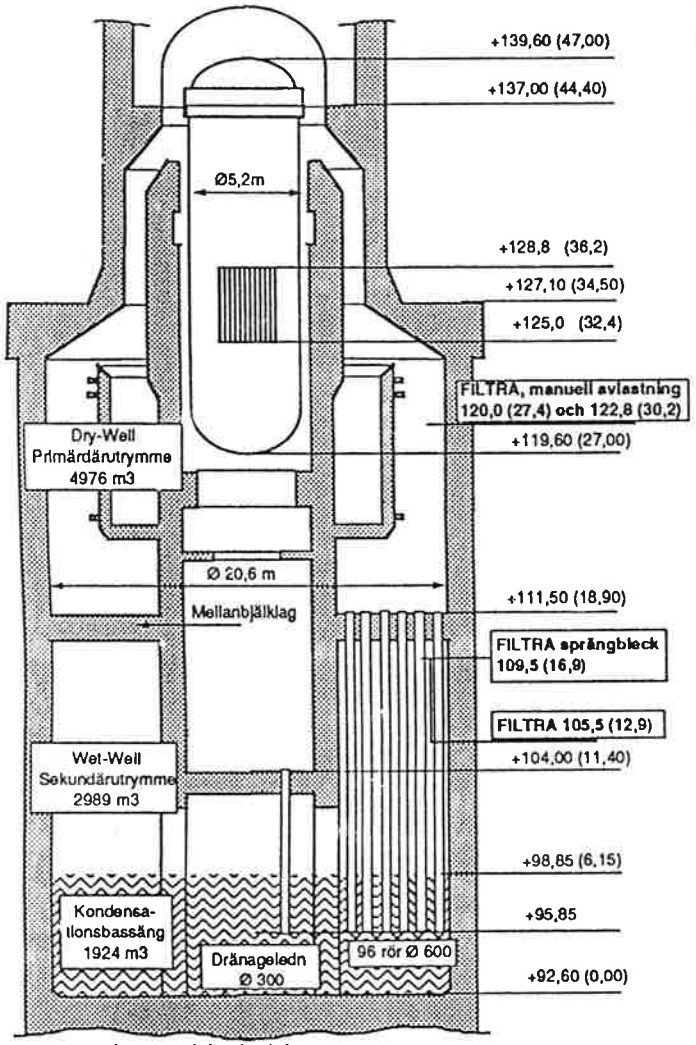
Sida 1 (1)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr



ONORMALT TRYCK I REAKTORINNESLUTNINGEN (PS)

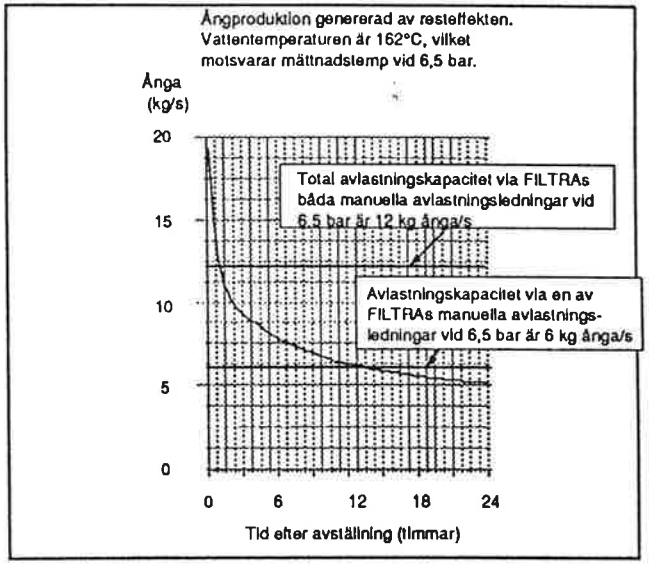
Reaktorinneslutning



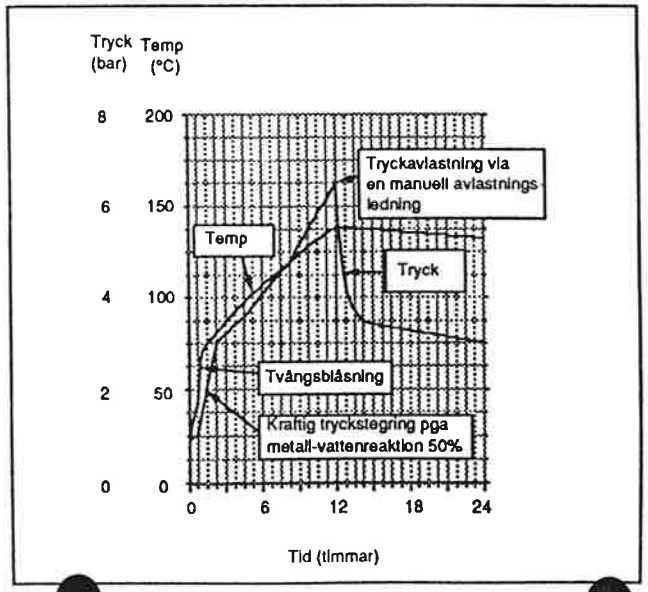
Automatiska funktioner

I3	Tryckstegring i drywell > 10 mbar (ändringshastighet >1 mbar/s)
I4	Tryckstegring i wetwell > 10 mbar (ändringshastighet >1 mbar/s)
I5	Temperatur i drywell > 80°C
I6	Temperatur i drywell > 80°C
I7	Temperatur i wetwell > 80°C

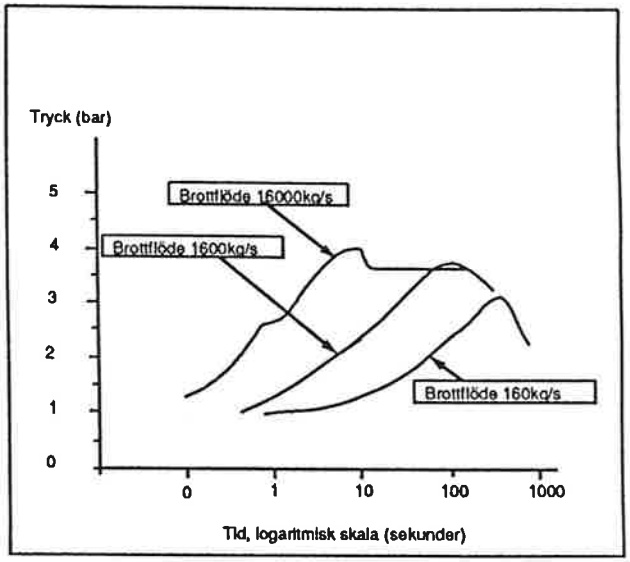
Resteffekt



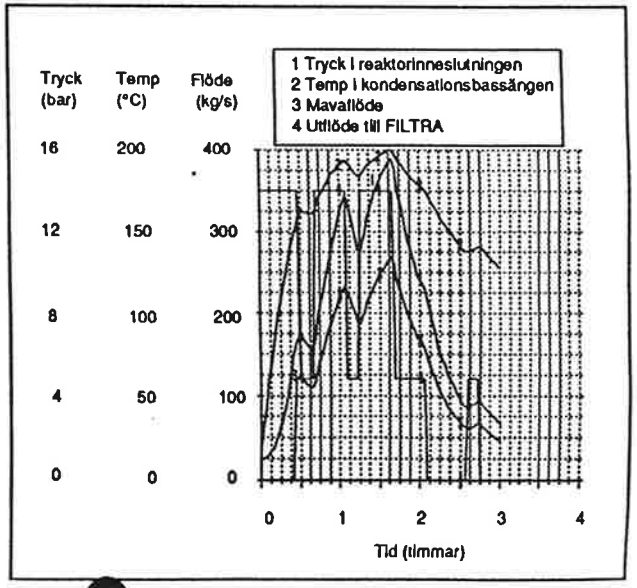
Tryck- och temperaturförlopp vid transient och helt utebliven spädmatning av reaktorn och helt utebliven resteffektkylning (Designhändelse för FILTRA)



Tryckförlopp vid LOCA

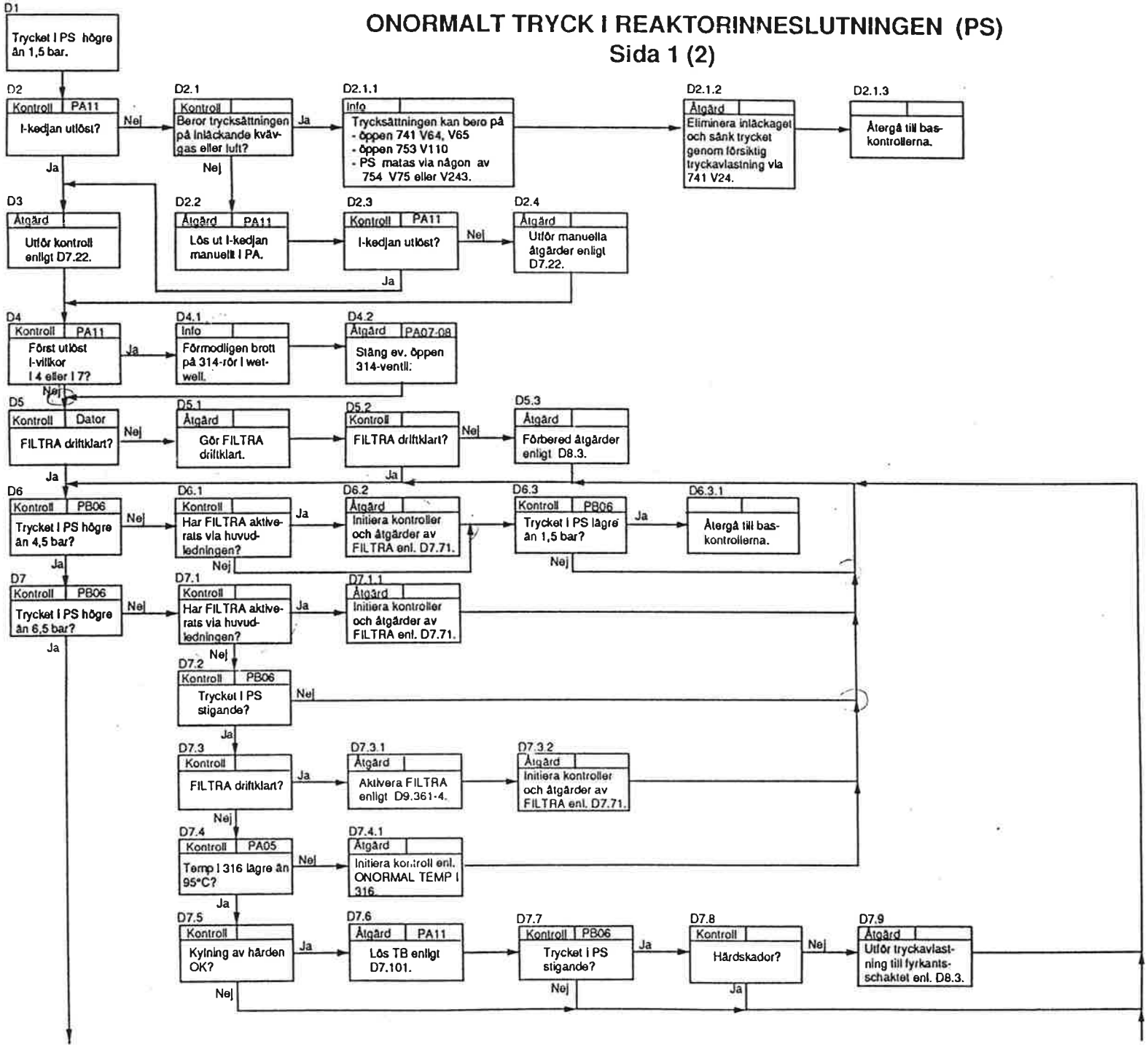


Tryckförlopp vid utebliven reaktoravställning



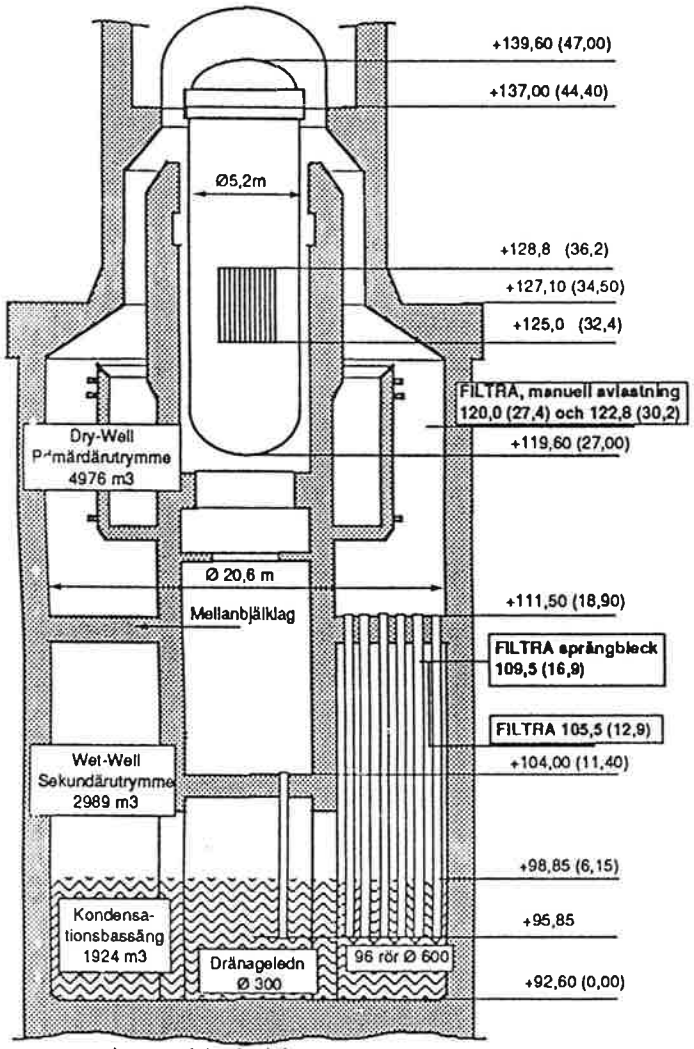
ONORMALT TRYCK I REAKTORINNESLUTNINGEN (PS)
Sida 1 (2)

SYDKRAFT	BARSEBÄCKSVERKET	
Godk	Datum	D Nr



ONORMALT TRYCK I REAKTORINNESLUTNINGEN (PS)

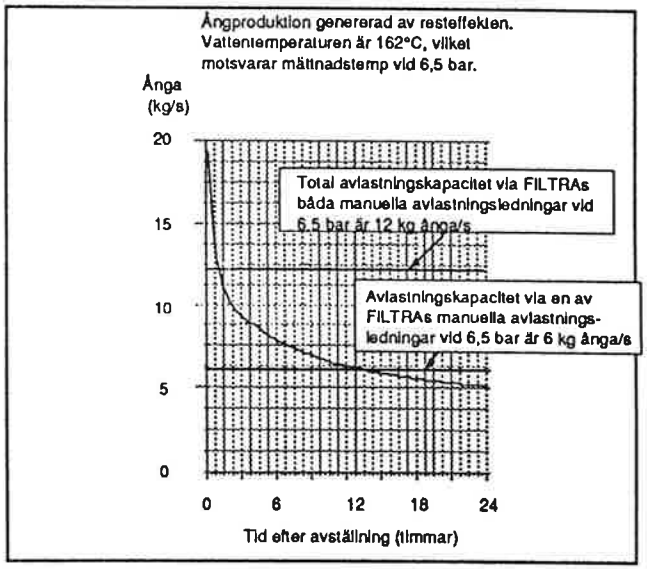
Reaktorinneslutning



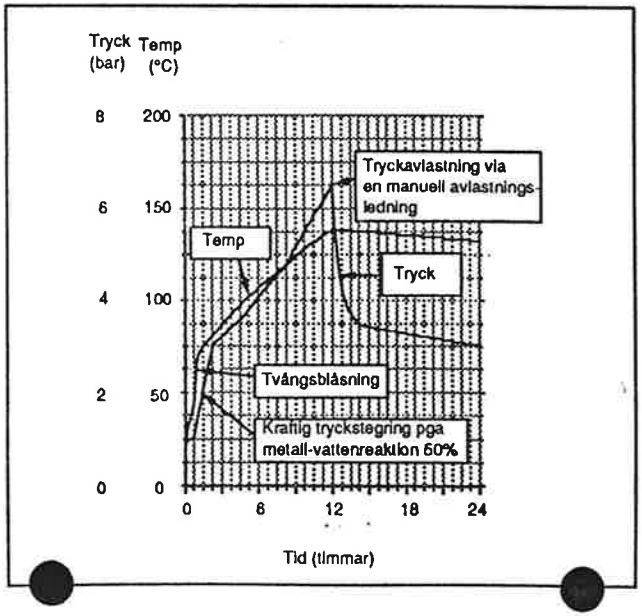
Automatiska funktioner

I3 Tryckstegring i drywell > 10 mbar (ändringshastighet >1 mbar/s)
I4 Tryckstegring i wetwell > 10 mbar (ändringshastighet >1 mbar/s)
I5 Temperatur i drywell > 80°C
I6 Temperatur i drywell > 80°C
I7 Temperatur i wetwell > 80°C

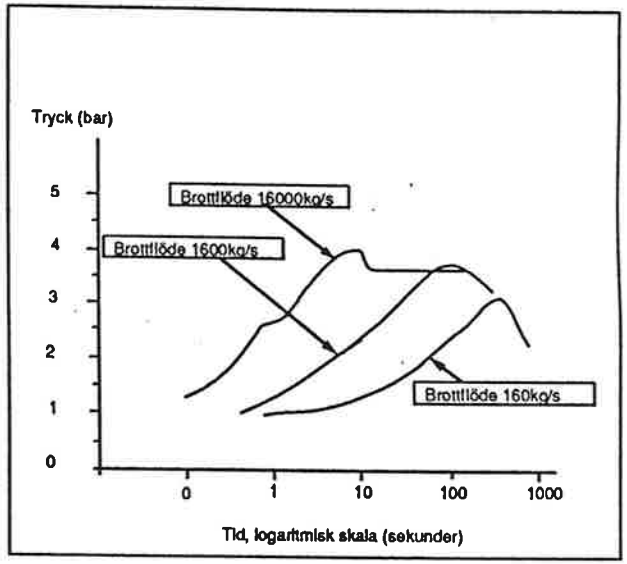
Resteffekt



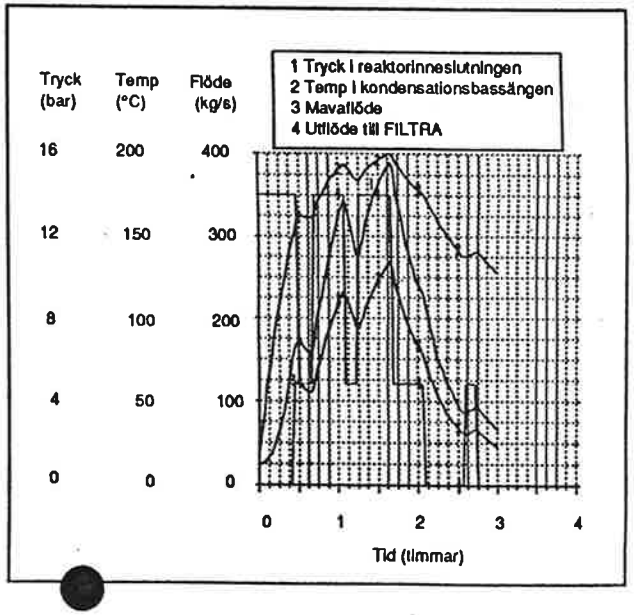
Tryck- och temperaturförlopp vid transient och helt utebliven spädmatning av reaktorn och helt utebliven resteffektkylning (Designhändelse för FILTRA)



Tryckförlopp vid LOCA

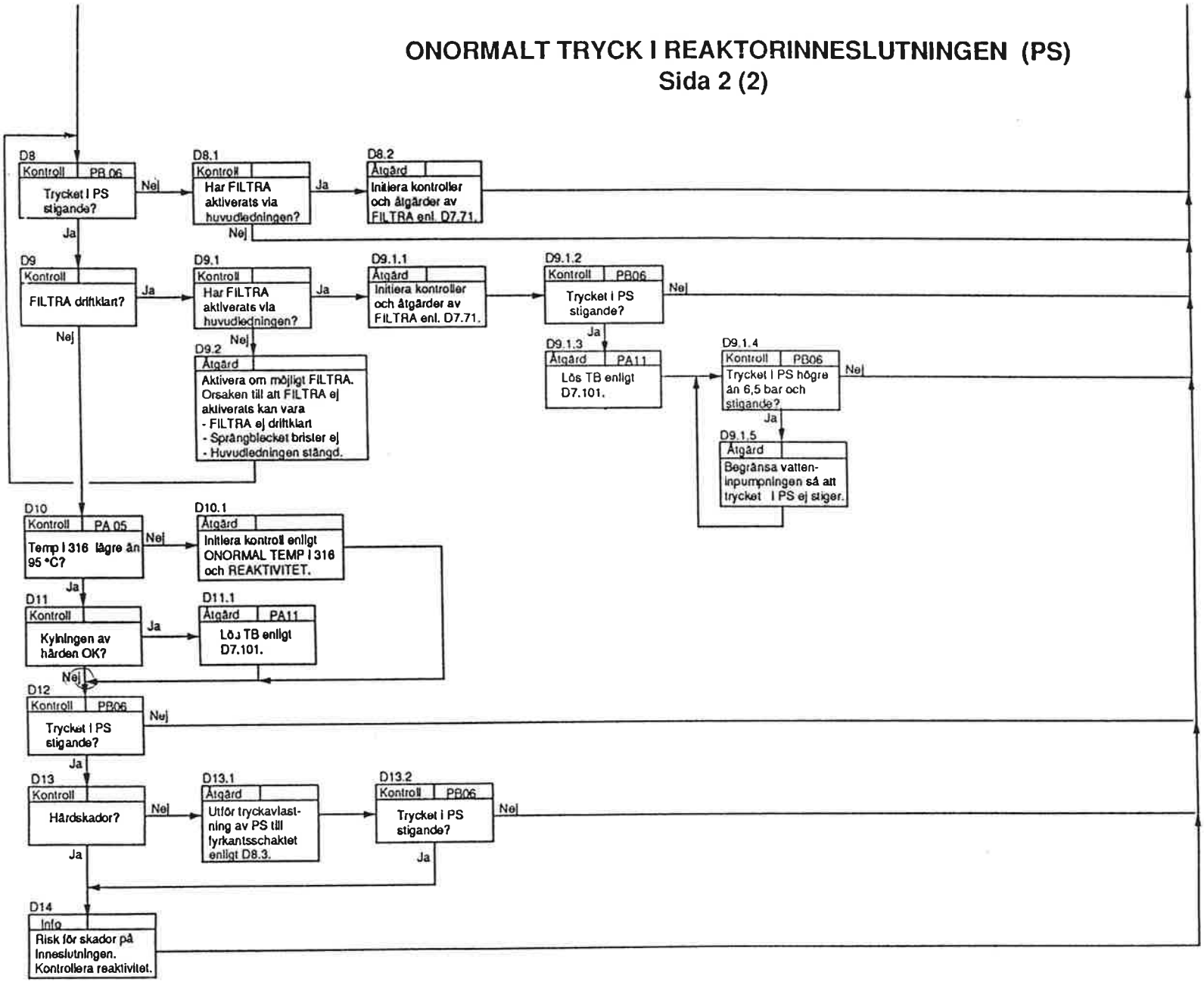


Tryckförlopp vid utebliven reaktoravställning



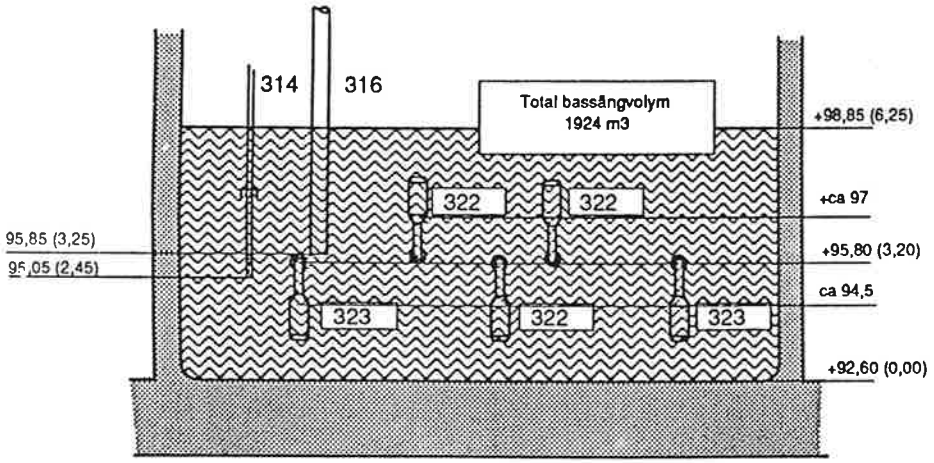
ONORMALT TRYCK I REAKTORINNESLUTNINGEN (PS)
Sida 2 (2)

SYDKRAFT	BARSEBÄCKSVERKET	
Godk	Datum	D Nr

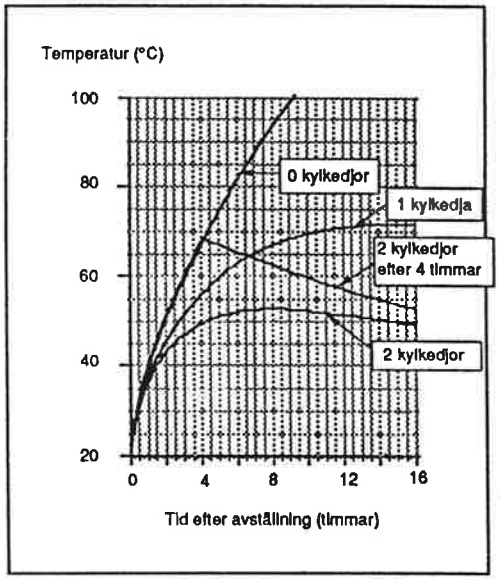


ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)

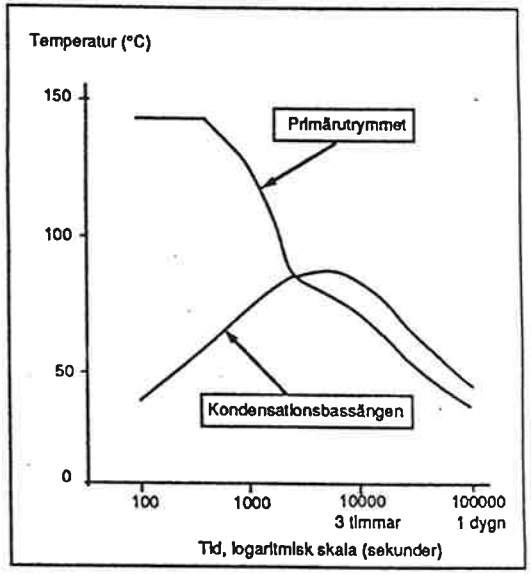
Kondensationsbassängen



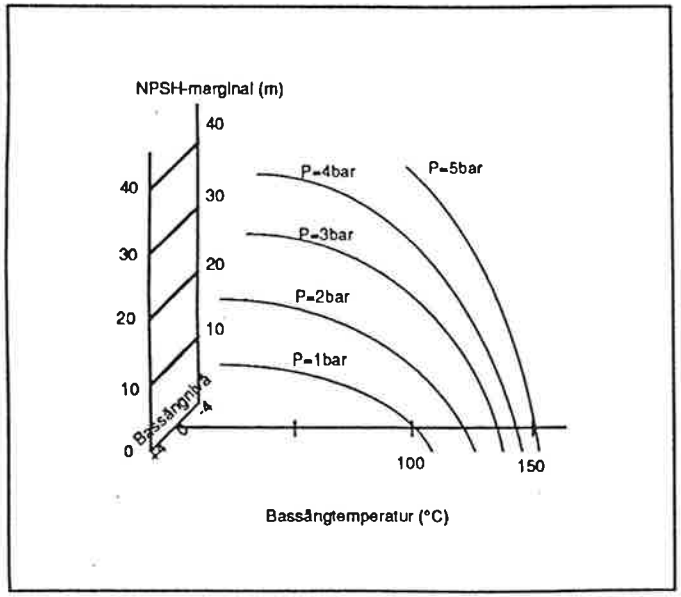
Temperaturförlopp s f a antal kylkedjor i drift vid varm avställning med dumpförbud.



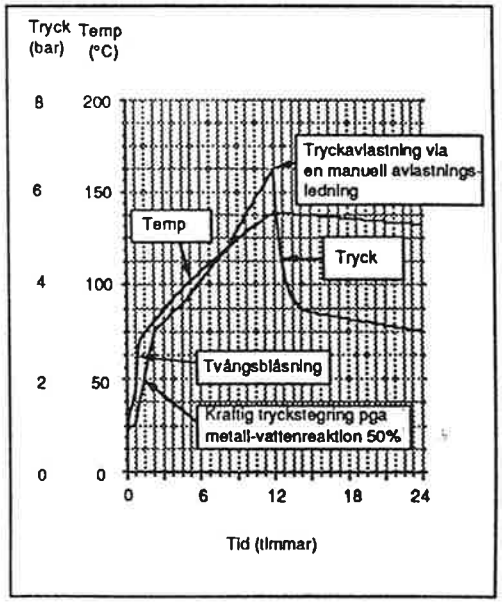
Temperaturförlopp vid LOCA



NPSH-marginal (kavitationsmarginal) för 323-pumparna s f a temperatur i kondbass, vattennivå i kondbass samt tryck i PS.



Tryck- och temperaturförlopp vid transient och helt utebliven spädmatning av reaktorn och helt utebliven resteffektkylning (Designhändelse för FILTRA)

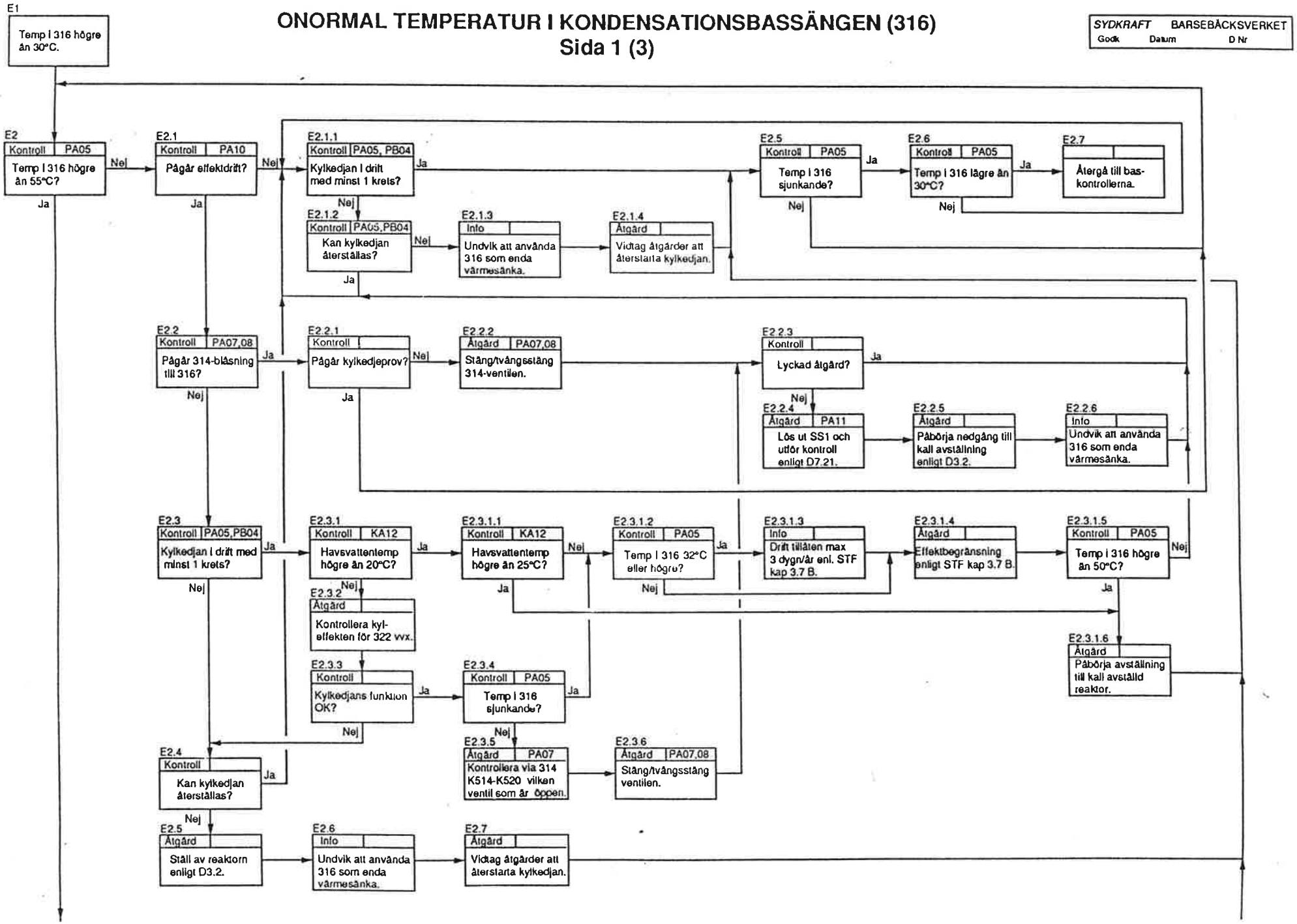


Automatiska funktioner

316 K508 ger start kylkedjorna 322/721/712.

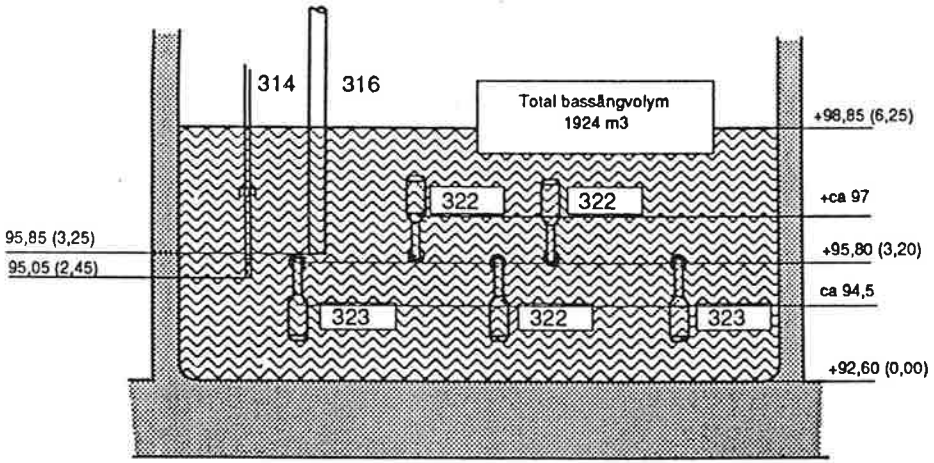
ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)
Sida 1 (3)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr

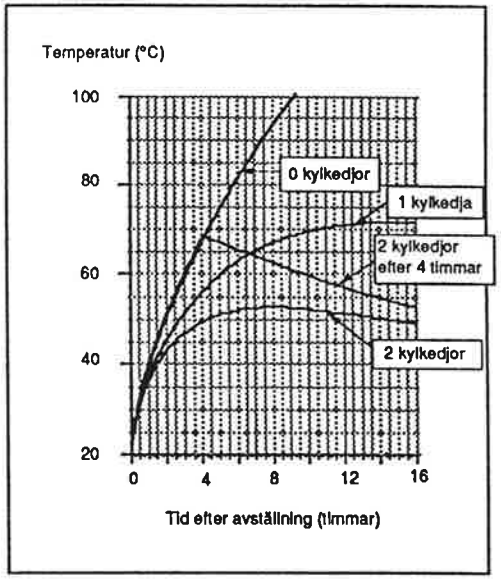


ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)

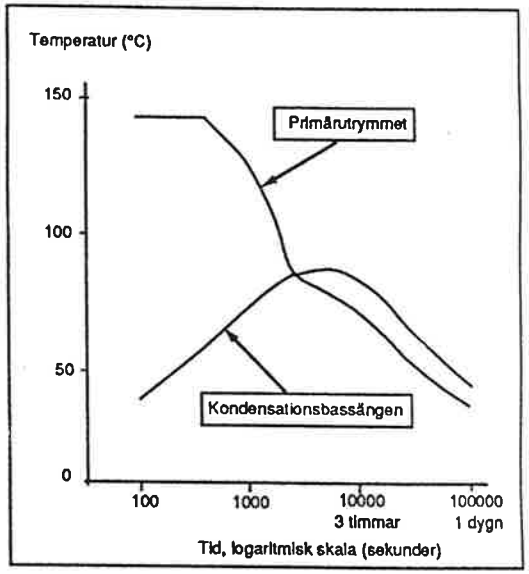
Kondensationsbassängen



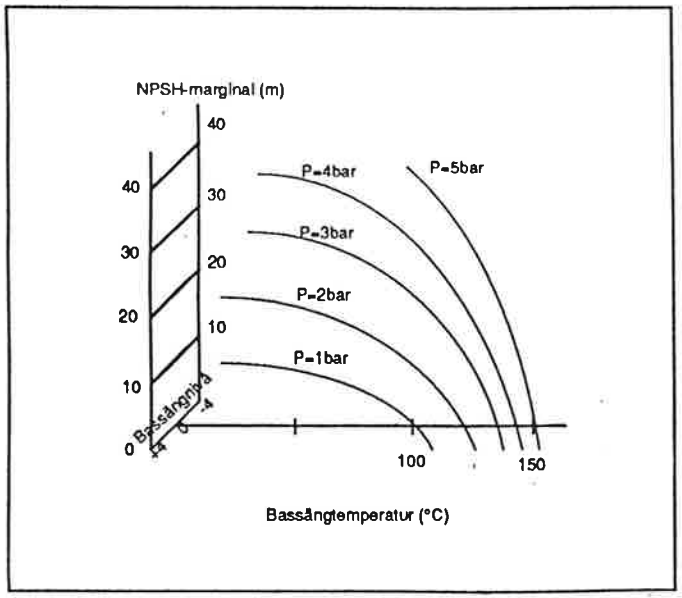
Temperaturförlopp s f a antal kylkedjor i drift vid varm avställning med dumpförbud.



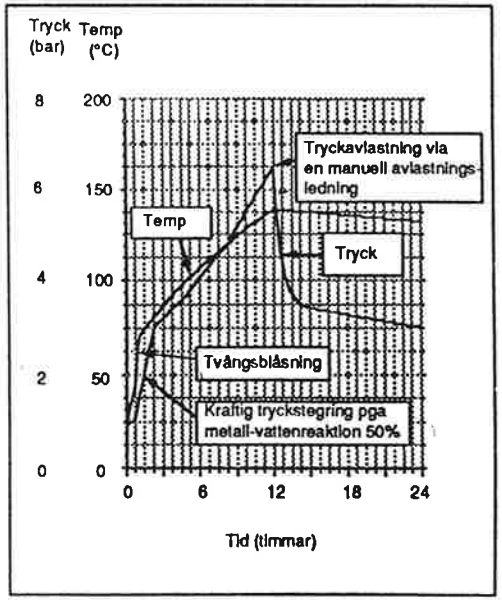
Temperaturförlopp vid LOCA



NPSH-marginal (kavitationsmarginal) för 323-pumparna s f a temperatur i kondbass, vattennivå i kondbass samt tryck i PS.



Tryck- och temperaturförlopp vid translent och helt utebliven spädmatning av reaktorn och helt utebliven resteffektkylning (Designhändelse för FILTRA)

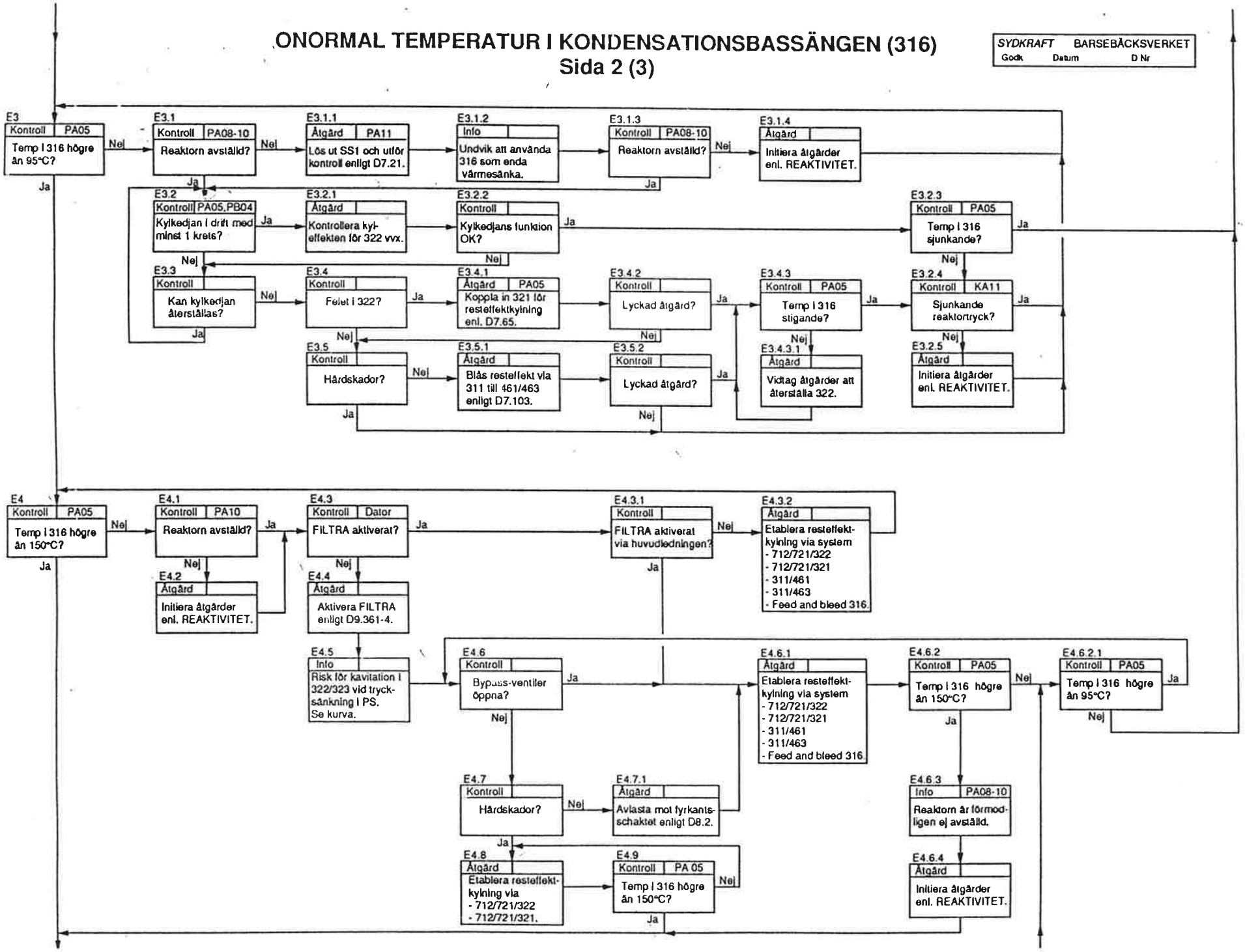


Automatiska funktioner

316 K508 ger start kylkedjorna 322/721/712.

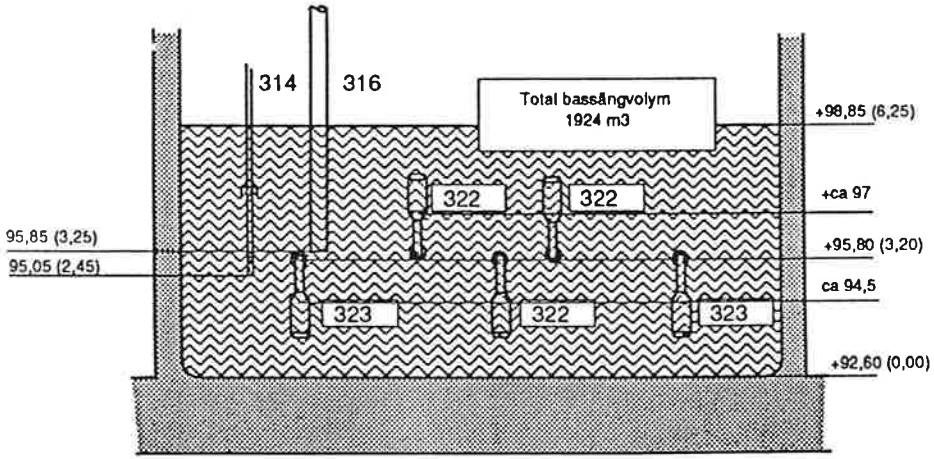
ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)
Sida 2 (3)

SYDKRAFT	BARSEBÄCKSVERKET	
Godk	Datum	D Nr

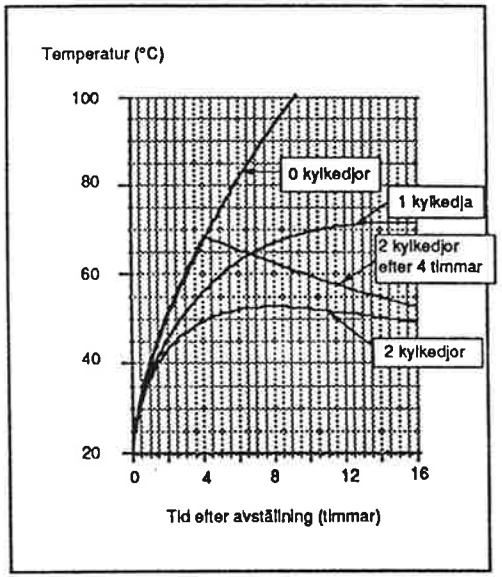


ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)

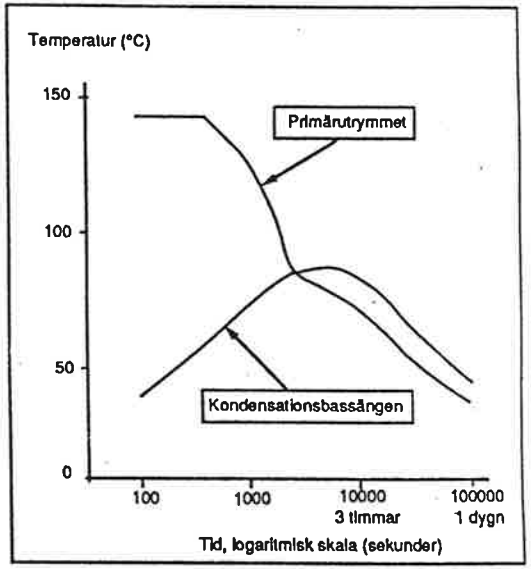
Kondensationsbassängen



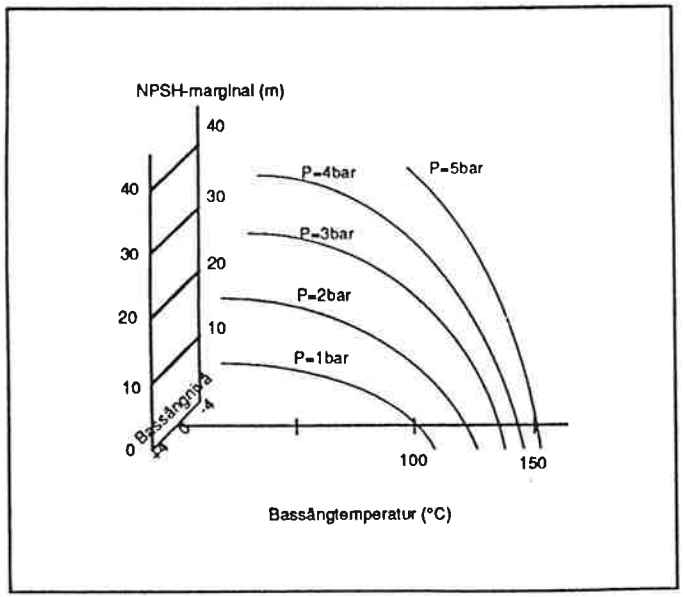
Temperaturförlopp s f a antal kylkedjor i drift vid varm avställning med dumpförbud.



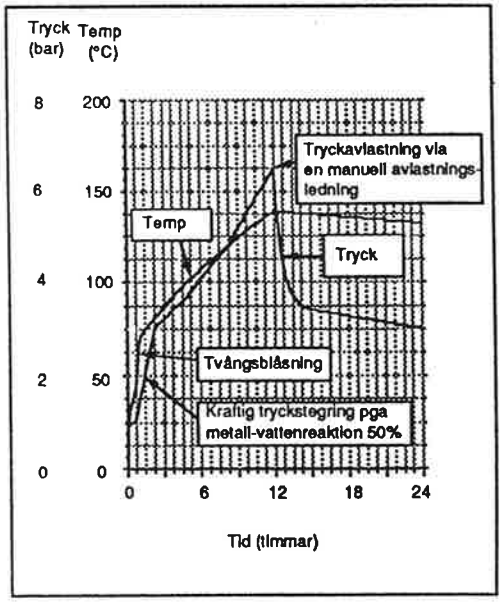
Temperaturförlopp vid LOCA



NPSH-marginal (kavitationsmarginal) för 323-pumparna s f a temperatur i kondbass, vattennivå i kondbass samt tryck i PS.



Tryck- och temperaturförlopp vid transient och helt utebliven spädmatning av reaktorn och helt utebliven resteffektkylning (Designhändelse för FILTRA)

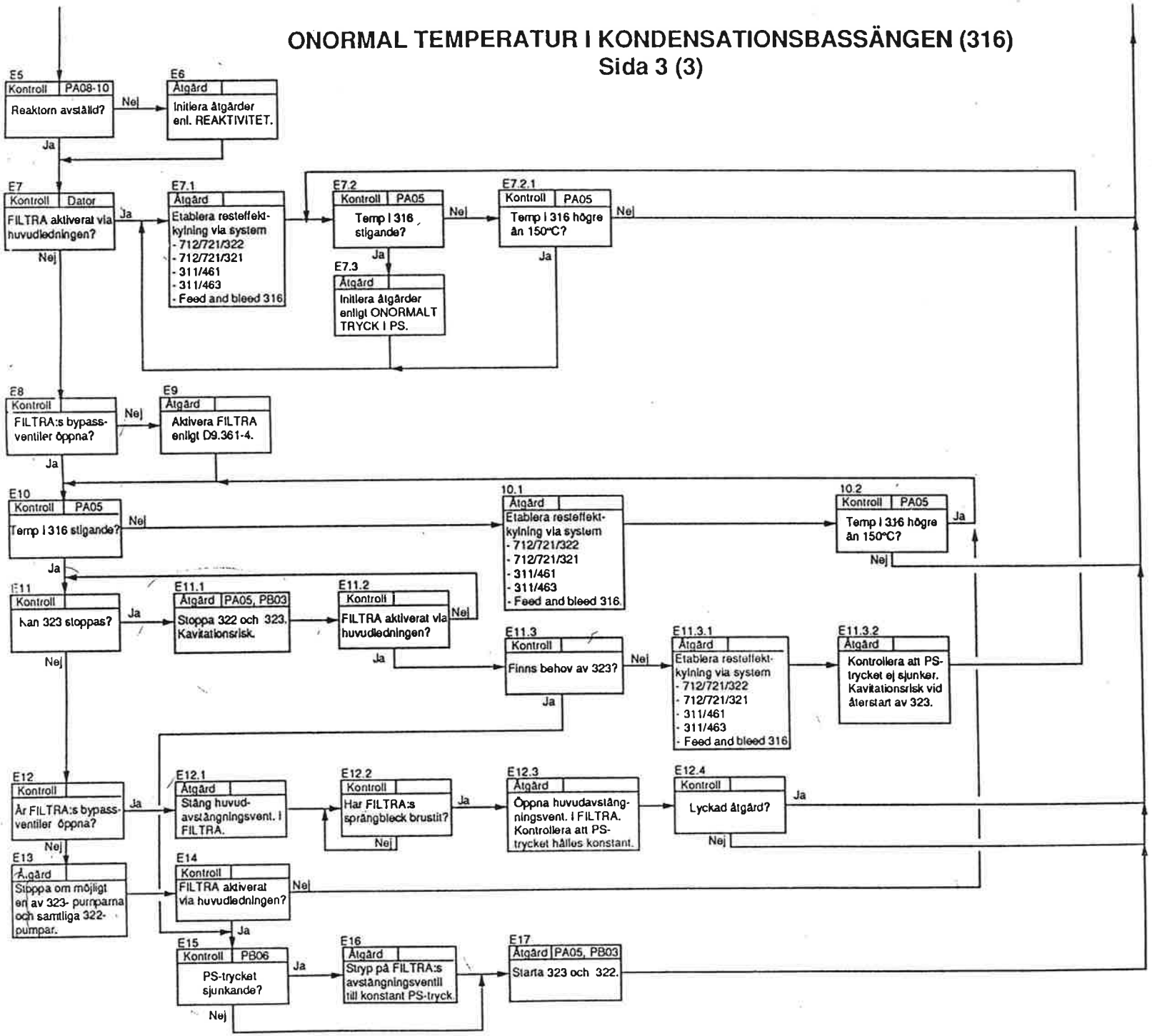


Automatiska funktioner

316 K508 ger start kylkedjorna 322/721/712.

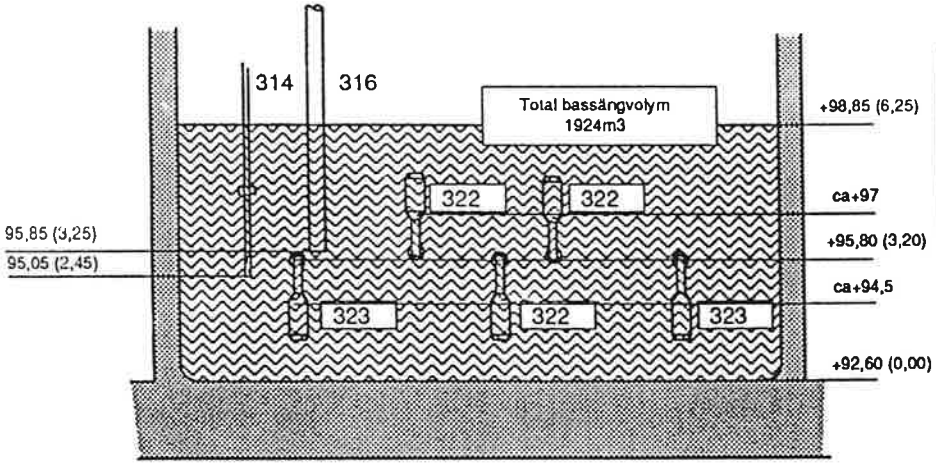
ONORMAL TEMPERATUR I KONDENSATIONSBASSÄNGEN (316)
Sida 3 (3)

SYDKRAFT	BARSEBÄCKSVERKET
Godk	Datum D Nr

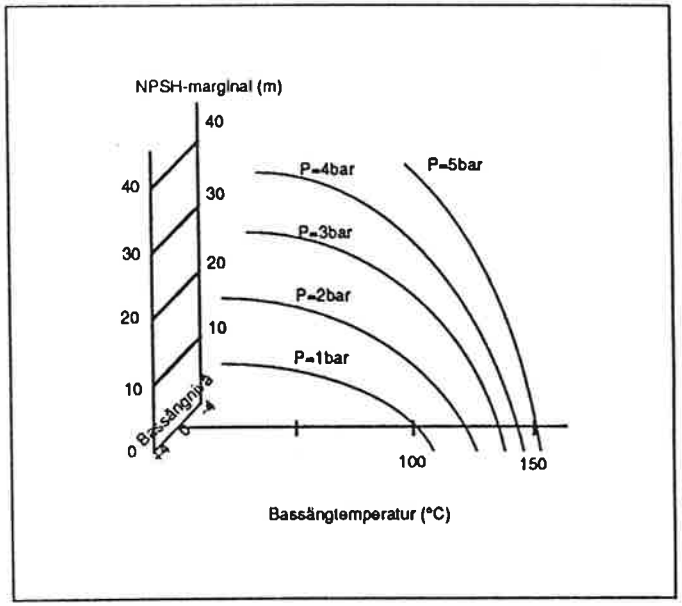


ONORMALT LÅG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)

Kondensationsbassängen



NPSH-marginal (kavitationsmarginal) för 323-pumparna s f a temperatur i kondbass, vattennivå i kondbass samt tryck i PS.

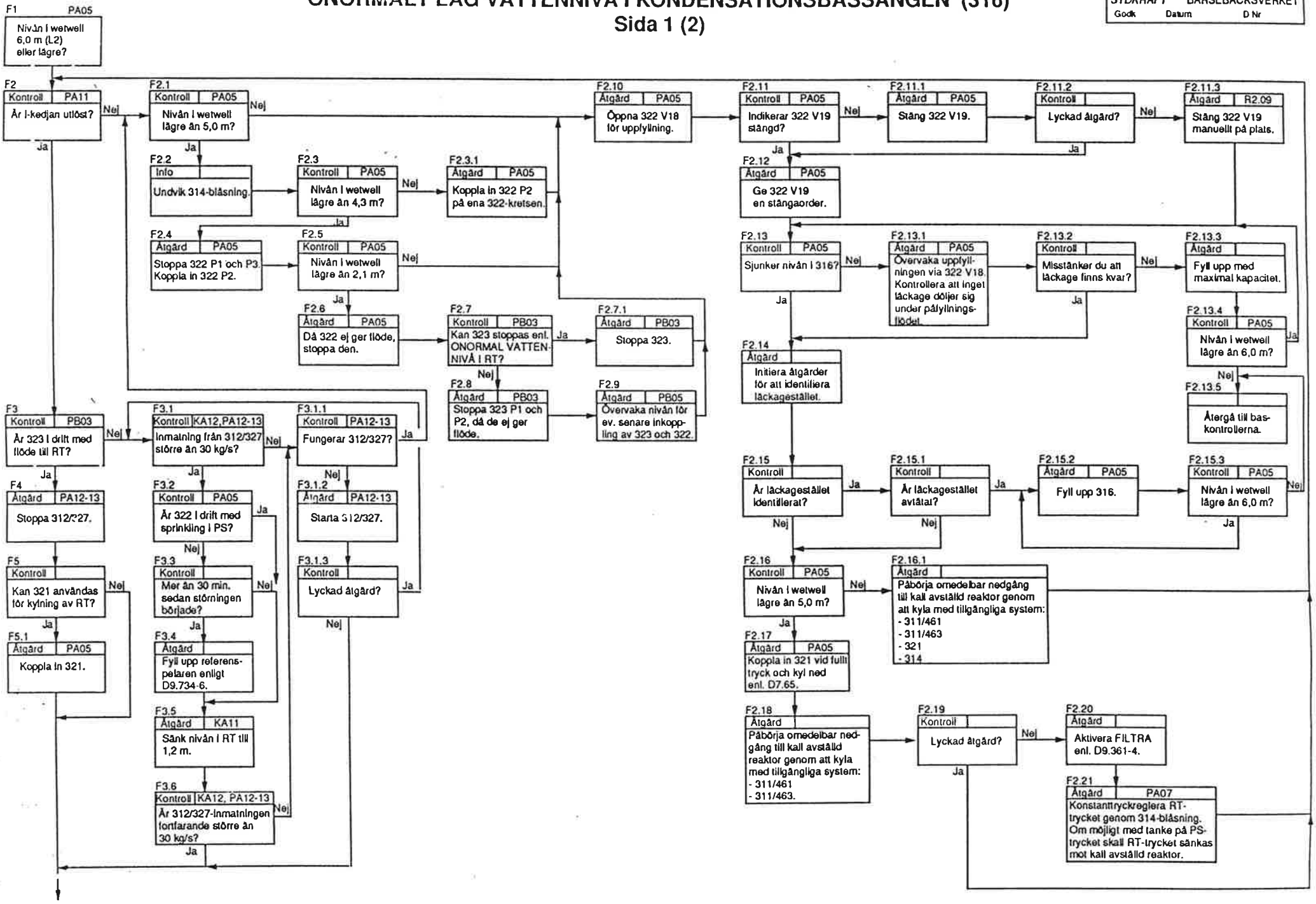


Automatiska funktioner

L1 stopp utpumpning av kondensationsbassängen genom stängning 322 V19.

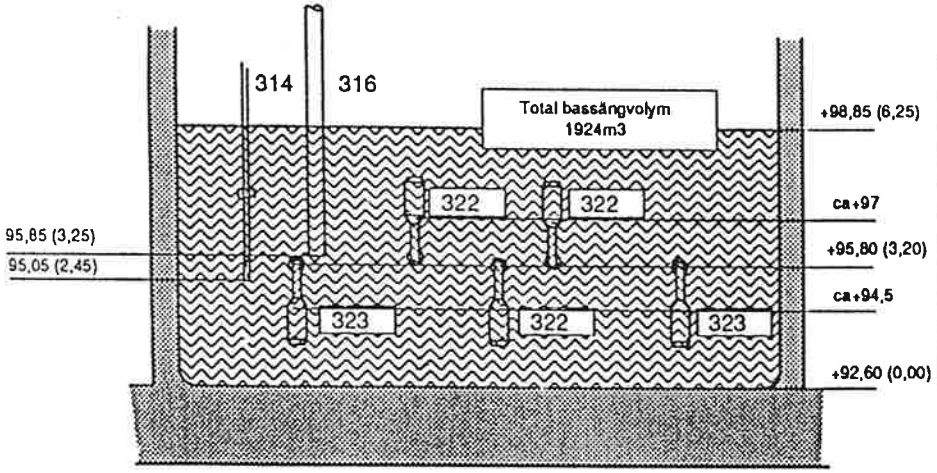
ONORMALT LÅG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)
Sida 1 (2)

SYDKRAFT	BARSEBÄCKSVERKET	
Godk	Datum	D Nr

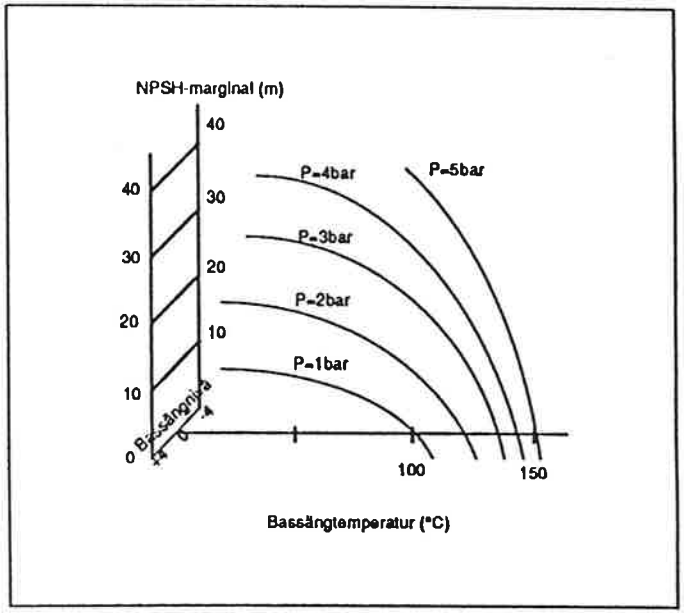


ONORMALT LÅG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)

Kondensationsbassängen



NPSH-marginal (kavitationsmarginal) för 323-pumparna s f a temperatur i kondbass, vattennivå i kondbass samt tryck i PS.

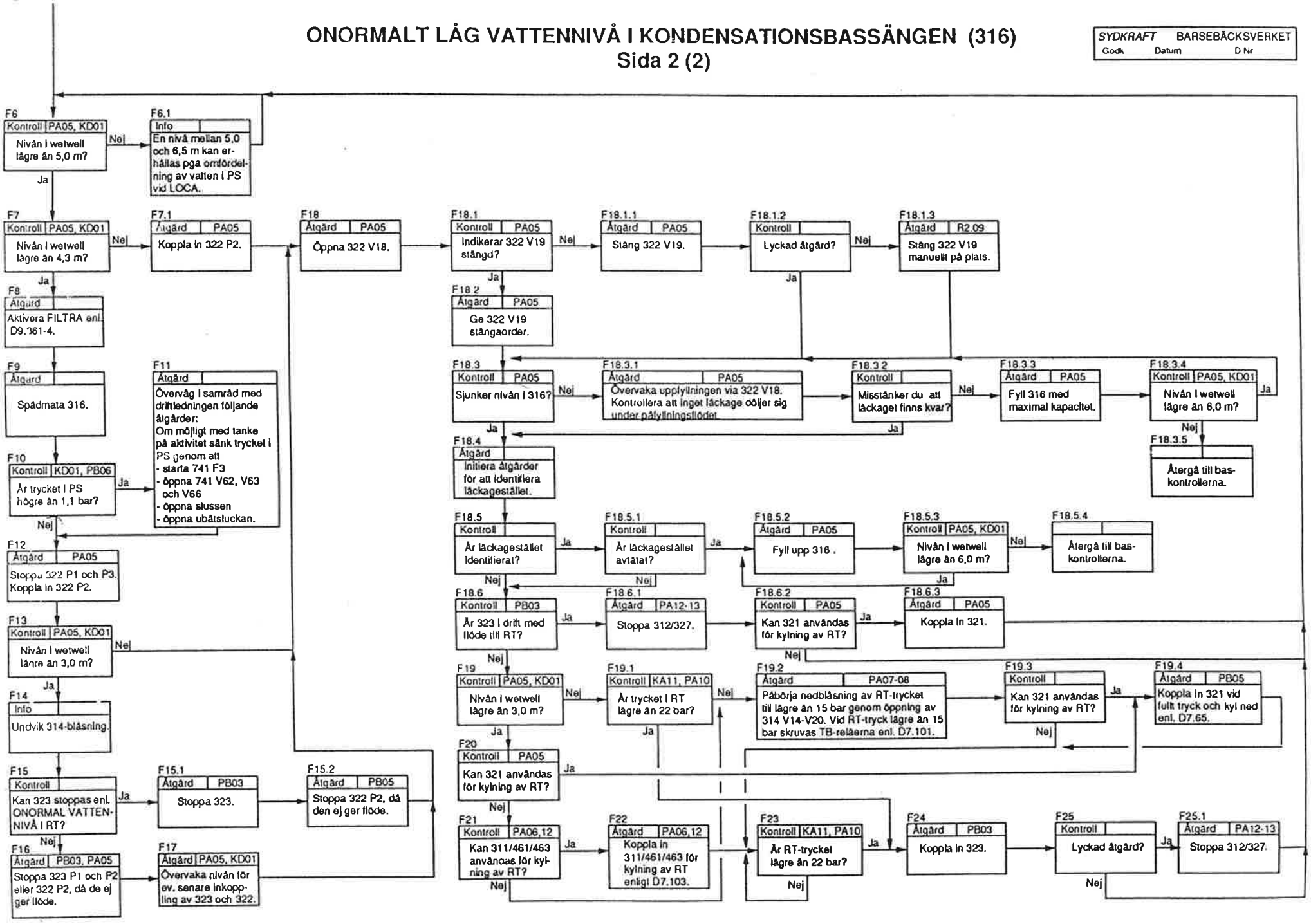


Automatiska funktioner

L1 stopp utpumpning av kondensationsbassängen genom stängning 322 V19.

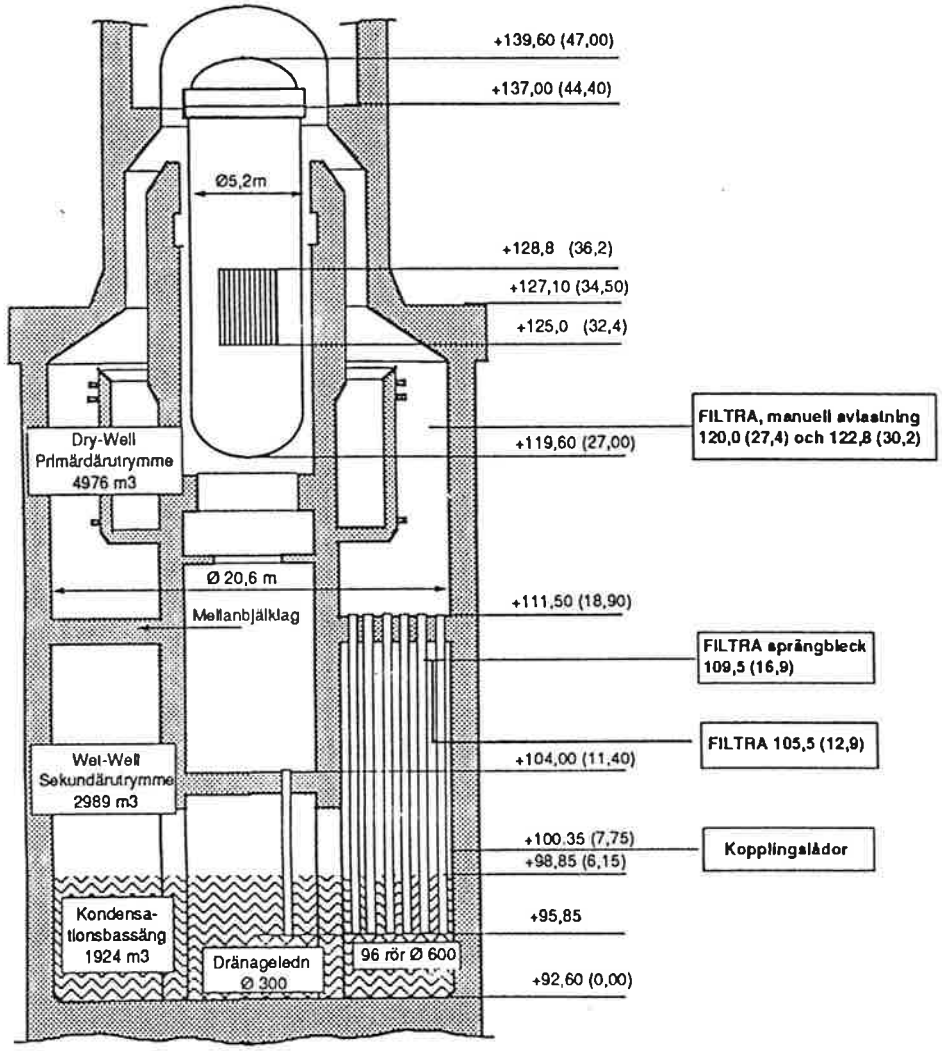
ONORMALT LÅG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)
Sida 2 (2)

SYDKRAFT	BARSEBÄCKSVERKET
Godk	Datum D Nr

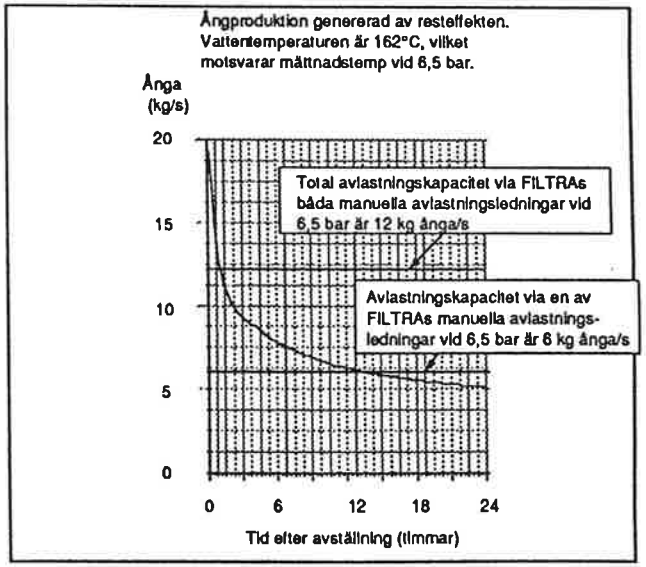


ONORMALT HÖG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)

Reaktorinneslutning

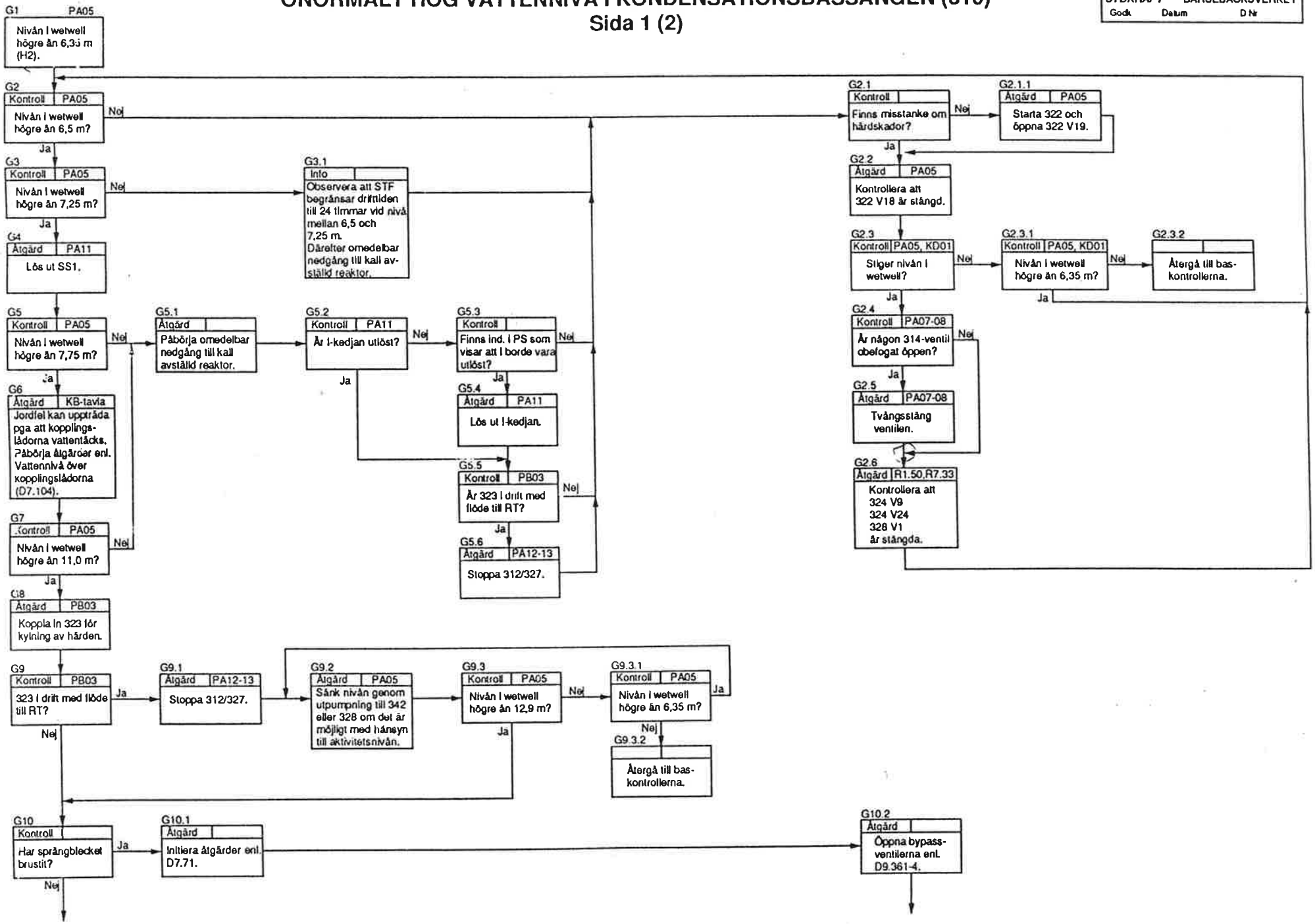


Resteffekt



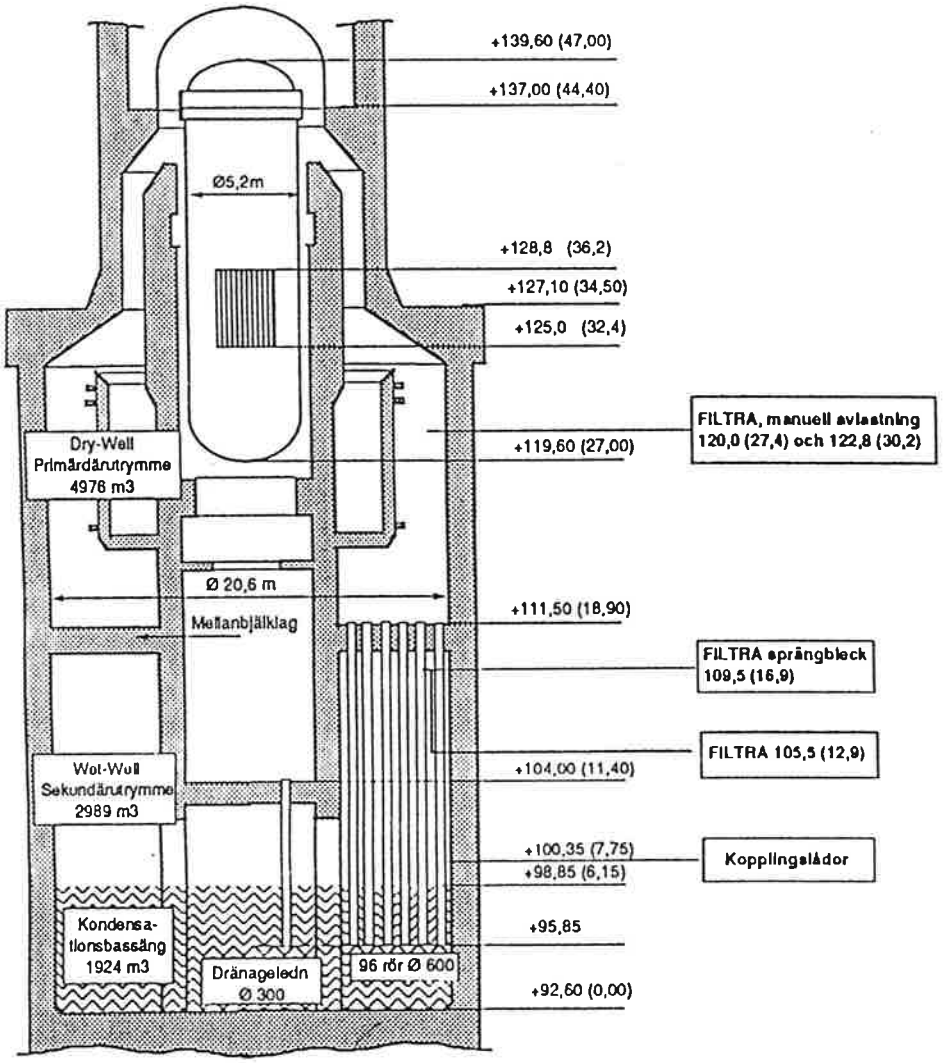
ONORMALT HÖG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)
Sida 1 (2)

SYDKRAFT	BARSEBÄCKSVERKET
Godk	Datum / D Nr

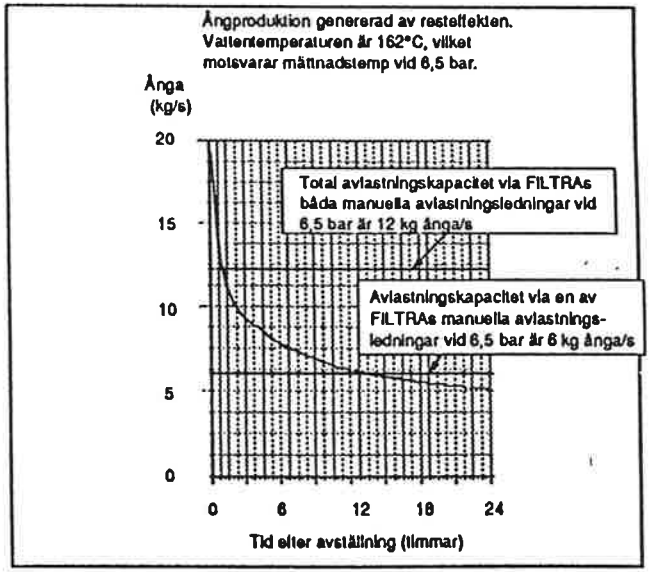


ONORMALT HÖG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)

Reaktorinneslutning

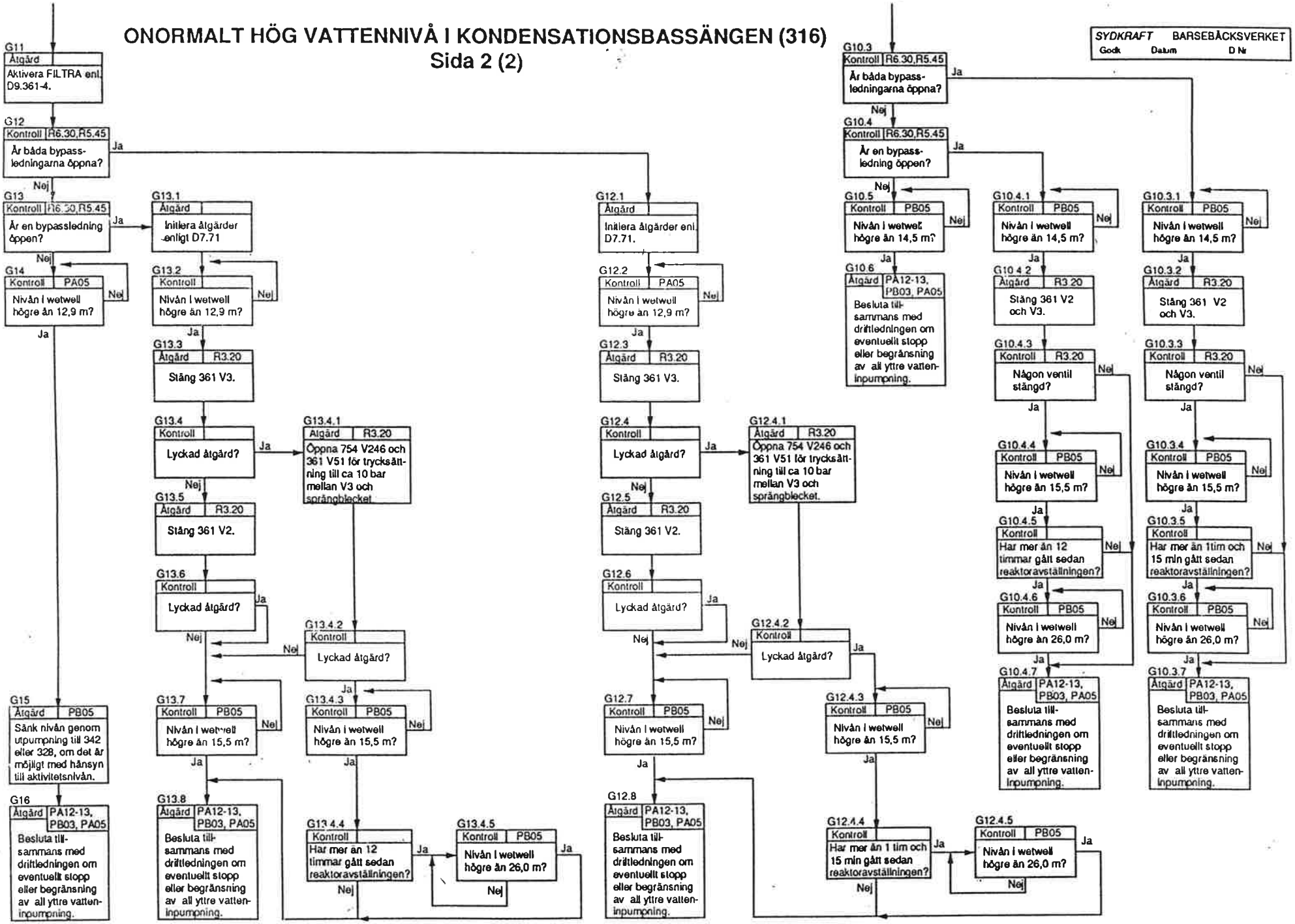


Resteffekt



ONORMALT HÖG VATTENNIVÅ I KONDENSATIONSBASSÄNGEN (316)
Sida 2 (2)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr



DIFFUST LÄCKAGE
Sida 1 (1)

SYDKRAFT BARSEBÄCKSVERKET
Godk Datum D Nr