CODEN:LUTEDX/(TEIE-5152)/1-67/(2001)

Industrial Electrical Engineering and Automation

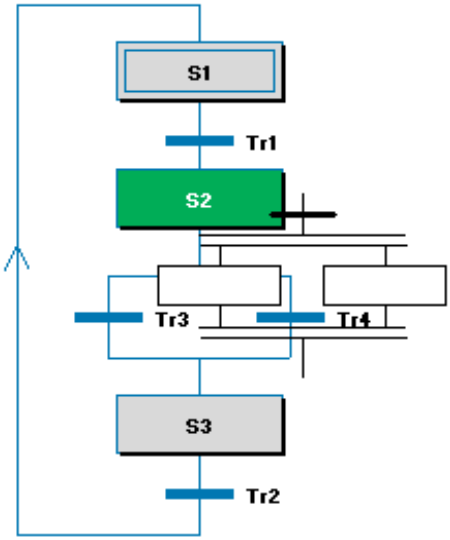
Sequential Function Chart Interfacing

Mattias Nilsson
Kristoffer Persson

Department of Industrial Electrical Engineering and Automation
Lund University

Master Thesis in Industrial automation
at Lund Institute of Technology

Sequential Function Chart Interfacing



Authors: Mattias Nilsson M96 Lund, Sweden 2001-04-17
Kristoffer Persson E96

Examiner: Professor Gustaf Olsson

Supervisor : Håkan Augustsson, ABB Automation Products AB

Preface

We would like to thank our supervisor at Lund Institute of Technology professor Gustaf Olsson for critical review and guidelines to the report. The master thesis has been performed at ABB Automation Products in Malmö. We would like to thank following people at ABB for ideas and help: our supervisor at ABB: Håkan Augustsson, Ulf Andersson, Ulf Hagberg and Andreas Hellström.

Summary

Kristoffer Persson and Mattias Nilsson made this thesis work for ABB Automation in Malmö. Our supervisor at ABB has been Håkan Augustsson and the examiner is Gustaf Olsson, Professor at the Department of Industrial Electrical Engineering and Automation (IEA) at LTH.

The task was to improve the SFC user interface of ABB:s product Advant Control Builder.

Advant Control Builder is a program for industrial control, the SFC interface is used to be able to construct an automation-program to control for example an industrial process. Alternative programming languages is function block diagram (FBD), Ladder (LD) or IL (Instruction List).

ACB is based on Visual C++ programming so this is the language we used in the thesis work. When neither of us have been using Visual C++ before the thesis work, we had to do some self-studying in this language.

The work began with a study in how the product works today, to get grip on which modifications and additions that we want to make.
We studied competitive programs and program in other working areas that uses similar graphical interface. This was made to get more ideas than our own ideas of how we wanted the new improved SFC interface to look like.

The next part of the thesis work was to put together the ideas into a function-specification.

The last and biggest part of the work was to implement the improved SFC interface. To be able to make this in the limited time we had we started out from the existing prototype program that we modified and added our functions to.
To add things to the program where we forced to learn the existing program, when this was made of a lot of files and that was programmed during a period of more than 15 years it sometimes where hard to find persons that had knowledge about this files.
When we had some knowledge we made modifications and added parts to the program and then we compiled and tested and then we modified again. This was repeated until we where satisfied with the function of the program.

The final result is not exactly the same as the one we specified in the function-specification, when you explore better and easier ways to program about the same things.

But totally our work has resulted in faster and user friendlier program to use for building SFC nets.

SEQUENTIAL FUNCTION CHART INTERFACING

Table of content

1 Introduction

A PLC is a microcomputer designed to work in industrial environments, it generates on/off signal outputs to control valves, electrical motors, lights etc. PLC stands for Programmable logical controller.
Small and simple PLCs is designed to replace relays and have some additional counter and timing functions. While more advanced can perform mathematical calculations and process analog signals and contains feedback control circuits (like e.g. PID controllers).

PLC:s is usually programmed via an external programming unit, for example a portable PC. The programming unit is not needed online and can be removed when the PLC is online. There is different program languages to program a PLC. Program languages can be simple assembler like or more advanced high-level languages.
When the programming is finished the program instructions is translated into machine code in a compilator.

This thesis work is about improvement of such high-level language named SFC. SFC is described in the IEC standard 848 and 6113-3.
We have worked with the SFC editor in the ABB program ACB. ACB is a Windows based program designed for programming of some of ABBs products.

1.1 The task

SFC stands for Sequential Function Chart and is a graphical programming language to describe sequential operations. SFC is closely related to Grafcet and it has some similarity with a Petri-net.

SFC is built up of steps with transition conditions in-between, where each step represents one state. When a step is active it means that the control program is in this state. When the succeeding transition conditions become fulfilled the control program execution continues in the next step (the token is moved). Then the succeeding step gets active and the previous active step becomes inactive. It is also possible to build parallel and alternative step-constructions. More about this in chapter 1.5

The task for this thesis work is to improve the SFC editing in ABB product Advant Control Builder.

In today's system one has to mark the place for insertion of the new element and then select by the menu what type of element to add. This makes it time-consuming and not very flexible to build a new sequential net.

ABB wants a modern interface where it is possible to build sequences with "drag and drop". With "drag and drop" one can select one object in the program toolbar and then drag it out over the workspace and directly place it anywhere. This makes it much smoother to build new sequence nets.

The thesis work can be divided into the following phases:
1. Study how the ABB and the competitor's SFC interface on the market today works.
2. Specify how the editing should work.
3. Implement a prototype in the ABB Advant Control Builder.

1.2 Limitations

Because of the limited time we have chosen to build our program on the existing SFC prototype for the Advant Control Builder. To start from scratch would have taken too much time and in this way we could concentrate on adding more functions instead of rebuilding the already working functions.

We have limited our thesis work to modify the off-line editing, i.e. the editing when the control program is off. In this mode the program development is made.

1.3 ABB Automation Products

ABB Automation Products is a part of the ABB automation-segment, the world-leading manufacturer of automation systems.
The business consists of the development and manufacturing of products for control, supervision, control and protection of processes within the industry and in plants for electrical power. These products are a platform for the automation solutions that ABB offer to customers all over the world.

The ABB effort is to make products that are open, easy to use and simple to enlarge and adept to modified needs.

ABB Automation Products has a sale of 2400 million SEK, 1400 employees and spend 25 percent of the sales on Research and Development.

1.4 Advant Control Builder

The control program where our improved SFC has to be implemented is the ABB Advant Control Builder. Below is a short description of Advant Control Builder.

General:
The Advant Control Builder is a fully integrated Windows application for programming and configuration of the ABB products Advant 210, Advant 250 and Advant Soft Controller. Advant Control Builder contains several programming languages. The control program is developed off-line and simulation of the control program can be made without having a controller connected. Advant Control Builder also has a number of on-line functions for test, program simulation and start-up. The status of variables, I/O-signals etc, can be supervised on-line.

Project:
Advant Control Builder contains a project explorer, which is a tool for navigation in, creating and modifications of projects. In the project explorer both the controller hardware and software are configured.

In a project the Advant Control Builder can handle an application that is maximum divided into three programs, which as standard are called Fast, Normal and Slow. In these three programs the program-code, function blocks and functions can be placed freely. In each of the three programs there is a connection to a task. For each task the interval time and priority is set. The users can create their own data types and function blocks when needed. They can be reused later.

Programming:
The programming in Advant Control Builder, supports data types such as Boolean, integers, floating point numbers, strings, time, date, etc, everything according to IEC 61131-3. The basic data types can also be combined with new structured data types.

Programming using the Advant Control Builder is made off-line. In the released version of Advant Control Builder there are two languages available in accordance with IEC 61131-3, that is ST (Structured Text) and (IL Instruction List).

ST is a high level programming language for very structured programming. It has a large menu of constructs for assignment, functions, function block calls, expressions, conditional statements, iterations, etc. IL is a programming language, where the instructions are listed in a column with one instruction per line. It is structured in the same way as a simple machine assembler.

The intention is that the Advant Control Builder would be complemented with more programming languages. There is already an existing prototype for SFC programming to Advant Control Builder available. This prototype is just under development and the thesis work is about improving its functionality.

The Advant Control Builder also has a program editor that contains several tools to make the programming easier and simplify debugging. They include for example functions for syntax checking, cut and paste, drag and drop and search and replace. It is possible to have several windows open at the same time, which gives a good overview of the entire application.

Compilation:
When the user program is compiled, machine code is created. This code is then optimized for the actual control system before it is downloaded to the control unit. When the compilation is started there is an extensive test of the program made and errors are easily detected. The error messages that come up works as links to the error in the code, the user just has to double-click on the error message.

To test the user program without having a control system connected the user can use the Advant Control Builders simulation mode. In the simulation mode all the tasks are executed locally in the Advant Control Builder. The control program Advant Soft Controller that can be run together in the same PC adds functions for development, testing and training.

Libraries:
Advant Control Builder contains a comprehensive set of predefined functions and function blocks. They are placed in several different libraries and can be used in the projects.
There is for example a System library, which is always available. It contains all basic data types and functions, for example type conversions, math and time.

The Advant Control Builder has a Logic function library with function blocks for flip-flop, timer and counters according to the IEC 61131-3.

There are several other libraries such as the Communication library containing client function blocks for different protocols. There is a Control library with regulators and an alarm library for alarm and event detection.

On-line functions:
The Advant Control Builder contains a number of functions to make the on-line testing easier.

- Status inspection. I/O signals and variables are inspected on-line.
- Force. The I/O signals can be forced to a chosen state.
- Overwrite. Temporary overwriting of a chosen variable for one run.
- Tasks. The user can select one-time execution to facilitate the program debugging. Warning and errors are indicated in the project explorer, when for example the program is stopped or overloaded.

Program modifications can be made on a running application. The modification can be implemented in the control system without stopping it, which mean that the variable values will be retained.

1.5 Outline of the Report

The outline of the thesis is described below.

1.5.1 Phase 1: Training and information obtaining

The purpose of this part of the thesis work was partly to learn and understand how SFC is used. It is necessary if we want to make a better program. The purpose is also to get new ideas to different program solutions, from other programs and from us.

The work began with learning how to use SFC. We tried SFC in SattLine and built different simple control programs in the studying purpose. We first had to understand and be able to use SFC ourselves to see how a new program shall work. We also studied a product from ABB Automation Products that is under development, which is a SFC to the Advant Control Builder.
Some of the documentation for SFC standard was also studied.
We then went on with studying SFC programming in the programs from competitive program manufactures. This was made by running the demos that ABB had an also by getting several more demos via Internet. We also looked at programs that are used to completely different things, suck as CAD-programs for electrical- or mechanical construction. These programs could add some ideas how our program shall look like and how it should work.

The next phase was to investigate the programs closely. Some programs were immediately set aside while other programs got our attention. Control programs with SFC language available, that were interesting were studied closely and then evaluated, see chapter 2.2.1. The CAD program where also studied and evaluated, see chapter 2.2.2.

We also work with own ideas and solutions to how a new SFC program shall work to be as simple and smooth as possible to use.

1.5.2 Phase 2: Producing a Function Specification

In this phase we worked with selecting the best of our ideas and thoughts from phase 1, to get a complete program. Different concrete solutions where created from the ideas in phase 1. Advantages and disadvantages by the different solutions where evaluated and finally the functions where decided one by one. Some functions were changed several times during the process.

The result of the work in this phase is a function specification. The function specification was written in a standard document for function specifications.

The prototype to SFC interface for Advant Control Builder is built with Visual C++.
Since none of us had used Visual C++ we also trained in programming with this language, to prepare us for the next phase.

1.5.3 Phase3: Implementation

Now it was time to try to create the theoretical program that we had made in the function-specification.
After discussions with our supervisor Håkan Augustsson we came to the conclusion that we should use the existing prototype of SFC in Advant Control Builder and try to improve it. We did this to save time so that all time was not spent on building the basic interface that is already made. Instead we could concentrate on implementing new functions.

We began to study the source-code to the prototype to learn the how the program works and how the procedures and functions in it work. Then we began to implement functions that we had in the function-specification. We began with the functions that we thought was the most important and left the less important to the future.

Due to lack of time and because some things are more difficult to solve in the reality than in the theory, the program did not become exactly as the program in the function-specification.

2 Sequential Function Chart

Here is a general description of SFC and SFC-programs. SFC stands for Sequential Function Chart.

2.1 Background

As long as the applications are small there is no need to structure the sequence process, but when the applications get larger and the complexity in the control events increase, there is also an increasing need for better functional descriptions.
As a tool for “top-down” analyze and representation of a control sequence, when logical expressions such as ladder diagram and function block diagram is not powerful enough, the SFC got introduced.

SFC is a special high-level language to describe control sequences in graphical schedules. At the late 70s the first function chart program Grafcet was developed in France and it has later been the base for the definition of the international standard IEC 848 (“Preparation of function charts for control systems”).

2.2 Example

The use of function charts is illustrated with an example (see Figure 2.1).

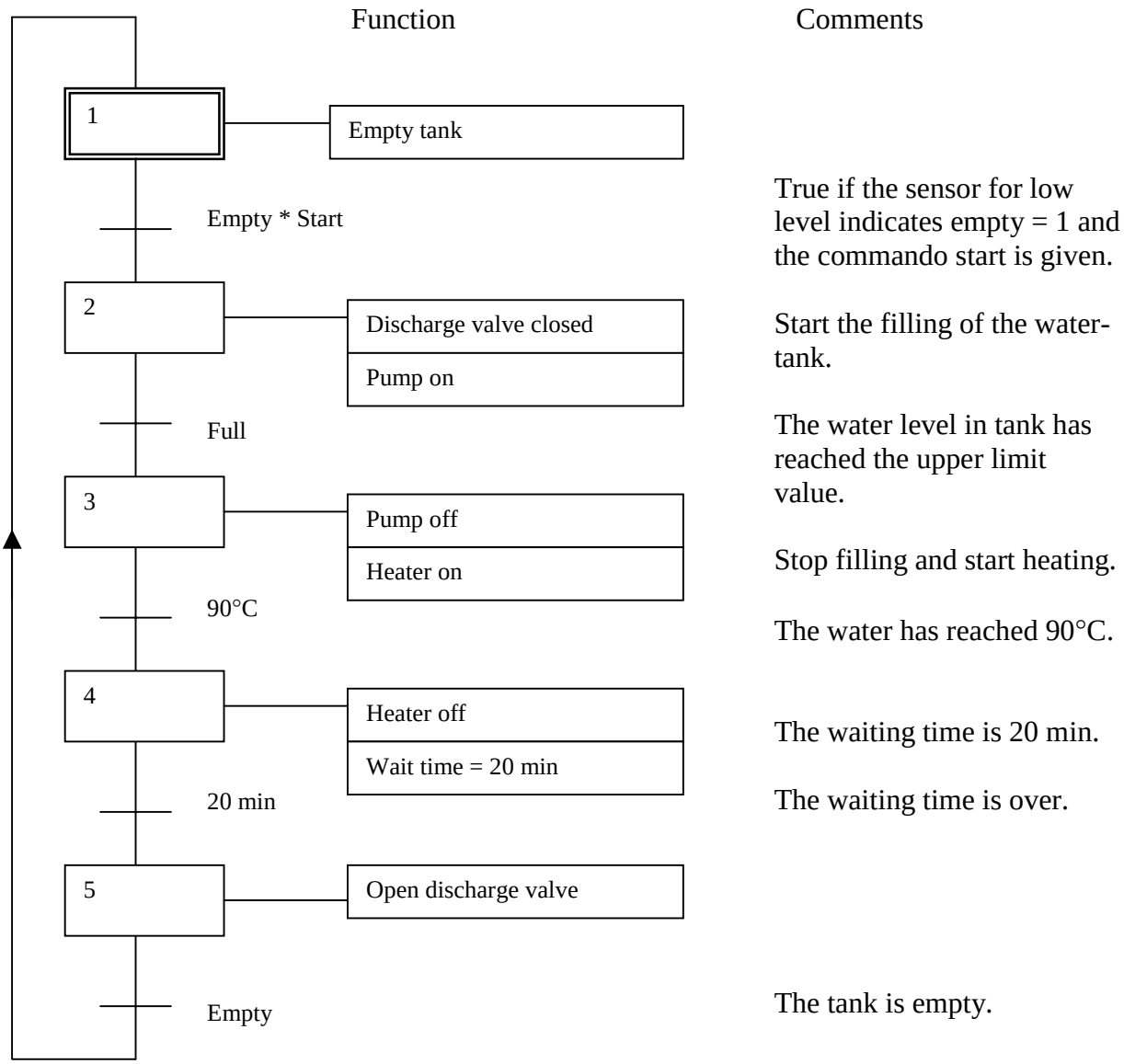


Figure 2.1 Grafcet illustration of a simple tank operation.

A tank shall be filled with water. A water-tank has to be filled and then the water has to be heated to 90°C. After 20 minutes the tank is emptied and the process starts all over again. When the water-tank is empty a sensor signals “Empty” and the tank can be filled again. This indication is connected to the “Start” signal, so the sequence can be initiated. In step 2 the filling operation is started, the discharge valve is closed and the filling pump is activated. When the tank is full a sensor signals that the upper limit is reached. The program continues to step 3, where the filling operation is stopped, the pump is turned off and the heating is started. The heating continues until the water has reached 90°C. Then there is a transition to step 4. In step 4 the heater is turned off and a timer begins to count. When the timer has counted to 20 minutes the program transits to step 5. In step 5 the tank is emptied, the discharge valve is opened. When the tank is empty the whole sequence can be repeated from the beginning.

2.3 Detailed Description

The SFC elements are used to structure the internal organization in a control program. They are written in a language that is defined in the standard [ref] to perform sequential control functions.

SFC describes the control sequences with predefined rules for:

- Controls that have to be executed and in which order they shall be done.
- Execution details for each instruction

The SFC can be divided into two parts, the “sequence“ part and the “object” or “control” part. In the “sequence” part the order between the control steps is described and in the “object” or “control” part is the internal actions that shall be executed. Graphically the “object” or “control” parts consist of boxes to the right of the sequence steps.

In this thesis work we will only treat the “sequence” part of the SFC.

According to IEC 611131-3 (1998-11-18) page 86 the SFC elements give a division of the control program in a number of steps and transitions connected to each other by directed links. To every step there is one or several actions and to each transition there is a condition connected.

Step:
The program behavior in a step follows a number of rules defined by the associated actions that is connected to the step. The step can be either active or inactive. At any given moment, its active steps, the internal and the output variable values define the state of the control program.

Graphically a block that contains a step-name represents the steps. A vertical line attached to the top of the step represents the directed link to the step. A vertical line connected graphically represents the link from the step to the bottom of the step.

The step that is “active” is the step that is currently executed. To indicate if a step is active or inactive, there is a step flag. The step flag is represented by a Boolean, the value of the step flag is one if the step is active and zero if the step is inactive.
The time that is spent in a step is saved as the variable “step elapsed time” it keeps it value when a step is inactivated. The value on “step elapsed time” is reset when a step is activated.

The control program must have an initial state, in this state the internal and output variables have their initial values and the control program stand in its initial step. The initial step is the step that is initially active and there shall be exactly one initial step. The initial step is represented graphically by a step with double lines for boarder.

The number of steps per SFC and the accuracy for the “step elapsed time” is dependent on the implementation.

Transitions:
There are transitions between every step Thanks to the transition the program can pass from one or more preceding steps to one or more successor steps. When the program passes a transition the successor step(s) becomes active and the preceding step becomes inactive. The transition is made along the vertical directed link. To each transition there are associated steps, which is called transition conditions.
The transition condition shall result in an evolution of a simple Boolean expression.
Sometimes the user wants the transition condition to always be true, and then the symbol 1 or the keyword true shall represent it.

Actions:
Every action is associated with a step. The step can have none or several actions associated. If there is no associated action to the step, it will be considered as a WAIT function. The WAIT function is a function that is waiting for the successor transition to be true. An action can be described in several ways, for example with a ladder-diagram, logical circuits or with Boolean expressions

Action blocks:
This is a graphical element for the combination of a Boolean variable with one of the action qualifiers to produce an enabling condition.
The action block contributes with a kind of Boolean indicator variable; it can be set by a specific action to indicate its completion, time-out, error conditions, etc.
The graphical concatenated action blocks can have multiple indicator variables, but just one common Boolean input variable, it shall act simultaneous for all the concatenated blocks.

Action qualifier:
Each step/action association shall have an associated action qualifier. The action qualifier can have the following values according to IEC 61131-3 (1998-11-18) page 97:

No.	**Qualifier**	**Explanation**
1	None	Non-stored (null qualifier)
2	N	Non-stored
3	R	Overriding Reset
4	S	Set (Stored)
5	L	Time Limited
6	D	Time Delayed
7	P	Pulse
8	SD	Stored and time Delayed
9	DS	Delayed and Stored
10	SL	Stored and time Limited
11	P1	Pulse (rising edge)
12	P0	Pulse (falling edge)

In addition, the qualifiers L, D, SD, DS and SL shall have an associated duration of the type TIME.

2.4 Building Sequences

The SFC syntax can handle much more than just an iterative execution of the same control instructions. The initial step, step(s) and transitions can be connected in several ways, which makes it possible to describe many complicated functions. Three possible combinations are:

- Simple sequences, this is just a step followed by a transition or a transition followed by a step.
- Alternative parallel sequences consist of two or more transition succeeding a step, so that the execution can take alternative ways depending on external conditions. These sequences can be used to for example if-then-else conditions and are useful to describe for example an alarm situation.
  When using these sequences it is very important to prevent a simultaneous start. This is done by verifying the condition for selection of one of the program execution branches so that they are consistent and unambiguous.
- Simultaneous parallel sequences, are made up of two or more steps placed parallel after a transition. The parallel steps can be simultaneously active. They represent a concurrent execution of several actions. The double horizontal lines indicate the parallel processing. When the condition for the transition condition for the parallel simultaneous sequences is fulfilled, both branches become simultaneously active and are executed concurrently. The transition to the successor steps below the lower double horizontal lines can take place only after all the concurrent processes are terminated.

There is also another possibility to modify the control program, by putting jumps into the SFC. With a jump the execution can jump from one step to another location in the program. The jump must be preceded by a transition condition and it is not allowed to jump to a transition.

2.5 Using a SFC-program

SFC programs operate under real-time conditions, which normally requires intensive efforts with considerable investments in time and employees. In this case the designer of the SFC compiler does most of the work, while the user can describe complex control sequences in a simple way. The real-time aspects on the programming are also important for the design of the PLC, but it affects the final user only indirectly and in a limited way.

Programming and compilation are easily made on a PC. After compilation the code is transferred in form of control instructions to a PLC for execution. When the code is transferred the PC is no longer necessary during real-time PLC operations. Some compilers have a built in simulation tool and can show the execution flow without being connected to a PLC. There are also some PLCs that have built in compilers.

There are several advantages with describing the control program in an abstract way such as a SFC. The SFC programs are independent of a specific hardware and are more oriented towards the task than the computer itself.

The SFC implementation allows the code to be divided into smaller parts, for example that each machine in a complex line has its own graph. Graphs from several machines can then be assembled. Such hierarchical structure is necessary when programming large, complex systems.

More advantages with SFC are that it is easier for the non-expert to understand compared with for example ladder-diagram. Because of this the SFC is useful not only for complex operations, but also for simpler tasks. The advantage with a standard for automated operations is that more program-code can be kept and re-used, which is impossible with incompatible languages and devices.

2.6 SFC in IEC 6113-3

IEC stands for the International Electrotechnical Commission and in this chapter is a summary of what this standard says about SFC.

2.6.1 Fundamentals

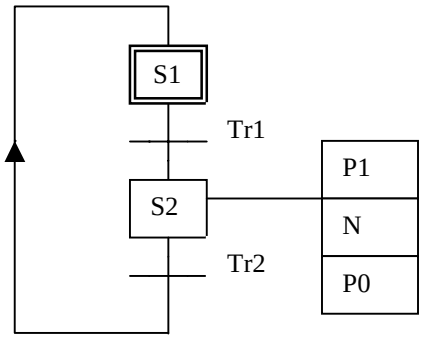


Figure 2.2 Simple sequence

The figure shows a simple sequence. S1 has a double line frame, which means that it is an initial step. The transition from S1 to S2 occurs when the expression in Tr1, which is a Boolean with process signals, is fulfilled. Then S1 is inactivated and during the entrance of S2 the P1 (pulse rising edge) is executed one time. Then S2 becomes active and equation N is executed as long as the transition condition Tr2 not is fulfilled. When Tr2 is fulfilled P0 (pulse falling edge) is executed one time S2 is inactivated. The sequence continues with P1 in step S1 and so on.

2.6.2 Single Sequence

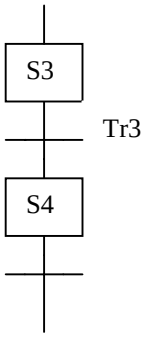


Figure 2.3 Illustration of a single sequence

A transition from S3 to S4 shall happen only when S3 is in active state and Tr3 is TRUE.

2.6.3 Divergence of sequence selection

A choice between different sequences is represented by as many transitions under the horizontal line as there are possible ways. The asterisk shows that there is a priority from left to right for the transition progress. If numbered branches follow the asterisk, the numbers indicate the user-defined priority of the progress.

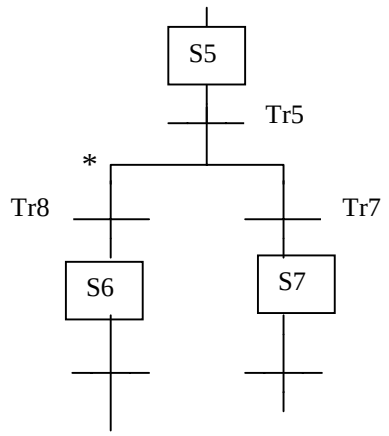


Figure 2.4 Illustration of a divergence

There has to be only one possible progress. From S5 to S6 when S5 is active and the transition condition Tr8 is TRUE or from S5 to S8 only if S5 is active and Tr7 is TRUE and Tr8 is FALSE.

2.6.4 Convergence of sequence selection

The end of a Sequence selection is represented of as many transitions above the horizontal line as there are selection choices that shall be terminated.

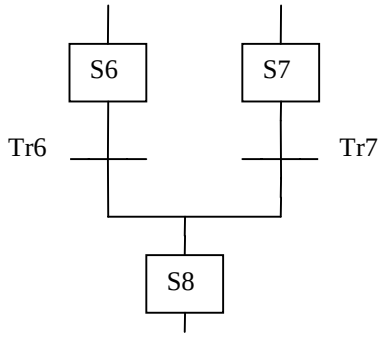


Figure 2.5 Illustration of convergence

A transition from S6 to S8 happens only if S6 is active and the transition Tr6 is TRUE or from S7 to S8 if S7 is in active state and Tr7 is TRUE.

2.6.5 Simultaneous sequence-divergence

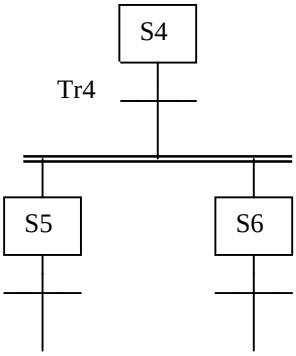


Figure 2.6 Illustration of simultaneous sequence - divergence

The execution from S4 to S5 and S& happens simultaneous when S4 is active and Tr4 is TRUE.

2.6.6 Simultaneous Sequence-Convergence

Below the double synchronization line only one transition is allowed.

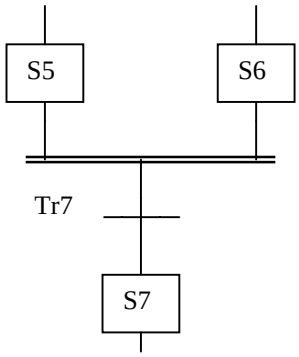


Figure 2.7 Illustration of simultaneous sequence - convergence

The execution from S5 and S6 to S7 shall happen, only if the steps above the synchronization line and connected to it is active and that Tr7 is TRUE.

2.6.7 Sequence skip

A sequence skip is a special case of sequence selection where one or more branches contain no steps.

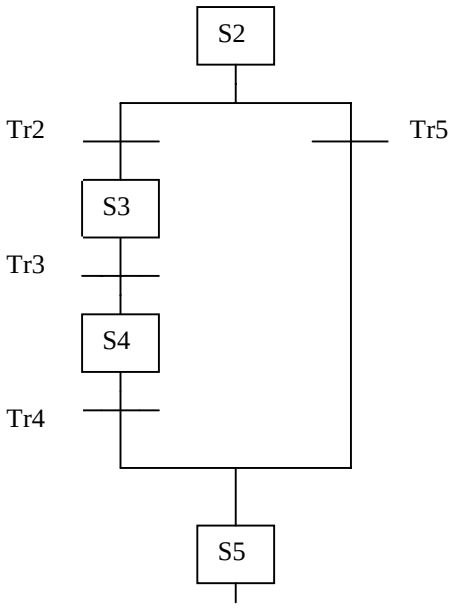


Figure 2.8 Illustration of sequence skip

An execution shall from S2 to S5 occur if Tr2 is FALSE and Tr5 is TRUE. S3 and S4 are then skipped.

2.6.8 Sequence Loop

Sequence Loop is a special case of sequence selection where one or more branches return to a successor step.

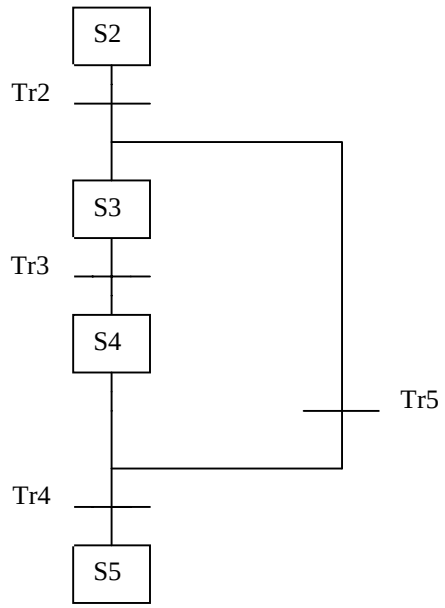


Figure 2.9 Illustration of sequence loop

An execution from S4 to S3 occurs if S4 is active state and Tr4 is FALSE and Tr5 is TRUE. That repeats S3 and S4.

2.6.9 Directional Arrows

The ”less then” sign (<) can be used to indicate right-to-left control flow and the “bigger than” sign (>) can be used for left-to-right control flow.
They shall then be placed on the flow-wire.

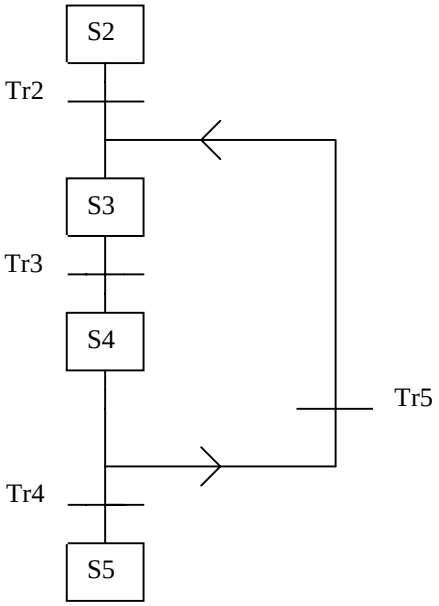


Figure 2.10 Illustration of directional arrows

The arrows show the direction of the flow from Tr5 to S3.

Another way to show the direction of the flow is by jumps. By the start point for the jump there is an arrow that shows the direction with a destination address next to it.

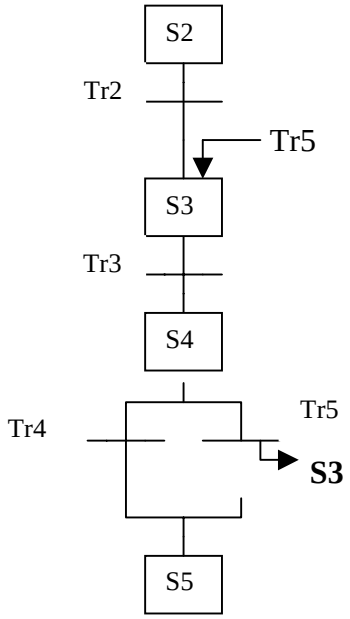


Figure 2.11 Illustration of a jump

The figure is exactly the same as with directional arrows but with jumps is used instead.

2.6.10 Jump into or out from simultaneous sequences

It is not allowed to jump in or out from simultaneous sequences.

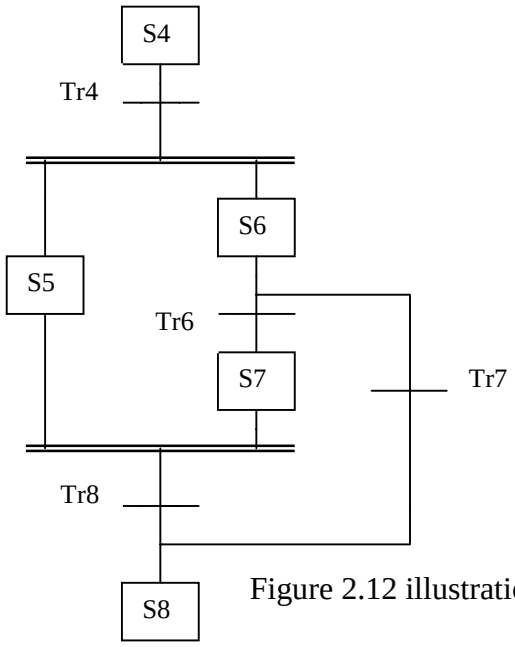


Figure 2.12 illustration of a forbidden jump

S5 and S6 are executed simultaneous when S4 is in active state and Tr4 is TRUE. If then Tr7 is TRUE when S6 is in active state S8 will be executed. The state S5 will wait in eternity for S7 to become active so the program can leave the simultaneous sequence. A forbidden locking has occurred in the SFC program.

2.6.11 Subsequences

To make a large SFC net easier to survey some parts of it can be placed in a subsequence that hides some part of it. The subsequence does not affect the SFC nets function. There are different ways to draw the subsequence depending on what type of elements the entrance and exit elements are.

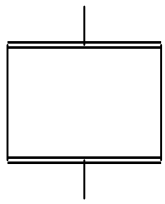


Figure 2.13 The entrance and exit elements are steps.

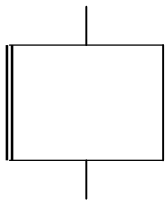


Figure 2.14 The entrance element is a step and the exit element is a transition

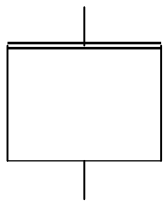


Figure 2.15 The entrance and exit elements are transitions.

3 Competitive Programs

In this chapter there is an examination of programs that can contribute with ideas to the development of a new SFC program. It is a study of similar, competitive programs, but also a study of programs in other areas, such as electrical- and mechanical-construction. It is possible that these programs can contribute with some ideas too.

We also took part of an opinion poll that were made earlier by ABB. We selected the parts that are interesting to this thesis work. The opinion poll gives us an idea of what the customer wants; the opinion poll is presented in 3.3.

3.1 Investigation of Competitive Programs

Limitations:
In this thesis work we have concentrated on the using of the SFC interface, the building and modification of the SFC net. Because of that limit the investigation include this and nothing more. Compilation and other functions in the program have been left uninvestigated.

Areas for the investigation:
In this investigation we have especially looked at the following areas:

- How plain is the interface? Does the program look nice and is it easy to understand what the different symbols mean?
- How easy is the program to use? Are the commands logical and easy to use?
- How fast can a new sequence be built? How fast and smooth is the program for the advanced user?

It can be very individual what a user finds good and bad with an SFC interface. To minimize this problem we have tried to think about who will use the program and their situation. It can also be difficult to set a value on the different areas in this investigation. To have something to relate the program performance (advantages and disadvantages) to we first studied all programs fast to get some kind of program average reference to have in the further, deeper investigation. Then we could relate for example the editing speed or the user friendliness to this average.

The “other programs” are the existing ABB products and programs from the competitors that we have studied the demo-versions of. The demos have we found and downloaded from Internet or that we got a demo CD from the manufacturer.

Others:
We have removed the names of the competitive programs, and just numbered them in order. This is done to avoid any possible irritation from the manufacturers of the programs.

3.2 Examination of SFC-Programs

In this section there is a short list of SFC-programs studied. The programs have been studied to learn how SFC works and to get ideas how a new SFC program can be designed. The programs represent different solutions to solve the same task and the intention is to sort out what the user considers positive and negative aspects of the different solutions.

The examination of the programs has been made in the following way:

- The programs were installed and some time was spent in starting the programs.
- The functions were investigated and some training in building simple sequences was done.
- A careful examination was done, all functions in the program where studied and noted.
- Finally the most important comments to the program where sorted into positive and negative things with the program.

On the following pages the most important observations are presented.

3.2.1 SFC in SattLine 2.2

SFC is one of the programming languages in ABB Automations product SattLine, which is a system for distributed industrial control.

General description:
Starting point is one step and one transition. To edit the SFC the user has to right click and select from the popup menu. There are no tools in the toolbar for this purpose. The program places the objects. There are no drag-and-drop functions.

The following things where noted as positive with this program:

- The graphics is plain with marked objects and errors displayed in color. The step-actions are divided in three parts, initial-, step- and entrance action.
- It is possible to divide the sequences in subsequences that increase the simplicity when the sequence is very big.
- There are three different levels of viewing the step-actions:
  Level 1, names are presented.
  Level 2, name and comments are presented.
  Level 3, name, comment and code are presented.
- The list of variables is plain and easy to use. On the other hand some of the commands in the list of variables are difficult to handle.

The following things where noted as negative with this program:

- There is no toolbox in the program and there is no undo command. Just a few commands have an accelerator key.
- There is no flexibility when placing the objects the program does it automatically. When there is objects deleted or cut out, the size of the wire is not adjusted.
- The step-actions can only be written in ST
- There are a number of commands that do not work satisfying and that probable nobody use.
- A lot of clicking and sometimes clicks in an unusual order. The position of an object must first be chosen, before the choice of object type is made.
- There is no warning when an incorrect operation is made instead the operation is blocked and can't be performed. This is not good for the beginner when he or she becomes uncertain if the error is in the program itself or if he or she is trying to make an incorrect operation.

Summary:

The program has a plain and simple graphics but the user can't control the position of step-actions and transitions. There are a number of commands that make about the same thing, for example: New branch, Parallel branch and Alternative branch. There is no toolbox so the user has to do a lot of clicks to perform some operations. In some situations there is a good indication of errors in other cases there is no indication at all.

3.2.2 Competitive Program No 1.

Program No1 is German. It is a program with the same field of applications as SattLine, where SFC is one way of programming. The demo-version which where tested was made in 1998 and was the latest available.

General description:
The starting point is an initial-step with transition and then a jump back. To edit the SFC, select one object and use the tools in the toolbar. Or right click when the objects are selected and use the popup menu. There are a lot of tools in the toolbar the user has the possibility to select to place new objects before or after. He or she also can choose to place parallel branches to the left or the right of the selected object.

The following things where noted as positive with this program:
- It contains a toolbar with plenty of tools, perhaps to many.
- There is an undo function and it works for several steps back.
- There are accelerator keys, but not for all functions.
- The step-actions are written in a separate and overlapping window. Several languages are available to write the step-action.
- A step-action can be written in SFC, this can be seen like some kind of subsequence. The plainness increases and the size of the sequence chart decreases. The size of the steps is easy to adjust.
- It is possible to set initial- and entrance-actions on the steps.
- The program contains a distinct list of variables that is continuously visible on the edge of the worksheet.

The following things where noted as negative with this program:
- It is only possible to delete separate parallel steps and transitions, because it is only possible to mark one object at a time. Serial steps must be deleted with the undo-function.
- The graphics is inflexible. There is no possibility for the user to move or place the objects. The program does it automatically.
- The zooming has only two steps and the program doesn't use colors, therefore the graphics feels indistinct.
- The step-actions must be opened with a text-editor to see the code. Consequently it is invisible in the sequence window.
- When incorrect operations are executed the program doesn't show any messages.

Summary:

The graphics in the program feels out of date, it's indistinct and inflexible. On the other hand the program toolbar is good with a lot of tools. Another good thing is that the user can write the step-actions in several different languages. It's also possible to write step-actions in SFC, that is the same as to make a subsequence and it makes complex sequences more distinct. Program No1 has an excellent list of variables, which is visible continuously and in which it is possible to direct declare new variables or change old ones.

There is an undo function and it works in several steps, which is very good but necessary since the delete command does not work properly. It can be hard for new users, when the program doesn't give any messages when he tries to do an incorrect operation. Instead the program blocks the operation so it can't be done.

3.2.3 Competitive Program No 2.

Program No 2 is a program from a French Company. With Program No 2 a PLC can be programmed with SFC, FBD, LD, ST and IL. In this examination it's only the SFC function that is interesting.

General description:

Program No 2 is a flexible program, the user construct the SFC with "drag and drop"-technique" and can place the objects freely. The program doesn't stop the user from making errors such as placing a step after another step. The starting point is an initial-step with very long outgoing wire. On the wire the user can place steps and transitions. To build parallel branches the user has four different commands in the toolbar to help.

The following things where noted as positive with this program:

- The program has a big toolbox with many tools. There are also a few accelerator keys.
- The step can be showed in four different detail modes, where the most detailed level shows the code or the LD:
- There is a lot of flexibility in the graphical SFC construction. The program is using "drag and drop"- technique.
- It is possible to have many windows opened at the same time for example step-actions or transition conditions.
- The list of variables is placed in an own window, which has a toolbar. It is distinct and easy to use.

The following things where noted as negative with this program:

- The step-actions can only be written in ST and the transition conditions can be written in ST or LD.
- The graphic is indistinct and feels out of date. No colors are used and the graphic has a strange construction. For example a new sequence has an endless output wire, furthermore the building of new parallel branches are complicated.
- The program contains no undo-commando.
- There is no indication in the graphics when an incorrect operation has been done; instead it will show up in the compiling of the program.

Summary:

The graphic is indistinct and it's hard to construct parallel branches. It has a good flexibility but with Program No 2 the programmer can do many faults in the graphic construction without having any fault detection.

There is a big toolbar where it's possible to use "drag and drop"-technique. The toolbox contains some strange commands, that are used in the construction of parallel branches.

The list of variables is in a separate window that has a toolbar; the list is very distinct and easy to use.

3.2.4 Competitive Program No 3.

Program No 3 from a German Software Company is a program with the same refinements like Program No 1.

General description:

SFC is just one of the languages in Program No 3. The starting point is a net with one step and one transition. To edit the SFC the user has to mark one object and right click. Then a popup menu appears the menu contains the necessary commands. Otherwise he or she can use the toolbar.

The following things where noted as positive with this program:

- The graphic is flexible. Steps and transitions can be moved along the wire.
- There is a toolbox with well-chosen functions.
- It has an undo function, but unfortunately it only works one step back.
- There are accelerator keys for most of the functions.
- There is distinct fault indication when the user tries to do an incorrect operation and the operation is stopped.
- The step-actions can be written in many different languages and the user can have several windows opened at the same time.
- An edit-wizard is available to help the user with common programming expressions, the edit-wizard changes when the user changes language.

The following things where noted as negative with this program:

- In the SFC graph the code of the step-actions and transitions is not visible, the user has to open a text-editor first.
- When a step with transition is deleted the SFC does not adjust its size. The same thing happens when parallel-steps or transitions are added.
- There is no separate list of variables. The variables can only be seen in code.

Summary:

Program No 3 is a modern program with nice, colorful graphic. It's distinct and clear but the automatic adjustment of the sequence net doesn't work properly. It is only possible to draw wires between objects that have nodes and it is only possible to connect input and output nodes. The toolbox is scanty; there are only four tools that can help the user under construction of the SFC-net. The edit-wizard is a good refinement to make the programming easier. It isn't possible to see the program-code in the steps and transitions without first opening a window. Fault messages appear when an incorrect command is to be executed.

3.2.5 Competitive Program No 4.

Program No 4 is a program from a German Company.

General description:

SFC in Program No 4 is build as a network with different sized squares. Steps are placed in the largest squares and transitions are placed in the smaller ones. Between these two types of squares there are smaller squares where wires are drawn. The step-actions can be written in FBD, IL or LD. Since the control is comprehensive so the risk of doing anything wrong is small.

The following things where noted as positive with this program:

- It is very smooth to copy and move steps and transitions.
- Immediately after a new step or transition is placed, there is a text-string opening to enter the name of the object.
- There are a lot of accelerator keys.
- The list of variables is distinct.

The following things where noted as negative with this program:

- There is no toolbox at all. This leads to a lot of clicking to place new objects and especially parallel steps and transitions.
- The graphic is tiresome. There are no colors used and the flexibility is small.
- The list of variables placed in an own window that always is maximized. The list must be closed if the user shall be able to see the worksheet.
- Transitions and step-actions can only be written in FBD, IL or LD.
- There is no indication of errors but it is quite difficult to do wrong when there is so little flexibility.

Summary:

The graphics is boring and inflexible when steps and transitions just can be placed in the given squares. Copying and moving objects between squares work well. It is a disadvantage that the step-actions and transitions only can be written in FDB, IL and LD. There is no indication of errors, but it is quite difficult to do wrong. Since there is no toolbox in the program there will be a lot of clicking.

3.2.6 Competitive Program No 5.

Program No 5 is a PLC program from a Canadian Company.

General description:

In Program No 5 the user has a little library window from which new objects can be dragged to the worksheet. The objects in the library is step, transition, action, stand-alone action, link, OR link and AND link. All objects have nodes where it connects to other objects. The nodes indicate if the connections are correct.

The following things where noted as positive with this program:

- Very flexible construction of the SFC net. The placing of objects are done in the following way: The objects are chosen in a list in a separate window where the objects are seen in there "natural size" (same size as when they are placed). When the objects are chosen, a picture of the object is following the pointer ("drag and drop" function).
- The objects have nodes which are black when they are connected and red when the objects are incorrect connected or unconnected. Wire is a separate tool in the toolbox this makes it possible to draw wire between distant object nodes.
- It is easy to move or delete objects in the chart.
- To move an object: Press the left mouse-button over the object and move the mouse.
- The zooming is working well. There are four different zooming tools in the toolbox.
  - Zoom in
  - Zoom out
  - Zoom window, zooms the drawn window.
  - Zoom all, makes the whole SFC visible in the worksheet.
- Both undo and redo functions in the toolbox.

The following things where noted as negative with this program:

- The graphic is tiresome with white background color and objects drawn of black lines. Both errors and marked objects are red.
- It's complicated to add the separate step action-box or transition-box. They are separate objects with own nodes to connect to steps or transitions.
- To add a parallel object the wire must be drawn separately.
- No automatic adjustment of the sequence-net. For example the user has to do the adjustments to make room for a parallel action.
- The list of variables is very difficult to use.

Summary:

Program No 5 is a very flexible program with "drag and drop" functions. But both steps and step-actions have to be separately dragged out to the worksheet. This is unnecessary when a step never is used without a step-action. The graphics is boring with white background and very simple objects. Both marked objects and incorrect nodes are red, which makes it a bit indistinct. The toolbox contains a lot of functions and is it is easy to modify, move and delete objects, in the sequence chart.

3.3 Examination of Other Programs.

Introduction:

Here follows notes from the studied programs, which were not SFC programs. These are programs that do not necessary have to be used in PLC programming. Instead it can be programs for electric construction and mechanical engineering. The reason why they were examined is that they contain graphic functions that is used or could be used in a SFC program.

3.3.1 MicroSim Schematic:

This is a program made for electric construction.

The following things where noted when the program were examined:

- MicroSim Schematic has ”drag and drop”-functions when a component is placed. After the component is chosen and dragged out to the worksheet, its outline follows the cursor. The user just presses the left mouse-button to place it.
- There are four different zoom commands and they are all available in the toolbar.
  - Zoom in
  - Zoom out
  - Zoom area, zooms the window defined by the user.
  - Zoom to fit page, the window is adjusted to fit the sequence net.
- If the component is connected wrong, for example the outgoing wire from a component is connected with another outgoing wire, there will be an error message displayed. The error message is a yellow exclamation mark and a cross over the incorrect connection point. The connection is made anyway.
- The net can be divided into different boxes, which can be useful when it is a big sequence net.
- It is possible to create separate text-windows for messages.

3.3.2 HiDraw:

HiDraw is a program especially made for electric, pneumatic and hydraulic design.

The following things where noted when the program were examined:

- There is a continuous open field on the worksheet, were there are different components showed. From this field the user can drag the components out to the construction. If the right mouse-button is pressed down in this field there will be a big component library displayed. When the user marks a component in this library it will become visible in the component field.
- When one object is marked and the right mouse-button is pressed there is a little menu displayed with the following possible choices: Rotate, mirror x-axis, mirror y-axis, cut, copy and properties.
- In ”Properties” the marked components color and line-type can be modified.
- There are connection-nodes on every component where the wire must be connected.

3.3.3 CircuitMaker:

In CircuitMaker electric circuits are constructed with different components.

The following things where noted when the program were examined:

- The wires are drawn to different connection-points, which become red when the position for connection is correct.
- The components are placed with ”drag and drop”, and the components can be rotated in the worksheet by a click on the right mouse-button.
- In the toolbox there is a text-line that immediately describes the type of tool. The user doesn’t have to wait for the yellow popup tag.
- There are accelerator keys for most of the commands.
- When two wire-ends are close enough to each other, they will be automatically connected.

3.3.4 WinDraft:

WinDraft is a program for design of electronics.

The following things where noted when the program were examined:

- To connect different components, chose ”Wire” in the toolbox. Then a ”W” is displayed next to the cursor and only click between the positions draws the wire.
- When the command Cut is chosen the cursor becomes a scissors.
- There is an undo-command and it is placed in the toolbar.
- When ”wire” is chosen in the toolbox, there is a little box around the wire-ends that indicate that the position is right for connection.

3.3.5 More programs:

Besides the programs above their where also an examination made of the following programs:

- EasyCAD
- Toshiba
- PSpice
- CAD Std Lite
- TraxMaker PRO
- Dw-2000
- DesignWorks 4
- FastCAD
- Siemens Simatic
- Melsec Medoc
- SoftControl 3.0

None of these programs could bring anything new.

3.4 Opinion poll

Introduction:
Andreas Hellström on ABB Automation made an investigation 1999. The investigation was directed towards experienced SattLine programmers and manufactures. It was based on the SattLine 2.2. We have selected the questions and comments that are interesting for this master thesis work.

Purpose:
The point is to get ideas and customer desires to the new SFC program that this thesis work shall result in.

The investigation:
From the investigation that were sent the following questions were considered interesting for this work:

3.4.1 Questions

(1) Several initial steps

In SattLine 2.2 it is possible to have several initial-steps in a sequence-block. We want to remove this possibility and just allow one initial-step per sequence-block.

- Is there any practical use in having several initial-steps?
- Can the possibility to have several initial-steps be deleted?

(2) Branch from one sequence to another

In SattLine 2.2 there is an opportunity to have branches from one sequence to a different sequence (within the same module). We want to remove this opportunity and just allow branches within one sequence. The synchronization between different sequences has to taken care of with own variables.

- Is it common to use branches from one sequence to another?
- Can we remove the possibility to have branches between the sequences?

(3) Branches

In today's SattLine 2.2 branches can be used almost anyhow. The system has no control of insane programming, for example branches in and out of parallel structures. The only protection there is today is that the danger is mentioned in the SattLine manuals. We think that the user has little practical use of the flexibility there is and instead get problems with the allowed bad programming. We want to restrict the functionality to just allow conditional jumps, and that the semantic forbids jump in and out of the own step.

For the continued discussion we refer to Appendix [?]:

Alternative 1 shows branch from one step. This corresponds to an alternative branch, the execution continues in either S1 or S2. The strange thing with this construction is that the transition condition is placed wrong. The evaluation order can seem strange; the branch has the highest priority.

Alternative 2 shows the branch from a transition. This corresponds to the beginning of a parallel branch, the execution continues in both the steps after S3 and in S4. Nobody know what will happen next?

Alternative 3 shows the construction that we believe is used in reality. It consists of an alternative branch where a break is placed after the transition where the branch comes out. In reality this is conditional jump. Either continue as usual or jump to S6.

If possible we want to keep the functionality shown in Alternative 3. We definitely want to remove alternative 2.

- Which construction is used in reality?
- Is alternative 2 ever used?
- Should one manage if just the functionality in Alternative 3 existed?

(4) Other points of view

Take the opportunity to mention other errors, desires or strange things that affect the sequences. Welcome to put priority on the functions that you consider has to exist or must be corrected.

Priority 1 has to be there.
Priority 2 ought to be there.
Priority 3 ought to be in a future version of the product.

3.4.2 Answers and Comments to the Opinion Poll

Here beneath is a summary of the answers and comments of the SFC investigation. When there were a lot of comments, they were examined and sorted into the following categories:

- Objects construction.
- Allowed construction.
- Construction of the SFC net.
- The graphic.
- Others.

(1) Several initial steps
Several initial-steps:
Nobody had the need of using several initial-steps in one sequence. IEC 1131 just allow one initial-step.

Branch from one sequence to another:
The majority thought that branch from one sequence to another should be removed.

(2) Branches:

The majority thought that it was enough with conditional jumps.

(3) Object construction

- Be able to put in comments in transitions.
- A chooses of fork gives automatically ”break”, writes it clearer. ”Break don’t need to exist as a concept.
- Observe that a sequence you usually jump to is of the type ”non connected”, this type of sequence must therefore be left.
- ToolTip to the step-action and to the outstearing of a step. You don’t need to double-click on the step-action.

(4) Allowed constructions

- It should be able to jump to the same step, for instance in cyclic functions.
- The handling of the fault management will be much more hard to work with if jump to another sequence is taken away.
- Automatic fault detection is desired. For instance by a special type of faults, like object in manual, I shall always jump to a certain step. When I have several fault-types that would imply a certain action than I wish that these jumps would be declared automatically. F1 to F4 would be declared very often in every (almost) step. This makes the sequence unwieldy (presentation) and hard to program.

(5) Construction of the SFC net

- Mark a whole step even if the step goes outside the screen, unfortunately sometime impossible to zoom out the whole sequence-step.
- Cut and paste function should be improved
- The whole step should be able to click on and not only the text and frame.
- When a variable is changed it should be enough to just a click on the value and than write the new straight in the diagram, without need to right-click, menu and a dialog.

(6) The Graphics

- If we would have three shared code in one step, so must the line between the code be different from the connections.

- Is it possible to configure the background-color. The Fonts must be the same as in the other editors.

- More 3-D in the graphic, step, conditions and others.

- Hard to separate the three ”equations” on the screen because the same type of lines is used even in equations.

(7) Others

- Names of the forks in a list with possibility to mention search profile, the beginning sign at the name in a fork.

- Functions for ”hot keys” with programming and online: go to the active step
  At a fork: go to the step before the jump, go to next/previous step, go to first/last
  Step in a fork/sequence.

- Printing of a sequence on text format, the graphic print today generates many sides with much air on every side.

4 Function Specification

In this chapter it is presented how our function specification was developed. In the function specification all functions of the program is specified. We use a standard ABB document as a template for the function specification.

In 3.1 we describe the demands we have on the new program. In 3.2 we show the different solutions we have to the program functions that we have chosen.

4.1 Development of the Program

4.1.1 General demands

The first thing to do in the development of the new program is to structure the demands and desired qualities on the new program.

Limitations:
In this thesis work we have only been interested in the using of the SFC user interface, because of that the demands are also limited to this area.

The user:
A very important aspect in the development of a program is to identify the user of the program. We have estimated that the user of this type of programs usually is an experienced application programmer that uses the program almost daily. Because of this the program does not need to have a lot of error messages and warnings that pop-up. We have given priority to make the program fast and flexible instead of making it easy to use for beginners.

Demands:
It is difficult to specify the demands on the program, but we say that the program must be:

- At least as plain as the old program.
- Decrease the time for building new sequences, which in fact is the main goal with the new program.
- Be user-friendlier than the old version.

It is difficult to set weighting factors on the different demands, so we have to relate the program to the older programs from ABB and the competitive programs.

It is also quite individual what the user finds good or bad with different functions. We have tried to have the average user in mind and also got some feedback from the ABB personal.

4.1.2 Method

The development process went as following:

- We tried to have three to four different ideas to how each function in the program should work. The ideas were our own ideas together with ideas from the competitive programs.
- Evaluation of the different ideas according to the demands and wishes.
- Selection of the best solution.

4.2 The Selection of Program Solutions

In this chapter different program solutions are described. It is also mentioned which solution that is chosen and why it has been chosen.

4.2.1 SFC construction methods

Right click menu

This construction method is used in Sattline. It takes long time to place an object. The user must first select a step or a transition by left clicking on the element. Then the user has to right click and choose from a list the object that is placed. The object will be placed after the released object. This means that the user must click on the mouse three times to place one object.

Toolbox

The user must first select a step or transition by left clicking and then left click on the symbol of the object, which is to be placed, in the toolbox. It will be the opposite order of the placement. The user will first choose an object type that has to be placed and then place it. This means that the user must click on the mouse two times to place an object.

Drag-and-drop

The object is in a toolbox or in a window. The user chooses an object and then the object follows the cursor. The object is then placed when the object is in correct position and the user left clicks. This is a fast way to construct a SFC net and easy way for new users. When the user has placed one object the drag-and-drop object still follows the cursor. The user can then place the objects by left click one time.

Conclusion:

Drag-and-drop is a fast and easy way to understand the construction for the users. This is the construction method that we suggest to be used.

4.2.2 Drag-and-drop menu

- Toolbox
- A special window with the drag-and-drop objects in natural size
- Right click menu

Conclusion:

We have placed the drag-and-drop objects in a toolbox. A window would be a cleaner design but we have only five drag-and-drop objects. They can be shown proportionately plain in a toolbox. Right click menu can not be used with drag-and-drop object. The object can then not be dragged out in the window.

4.2.3 Choice of Objects

Level 1 The objects are divided in components

The SFC net is constructed with single object like wires, step, transitions and jumps. It takes a long time to draw wires between every object but it is a free way to build the SFC net.

Level 2 The objects are divided in elements

The SFC net is constructed with step, branch, transition and jump. It takes a long time to place every object.

Level 3 The objects also consist of several elements

The net is built with the following elements: Step with transition, branch, simultaneous sequence, jump and sequence selection. The Simultaneous sequence consists of two simultaneous steps and one transition, which are placed before or after the simultaneous lines. The sequence selection consists of two parallel transitions and one step, which are placed before or after the parallel transitions.

Conclusion:

The object has been divided like level 3. There is no need to divide the objects in more pieces, since the construction time would not be faster. A step should always be followed by a transition. The fault control of the net will be much easier.

4.2.4 User help to correct placement

The node method

The first idea for construction method we called the node method. In this method all drag-and-drop objects have wires and nodes on their outputs and inputs. When an object is to be placed is it moved to the worksheet and pulled over the existing SFC net, so that the nodes of the drag-and-drop object coincident with nodes in the SFC net. If it is a correct placement the nodes at the drag-and-drop object will be green and if it is an incorrect placement they will be red. Another variant is that step has a filled input node and unfilled output node. Transitions shall have a filled output node and an unfilled input node. When the drag-and-drop objects top node coincident with a node of the same type can it be placed. The disadvantage with the method is that the nodes will be small and hard to interpret.

The wires change color

Another idea is that the input and output wires to the drag-and-drop object shall change color when they are to be placed. The wire can be for instance green for correct connection and red for incorrect connection. The disadvantage with this method is that it would be interpret for users with color defects to interpret the wires when they overlap in the existing SFC net.

The objects change color

Another way is that the objects change color when an object is to be placed in the SFC net. The object is for instance green when it is a correct connection and red when it is an incorrect connection. The disadvantage this method is also that it is hard for users with color defects to interpret the colors. It would be easier if the colors were black and white. The disadvantage is that the net will be hard to interpret if the wires are black and the drag-and-drop object is also black.

Stop lamp

Another variant is to have a stop-and-go lamp, for instance red and green, like Winzip, which turns to green when it is a correct connection and red when it is an incorrect connection. The disadvantage this method is also that it is hard for users with color defects to interpret the colors. Black and white would be easier to interpret.

The cursor change appearance

The final variant is that the cursor change appearance for correct connection and incorrect connection. One idea is that a question mark would be seen next to the cursor when an object is to be placed incorrect. Another idea is that the cursor shall change appearance to a hair-cross when it is a correct connection. This is the same cursor that is used in drawing-programs.

Conclusion:

We decided to use the method where the cursor changes appearance. This is an easy and obvious way to show when it is a correct connection.

4.2.5 Handling of faults

1. It is possible to do faults during the construction of the SFC net. The faults are detected and shown in the compile.
2. It is not possible to do faults. Help-texts show what is wrong and the command is not executed.
3. It is not possible to do faults. Warning dialog-boxes pops up and the commands is not executed.
4. The program sends out warning dialog-boxes but it is possible to do faults in SFC during the construction. The fault is also detected and shown in the compilation.
5. Help-text shows what is wrong but it is possible to do faults in SFC during the construction. The faults is also detected and shown in the compilation.

Conclusion:

The method number two, for handling faults, has been chosen. It guarantees that it is not possible to do any faults and the command is not executed. This choice has been made because the only way to cutting down the construction time is to use this method. If instead method number one had been used the user instead had to look for faults in the compile file. The total construction time would be the same. Method number two will be a secure and direct way to control that the user constructs correctly. There are no dialog-boxes that irritate and stop the user. Instead there is a help-text popping up in the window frame that shows what the user made wrong. This is also good for a new user to understand the construction principle of the SFC.

4.2.6 How to rotate objects

Some drag-and-drop objects, for instance step with transition, must be rotated when they are moved over the existing SFC net. If it shall be placed after a step the drag-and-drop object should have the transition on the top and the step bellow. This indicates that a new transition and step will be added after the existing step. If instead a new step with transition will be placed after a transition the drag-and-drop object should have the step on the top. The rotation of the drag-and-drop objects can be done in two ways:

1. The object rotates with right-clicks.
2. The object rotates automatically when it is to be placed in the existing SFC net.

Conclusion:

The objects shall be rotated automatically (method 2). This is to having the user to avoid to manually rotate the object so it is fitted in the existing SFC net. That would increase the construction time.

4.2.7 The placement of drag-and-drop objects

The cursor is overlapping a wire below an element. If it is a correct placement when the user left-clicks the drag-and-drop object is placed after the element.
The cursor is overlapping a node. If it is a correct placement when the user left-clicks the drag-and-drop object is placed after the node.
The cursor is overlapping an element. If it is a correct placement when the user left-clicks the drag-and-drop object is placed after the element.

Conclusion:

The method number one has been chosen. The cursor should hit the wires when the drag-and-drop objects are to be placed. The drag-and-drop object is pasted between the objects that the wire is connected between. The user directly sees where the object has to be placed in the existing SFC net. The user left-clicks to place the object. This is a normal behavior in programs.

4.3 Summary of Program Solutions

The main features of the program solution is:

- The SFC net shall be constructed with drag-and-drop technique.
- The drag-and-drop object should be placed in a toolbox.
- The drag-and-drop object has to be a new step with transition, new branch, new simultaneous sequence, new sequence selection and new jump.
- The cursor should change appearance to a hair-cross when it is a correct placement in the net for the drag-and-drop object.
- It should not be possible to construct incorrectly. The program auto adjusts and help-text in the window frame shows the user what is wrong.
- The cursor should hit the wires when the drag-and-drop objects are placed. The drag-and-drop object is pasted in between the object that the wire is connected between. The user directly sees where the object will be placed in the existing SFC net.
- The drag-and-drop objects are rotated automatically when it is moved over the existing SFC net. When the user for instance has chosen Step with transition, the element on the top to changes to a transition when the cursor is on the wire under a step. Furthermore the element on the top changes to a step when the cursor is on the wire under a transition.
- The user has to be able to save user settings.
- To reduce the size on the worksheet the user has to be able to place objects in a subsequence.
- The user has to be able to save parts of the SFC net that he usually uses in the construction, in a special subsequence library.
- The drag-and-drop object that follows the cursor is removed when the user presses down the right mouse button.

5 Implementation

5.1 The Functions of the Program

The implemented program has the Advant Control Builder as base and then we have added several functions.
In this chapter the new functions that have been implemented to the program are explained.

5.1.1 Drag-and-drop Toolbar

The toolbar consists of five symbols (from left in the figure 4.1): new step with transition, new sequence selection, new simultaneous sequence, new jump and new branch.

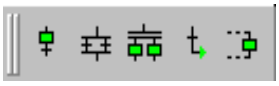


Figure 4.1 Toolbar for drag-and-drop objects

5.1.2 New step with transition

To add a new step with transition:

- Press down the symbol for new step with transition
- Move the cursor to the SFC worksheet. An object that looks like step with transitions follows the cursor. A help-text will pop up in the window frame " The step with transition will be placed after the selected element".

1. Move the cursor over a step. The step is highlighted with green colour, which symbol that it is selected. The object that follows the cursor has the transition at the top and the step below. That means that if the user presses the left mouse button a new transition and a new step is placed after the selected step.
2. Move the cursor over a transition. The transition (Tr1 in the figure 4.2) is highlighted with green colour, which symbols that it is selected. The object that follows the cursor has the step at the top and the transition below. That means that if the user presses the left mouse button a new step (S2 in the figure 4.3) is placed after the selected transition and a new transition (Tr2 in the figure 4.3) after that new step.

- Move the cursor over an alternative or parallel elements so that the whole alternative or parallel sequence is highlighted with green colour. If the user presses the left mouse button nothing happens. It is not possible to place a new step with transition in that way. The drag-and-drop object is still following the cursor.

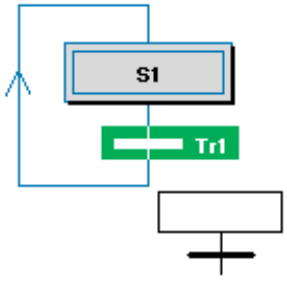


Figure 4.2 The cursor is placed over a transition

- After a new step with transitions is placed the object is still following the cursor and a new step with transitions can be placed. If the user wants to select another drag-and-drop object they can directly choose another in the toolbar.
- The user press the right mouse button if they want to get rid of the drag-and-drop object and return to normal mode.

5.1.3 New sequence selection

To add a new sequence selection:

- Press down the symbol for the sequence selection in the drag an drop toolbar
- Move the cursor to the SFC window sheet. An object that looks like a sequence selection follows the cursor. A help-text will pop up in the window frame " The sequence selection will be placed after the selected element".

1. Move the cursor over a step. The step (S2 in the figure 4.3) is highlighted with green colour, which symbols that it is selected. The object that follows the cursor has the alternative transitions at the top and the step below (figure 4.3). That means that if the user press on the left mouse button two new alternative transitions (Tr3 and Tr4 in figure 4.4) is placed after the selected step (S2 in the figure 4.3) and a new step (S3 in figure 4.4) after those new alternative transitions.
2. Move the cursor over a transition. The transition is highlighted with green colour, which symbols that it is selected. The object that follows the cursor has the step at the top and the alternative transitions below. That means that if the user presses the left mouse button a new step is placed after the selected transition and two new alternative transitions after that new step.

- Move the cursor over an alternative or parallel elements so that the whole alternative or parallel sequence is highlighted with green colour. If the user presses the left mouse button nothing happens. It is not possible to place a new sequence selection in that way. The drag-and-drop object is still following the cursor.

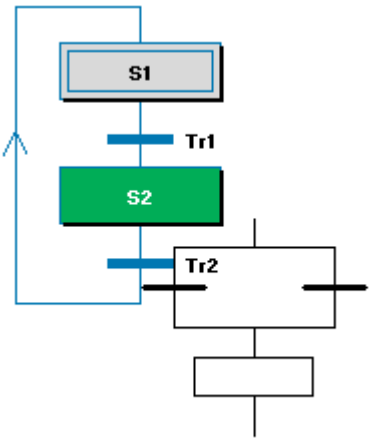


Figure 4.3 The cursor is placed over the step S2

- After a sequence selection is placed the drag-and-drop object is still following the cursor and a new sequence selection can be placed. If the user wants to select another drag-and-drop object they can directly choose another in the toolbar.

- The user presses the right mouse button if they want to get rid of the drag-and-drop object and go to normal mode.

5.1.4 New simultaneous sequence

To add a new simultaneous sequence:

- Press down the symbol for the simultaneous sequence in the drag-and-drop toolbar.
- Move the cursor to the SFC window sheet. An object that looks like simultaneous sequence follows the cursor. A help-text pop-up in the window frame “The simultaneous sequence will be placed after the selected element".

1. Move the cursor over a step. The step (S2 in the figure 4.4) is highlighted with green colour, which symbols that it is selected. The object that follows the cursor has the transition at the top and the simultaneous steps below (figure 4.4). That means that if the user presses the left mouse button a new transition (Tr5 in figure 4.5) is placed after the selected step (S2 in the figure 4.4) and two new simultaneous steps after that new transition.
2. Move the cursor over a transition instead. The transition is highlighted with green colour, which symbols that it is selected. The object that follows the cursor has the simultaneous steps at the top and the transition below. That means that if the user presses the left mouse button two new simultaneous steps is placed after the selected transition and a new transition after those new simultaneous steps.

- Move the cursor over an alternative or parallel elements so that the whole alternative or parallel sequence is highlighted with green colour. If the user presses the left mouse button nothing happens. It is not possible to place a new simultaneous sequence in that way. The drag-and-drop object is still following the cursor.

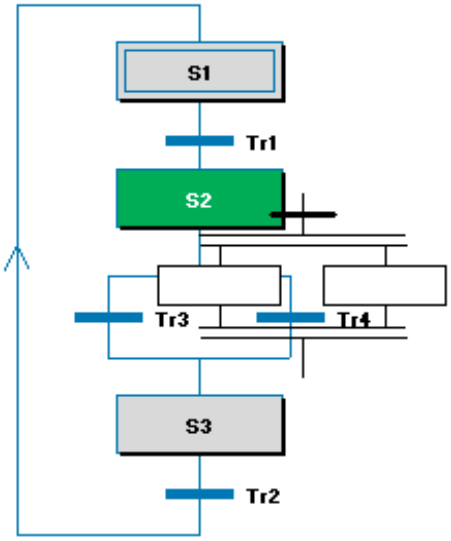


Figure 4.4 The cursor is over the step S2

- After a new simultaneous sequence is placed the object is still following the cursor and a new simultaneous sequence can be placed. If the user wants to select another drag-and-drop object they can directly choose another in the toolbar.

- The user press the right mouse button if they want to get rid of the drag-and-drop object and return to normal mode.

5.1.5 New jump

A new jump is added in the following way:

Select the placement for the jump.

- Press down the symbol for the jump in the drag-and-drop toolbar.
- Move the cursor to the SFC window sheet. An object that looks like a jump arrow follows the cursor. A help-text pop-up in the window frame “Select placement for the jump”.

1. Place the cursor over a transition. The transition (Tr1 in the figure 4.5) is highlighted with green colour, which symbols that it is selected. If the user press down the left mouse button a new transition (Tr7 in the figure 4.6) is placed parallel to the one that was selected. The new transition is then selected and it will be a jump when the jump-destination is decided. A help-text pop-up in the window frame “DoubleClick on the destination step or the background”.

2. Place the cursor over a step with a transition after instead. The step is highlighted with green colour, which symbols that is it selected. If the user press down the left mouse button a new transition is placed parallel to the transition that is under the selected step. The new transition is then selected and it will become a jump when the jump-destination is decided. A help-text pop-up in the window frame “DoubleClick on the destination step or the background”.

- If the user left-click on a step that is inside a simultaneous sequence with no transition below the step inside the simultaneous wires will a help-text pop-up. The help-text says “ Incorrect placement for the jump”. The drag-and-drop object is still following the cursor and the user can place the jump some place else.
- Move the cursor over an alternative or parallel elements so that the whole alternative or parallel sequence is highlighted with green colour. If the user presses the left mouse button nothing happens. It is not possible to place a new jump that way. The drag-and-drop object is still following the cursor.
- If the user press down the left-mouse button when nothing is selected nothing happens. The drag-and-drop object is still following the cursor and the help-text is still in the window frame “Select placement for the jump”.

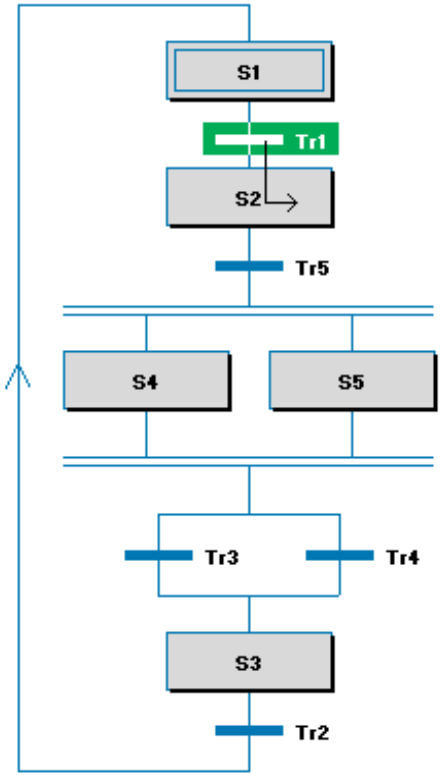


Figure 4.5 The cursor is over the transition Tr1.

Select the jump-destination.

- The help-text “DoubleClick on the destination step or the background” is in the window frame. That means that the user shall double click on the step (S3 in the figure 4.6) that shall be the destination of the jump or double clicks on the background so that a dialog-box pop-up. There shall the jump destination be typed. It can be a step that not yet is added to the SFC net. The added jump is show in the figure 4.7.
- If the user Double Clicks on a transition will a help-text pop-up "Please choose a step or hit the background" and the user has to double click on a step or hit the background so that the dialog-box pop-up.
- If the user Double Click on a step that is inside a simultaneous sequence and the transition that the jump goes from is not inside the simultaneous sequence a help-text will pop-up. The help-text is "Incorrect placement for the jump, can't jump into a simultaneous seq". The user has to double click on another step or hit the background so that the dialog-box pop-up.

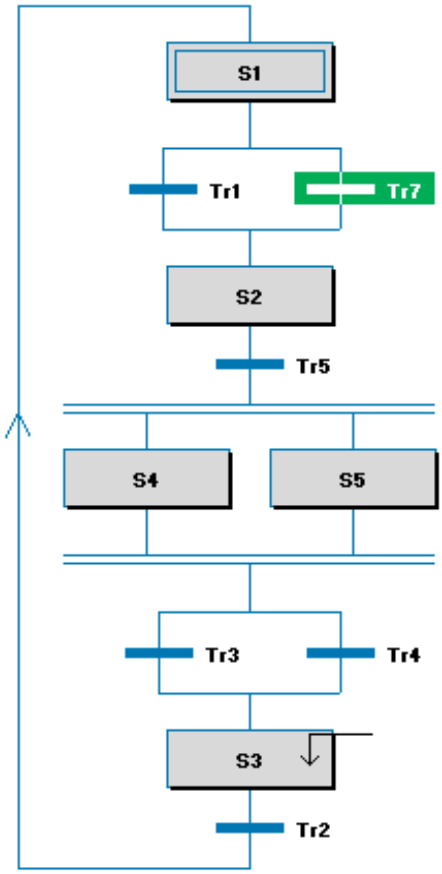


Figure 4.6 A new transition (Tr7) is added and the cursor is over step S3

To remove the drag-and-drop object or choose some other type

- If the user has not placed the jump it is possible to directly choose another drag-and-drop object in the toolbar or right click with the mouse to go to normal mode, which means the drag-and-drop object disappears.
- If the user is just to choose a destination of the jump they can right click and the drag-and-drop object disappears and the program returns to normal mode as soon as the mouse is moved. The transition that was added when the jump was placed is still in the SFC net. The user cannot directly choose another object if they are about to choose destination of the jump.

5.1.6 New Branch

A new branch is added by selecting the element that the branch starts before and the element that the branch ends after. The element must be of the same type but the do not need to be selected in order for instance up and down in the SFC net.

First branch element

- Press down the symbol for branch in the drag-and-drop toolbar a help-text pop-up "Choose placement for the Branch". Move the cursor to the SFC window. An object follows the cursor. The object looks like a transition (figure 4.7) when the cursor is placed over a transition and it looks like a step when the cursor is over a step. This means that an alternative transition is placed when the object is a transition and a simultaneous step when it is a step.
- Left click on the step or the transition where the branch shall start (the branch starts before the selected one) or the end (the branch ends after the selected one). The branch can start and end on the same element.

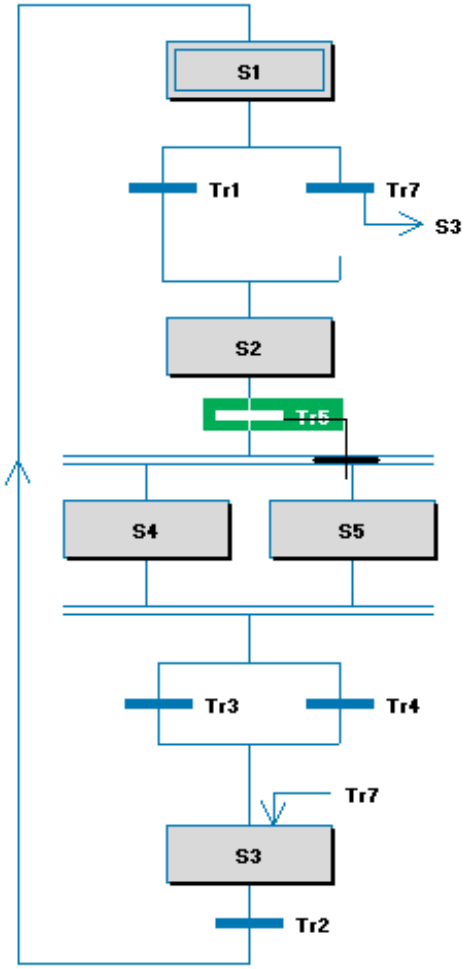


Figure 4.7 The cursor is over the transition Tr5.

Last branch element

- A new help-text "Select destination of the branch" popsup when the first element is decided. The drag-and-drop object has change appearance to a wire (figure 4.8).
- The user has to double click on the element, which will end or start the branch. In the figure the user has to double click on a transition (Tr 2). When the user has double clicked on the transition (Tr2) the SFC net looks like figure 4.9. The user then can place a new branch. The drag-and-drop object first branch is following the cursor.
- The element should be of the same type as the first one. If for instance the user first left clicks on a transition and then double clicks on a step a help-text that says "Incorrect placement for the Branch, only step to step or trans to trans" will pop-up and be shown in 4 s. Then the help-text "Choose placement for the Branch" pops up and the user have to start again and left click on the first element.
- The branch cannot go outside a simultaneous sequence the SFC program will then get locked. If the user double clicks on an element that is not in the same simultaneous sequence, as the first element, a help-text will pop-up. The help-text says, "Incorrect placement for the Branch" in 4 s. Then is the help-text "Choose placement for the Branch" pop-up and the user has to start again and left click on the first element.

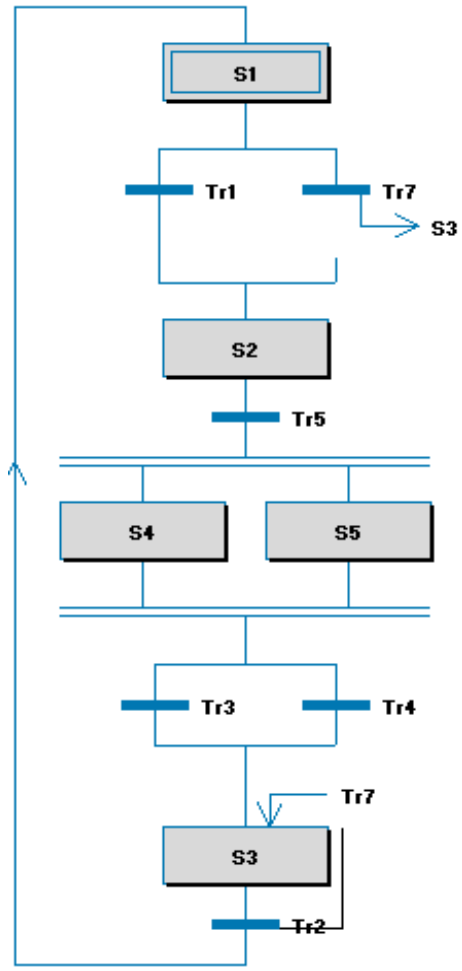


Figure 4.8 The cursor is over the transition Tr2

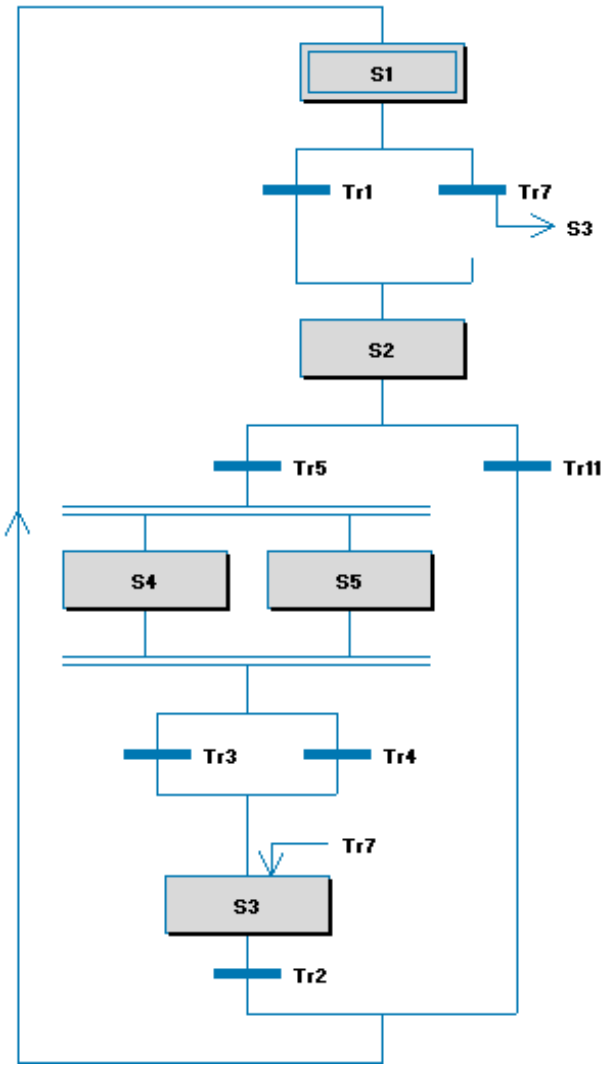


Figure 4.9 The branch is placed

- If the user wants to remove the drag-and-drop object is it just to right click with the mouse button and the object disappears and the program goes to normal mode.
- If the user wants to change drag-and-drop object directly after the branch object is chosen they can only do that when the first element has not been decided.

5.2 The Function-specification compared with the Implementation

All the functions that were specified in the function-specification have not been implemented and some of them are not exactly the same as in the function-specification.
Here follows the functions that have not been implemented and some ideas of how they could be implemented.

- The cursor does not change appearance when the cursor is in correct position. The cursor is of hair-cross type when drag-and-drop object is selected. The reason this is not implemented is that the master thesis is of the limited time. The function that controls what element the cursor is overlapping could possibly be implemented in the procedure OnMouseMove in the file *SFCView*. The program shall there control what kind of element the mouse is over. The control should be the same as in the jump and branch procedures in the file *Editseq*. If for instance the jump destination is to be decided and the cursor is over a transition the cursor shall be a stop sign. If instead the cursor is over a step the cursor shall be a hair-cross.
- The cursor shall not be over the wire between elements when a drag-and-drop object shall be placed. We use the existing method where the elements are selected. The drag-and-drop object will be placed after the selected object when the user left clicks. In the existing program the wires are not objects, so they are impossible to select. If the wires shall be selectable the wire must therefore be an object and how this is done we have no good answer to.
- The menus have not been modified according to the function specifications. This is not important according to the task and has been disregarded because of lack of time.
- The destination of the branch and the jump is decided with double-click in the implementation. In the function specification they are decided with left-click. This is a normal behaviour. We tried to use the existing left-click functions in the implementation but it didn't work. It seems that Double-click work properly and we decided to use it instead.
- No undo-function has been added in the implementation. This is not important according to the task and has been disregarded due to lack of time.
- No keyboard accelerators have been added. This is not important according to the task and has been disregarded because of lack of time.

5.3 Program-construction

This chapter is especially written to describe the program solution for the personnel at ABB, who will use or further develop the program. The part can be difficult to understand for persons with no access to the source code. Program files have been written in *italic* style.

5.3.1 Selection of Elements

If the drag-and-drop functions are to work, the elements must be automatically selected when the cursor is over them. In the present version of Advant the selection take place when the user left clicks on an element. The program handles the selection by use of events, that a process waits for. When the user presses down the left button a PickEvent appears and when it is released a ReleaseEvent appears. There is also a MoveEvent that appears when the cursor is moved. MoveEvent is not used in the present version of SFC. Therefore MoveEvent has been added in the file *InputEvent* so it works in the same way as PickEvent, which means that the cursor position is updated.

The process for selection handling is waiting for events in the file *mainprog*. If an event appears for instance menus and selections it is handled.

If a MoveEvent appears nothing happens in the file *mainprog*, the processes keep waiting. Additions have been made in the file *mainprog* so the process only waits for MoveEvent when a drag-and-drop object has been chosen. It handles as a PickEvent every other and as a ReleaseEvent every other. This is made to simulate the selection procedure that exists today. In the file *Redrawandwaitfor* it is control if the events that have to happen in *mainprog* actually have happened. In *Redrawandwaitfor* additions have been made, so the only event that can happen when drag-and-drop is selected is MoveEvent. This is done to prevent the program from getting locked in the file.

In the file *picking* the multiselection is disabled by turning off the selection frame, which is used to drag over the elements to be selected. This is made so that the selection frame does not pop up when a drag-and-drop element is placed, which can be irritating. If more than one object is selected all selected object will be deselected.
In the file *selection*, which handles the selection, is the multiselection disabled when drag-and-drop object is chosen.

If the user hits the wire that goes from the last transition to the first step, the whole SFC will be selected. This is irritating for the user, additions have been made in the file *drawseq* to disable this command for drag-and-drop.

Addition has been made in the file *SFCView* so that the command OnMouseMoveEvent(true,true) is executed when a drag-and-drop object is chosen. The first variable controls all drag-and-drop additions for selection in the files *mainprog, Redrawandwaitfor* and *picking.* The second variable controls the drag-and-drop additions for selection in the files *selection.* To prevent the program from selecting the whole SFC is done by executing OnMouseDeselect(true). This must be done separate because the file *drawseq* is not in the same directory that the other mentioned files.

5.3.2 Toolbar

A toolbar has been added for the drag-and-drop commands. It is placed in the *resource* file and has the name IDR_SFCDARGTOOLBAR. It contains the symbols: New step with transition, new sequence selection, jump and branch.

Adjustments for the toolbar, for instance that it shall be moveable, is made in the file *MainFrm*. Additions have also been made so that the toolbar only is visible when SFC language has been chosen. The initiation of the toolbar is made in the file *SFCView*.

5.3.3 The Drawing of drag-and-drop objects

All drawing procedures of the drag-and-drop object that follows the cursor is made in the file *SFCView*. When the user presses down a symbol in the drag-and-drop toolbar a boolean variable that is specific for each object will be true.
From the procedure OnDraw(CDC*pDC) the added procedure OnDragDraw(CDC*pDC) is executed. To reduce the flicker there is an area around the cursor made as a bitmap. The area has to be updated when the cursor is moved over the window. This is made in the procedure OnMouseMove(UINT nFlags, Cpoint point).

In OnDragDraw it is controlled which type of element that is selected. This is used for new step with transition, sequence selection, simultaneous sequence and for branch. If a transition is selected the drag-and-drop object shall have the step on the top for the three first mentioned cases. For branch the drag-and-drop object shall be a transition when a transition is selected and a step when a step is selected. The change in appearance after what is selected is made by the variable RotateStepTrans that is true if a step is selected. This variable is than sent into the drawing procedure. The command *jump* always has the same symbol that follows the cursor. This is due to the fact that jumps always start from a transition and goes to a step. If a jump is placed from a step to a transition a new transition will first be added after the selected step. The jump then starts from this new transition. If the jump instead goes from a transition a new transition will first be placed parallel to the selected one. The jump then starts from this new transition. The addition of a transition is in the present version of the SFC program.

The procedures for the drawing of the different object is OnDrawStepTrans(pDC, RotateStepTrans), OnDrawSeqSel(pDC, RotateStepTrans), OnDrawSimSeq(pDC, RotateStepTrans), OnDrawJump(pDC) and OnDrawBranch(CDC* pDC,bool StepBranch). The variable StepBranch has the same value as RotateStepTrans.

5.3.4 Helptext

The following have been added in the file *SFCView* to make helptext work:
The procedure DragHelpText(int HelptextType) has been added to write the help-text. The variable HelpTextType is sent with the procedure and controls what help-text that shall be written. DragHelpText is called in the procedure OnDraw, which is executed when the window is redrawn.

The following have been added in the file *Editseq* to make helptext work:
It is not possible to get the class CeditorMainFrame from the file *Editseq* and it is therefore not possible to write helptext commands in Editseq. It is also not possible to include *SFCView* in the file *Editseq* so that the procedures in *SFCView* will be accessible from *Editseq* and not only reverse. Instead has the procedure DragTextSeq been added in the file *Editseq.* In *Editseq* is an integer m_DragText set to a value when a helptext shall be written. The procedure DragTextSeq is called from the file *SFCView* and returns the value on m_DragText, which agrees with the value that shall be sent into the DragHelpText procedure.

5.3.5 The Cursor appearance

In the procedure OnMouseMove in the file *SFCView* changes the cursor appearance to hair-cross when drag-and-drop objects have been chosen.

5.3.6 New step with transition

All procedures where additions have been made, to make new step with transition work for drag-and-drop, are in the file *SFCView.*
If the user presses the drag-and-drop symbol for new step with transition the following happens in the procedure OnNewDragStepTrans when the user press:
The variable m_bStepTrans is set to true if the variable m_bDestJump and m_bDestBranch are false. These variables are true when the user shall select jump destination or branch destination. This is done to prevent a step with transitions to be placed when a jump or a branch is not completed. When m_bStepTrans is set to true all other drag-and-drop objects is set to false. This is made so that the user directly can choose a new object in the toolbar.

Then OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) is set, which is mentioned in the selection description.

If the user presses the left mouse button on a selected object the following happens in the procedure OnLButtonDown(UINT nFlags, Cpoint point):
The variable m_bStepTrans is true and therefor a step with transition is placed in the net. After that is it possible to place another step with transition in the same way.

When the user presses the right mouse button and release it the procedure OnRButtonUp(UINT nFlags, Cpoint point) is executed. There the following happens: Variable m_bStepTrans is set to be false, which means that the drag-and-drop object disappear. After that the procedure OnMouseMoveDeselect(false) executes, which restore the possibility to select the whole SFC net. Then is the procedure OnMouseEventEvent(false,false), which restore so the user has to left click on the element to get it selected.

5.3.7 New sequence selection

All procedures where additions have been made, to make new sequence selection work for , are in the file *SFCView*.
If the user presses the symbol for new sequence selection in the drag-and-drop toolbar the following happens in the procedure OnDragSeqenceSelection:
The variable m_bSeqSel is set to true if m_bdestJump and m_bDestBranch are false. Then all other drag-and-drop variable is set to be false. In the same way as new step with transition is the procedures OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) is executed.

If the user presses the left mouse button on a selected object the following happens in the procedure OnLButtonDown(UINT nFlags, Cpoint point):
The variable m_bSeqSel is true and therefor a new sequence selection is placed in the SFC net. After that it is possible to place another sequence selection in the net.

When the user presses the right mouse button and releases it is the procedure OnRButtonUp(UINT nFlags, Cpoint point) is executed. The following happens for new sequence selection:
The variable m_bSeqSel is set to be false, which means that the drag-and-drop object disappears, and the user can make other commands. After that is, in the same way as step with transition, the procedures OnMouseMoveDeselect(false) and OnMouseMoveEvent(false,false).

5.3.8 New simultaneous sequence

All procedures where additions have been made, to make simultaneous sequence work for drag-and-drop, are in the file *SFCView*.

If the user presses the symbol for new simultaneous sequence in the drag-and-drop toolbar the following happens in the procedure OnDragSimultaneousSeqences:
The variable m_bSimSeq is set to true if m_bDestJump and m_bDestBranch is false.
Then all other drag-and-drop variable is set to be false. On the same way as new step with transition the procedures OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) is executed.

If the user presses the left mouse button on a selected object the following happens in the procedure OnLButtonDown(UINT nFlags, Cpoint point):
The variable m_bSimSeq is true and therefor a new simultaneous sequence is placed in the net. It is then possible to place another simultaneous sequence.

When the user presses the right mouse button and release it the procedure OnRButtonUp(UINT nFlags, Cpoint point) is executed. The following happens for new simultaneous sequence:
The variable m_bSimSEq is set to be false, which means that the drag-and-drop object disappear. In the same way as new step with transition the procedures OnMouseMoveDeselect(false) and OnMouseMoveEvent(false,false) is executed.

5.3.9 New jump

In the file *SFCView* the following happens:
When the user presses the symbol for new jump the procedure OnDragJump is executed. The following happens:
The variable m_bJump is set to true if m_bdestJump and m_bDestBranch are false.
Then all other drag-and-drop variable is set to be false. The procedure DragJumpEvent(true) is then executed, it sets the variable m_bJumpDrag to true in the file *Editseq*. This is done so that a procedure in *Editseq* shall work both by the old toolbox and the new drag-and-drop toolbox.
In the same way as for new step with transition the procedures OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) are executed.

If the user presses the left mouse button on a selected object the following happens in the procedure OnLButtonDown(UINT nFlags, Cpoint point):
The variable m_bJump is true and the variable m_bDestJump is set to true so that another drag-and-drop command not is executed before a jump destination is decided. After that it is controlled what kind of element the user has clicked on. If the user has clicked on a step with simultaneous wires will a help-text pop up and the program waits in 4 s so that the user understand the message. After that the procedure is interrupted, the user has to click on another element to place the jump. This is done to prevent the program from getting locked.
If there is not a simultaneous wire after the step that the user clicked on the program will call the file *Editseq.* There the following happens:

If the user has clicked on a step, two alternative transitions will be placed after it.
If the user instead has clicked on a transition an alternative transition will be created parallel to it.

The transition that has been created will be selected and be sent in the procedure SeqElemTurnIntoJumpAction(pSeqElem eSEqElement). There the following modifications have been done:
First the procedure OnMouseEvent(false,true) is executed. If this is not done the command WaitFor(DoubleClickEvent,100000,&Event) will not work. It will wait for MoveEvent, which happens when the cursor is moved. The boolean true is sent in the procedure to prevent that the selection frame, that is dragged over elements, will pop up. WaitFor is waiting in 100000 ms for the user to double click.

When the user has double clicked the JumpSelection(Event,true) is executed, which is a new procedure with selection commands. The Boolean true is sent in so that it works properly. In the branch the same procedure is used but then false is sent in.
Then the element that the user double clicked on is controlled.

If the user has double clicked on a transition a help-text will pop up. A help-text will also pop up if the jump goes out of or into a simultaneous sequence. If one of these faults happens the program will wait for a new destination of the jump.

If the user double clicks on the background or when 100000 ms have proceeded a dialog-box will appear where the destination has to be written. After that OnMouseMoveEvent(true,true) is executed, which means that the selection is working for drag-and-drop again.

When the program returns to the file *SFCView* and the procedure OnLButtonDown, m_bDestJump will be set to false and a new jump or another drag-and-drop command can be executed.

If the user right clicks the procedure OnRButtonUp(UINT nFlags, Cpoint point) in the file *SFCView* will be executed and there the variable m_bRightClick will be set to true. After that OnMouseMoveEvent(true,true) is executed. This means that if the program is waiting for double click in the procedure SeqElemTurnIntoJumpAction in the file *Editseq* it will also react on MoveEvent. Waitfor is passed when the cursor is moved and no jumpadress is placed. When the program has returned to the procedure OnLButton in the file *SFCView* and m_bRightClick is true m_bDestJump and m_bJump will be set to false. This means that the drag-and-drop object disappears. In the same way as new step with transition the procedures OnMouseMveDeselect(false) and OnMouseMoveEvent(false,false) is executed. The variable m_bRightClick is then set to false.

5.3.10 New Branch

In the file *SFCView* the following happens:
When the user presses the symbol for new branch the procedure OnDragBranch will be executed. There the following happens:
The variable m_Branch is set to true if m_bDestJump and m_bDestBranch are false.
Then all other drag-and-drop variable is set to the value false. On the same way as new step with transition the procedures OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) is executed.

When the user presses the left mouse button the procedure OnLButtonDown(UINT nFlags, Cpoint point) is executed. There the following happens:
It is controlled if an element is selected. The variable m_bBranch is true and the variable m_bDestBranch is set to true so that no other drag-and-drop commands can be executed until a branch destination is decided.

After that the procedure SeqElemNewDragBranchAction(eSEqElem, eDummy, SiSoType,TiToTpe,Before) in the file *Editseq* is executed. This procedure has been added so that a branch between several elements shall work. There the following happens:
It is controlled how many elements that is selected. If no element is selected the procedure shall not be executed, the program will get locked. If an outgoing element is selected so the procedure OnMouseMoveEvent(false,true) is executed. This is done so that Waitfor(DoubleClickEvent,100000,&Event) shall work properly. It will else react on MoveEvent, which was explained in jump.

After that is the whole net unselected so that a branch destination could be selected. The procedure WaitFor is waiting in 100000 ms for the user to double click. If the user has double clicked the procedure JumpSelection(Event,false) is executed, which is a new procedure with selection commands. The selected element is then fetched. It should be an element and it should be of the same type as the element that the user first has selected. If the elements are of different types a help-text will pop up and the user must select a new element for the destination. Then it is controlled if the outgoing and the destination element are the same element. If that is the case the procedure SeqElemNewBranchAction is executed. If the elements are of different types it is controlled so that the branch does not go into a simultaneous sequence. In that case a help-text will pop up and the user has to double click on another element. After that it is controlled so that the user has chosen a branch destination that is over the outgoing branch in the net. If that is the case the pointer of branch destination and outgoing branch will change values. This is done because the procedure ConSeqElemNewBranchAction that is used must have the elements in correct order, or else the program will get locked.
Then OnMouseMoveEvent(true,true) is executed, which means that the selection works for drag-and-drop again.
When the program returns to the procedure OnLButtonDown in the file *SFCView* the variable m_bDestBranch is set to false and a new branch or another drag-and-drop command can be executed.

If the user right clicks the procedure OnRButtonUp in the file *SFCView* is executed. There the following happens:
The variable m_bRightClick is set to true. If a branch destination is decided is m_bBranch set to false, which means that the drag-and-drop object disappears. If the program instead is

waiting for a double click in the procedure SeqElemNewDragAction in the file *Editseq* will m_bDestBranch be true. That means that OnMouseMoveEvent(true,true) first is executed so that the procedure WaitFor shall react on MoveEvent and no branch will be placed. The program then returns to the file *SFCView* and the procedure OnLButtonDown. There will m_bDestBranch and m_bBranch will be set to false, which means that the drag-and-drop object disappear.
In the same way as new step with transition the procedures OnMouseMoveEvent(true,true) and OnMouseMoveDeselect(true) is executed.
The variable m_bRightClick is then set to the value false.

6 Conclusions and Continued Work

6.1 Conclusions and Experiences

The conclusion of the implemented program is that the construction of SFCs is more flexible and easier to understand then the existing program. The drag and drop figures are not ultimate but the user is helped by the help-texts.
The implemented program works reliably and the program did not get locked during our tests.

We spent a lot of time at searching and examination of similar programs in the beginning of the master thesis. Afterwards we realise that the examination may have been not completely adequate. The interesting facts to find out were the editing principle and the graphics. The ways that the programs worked were of much less interest.

We had no experience on object programming before the master thesis. Therefore we spent more than one month part-time practising Microsoft Visual C++ programming. This was very instructive to the following implementation programming.

It is not easy to get acquainted with the existing ABB software. It is a large, complex program with several processes. There are some files from 1985. After that new files have been added and modified several times. There is not much help text in the files from the programmer, who is usually mentioned in the file. In several cases the programmer of a file is not working with ABB anymore and no one else is really familiar with the file. We started to do small changes in the program to understand the principle of the program. It is not possible to understand every detail in the program and it is not necessary.

The order of the functions we have implemented has been adequate. We started to add the selection principle of drag and drop, which is fundamental for its functionality.

The first function we added was new step with transition. This forms the basis for the next functions, new simultaneous sequence and new sequence selection.

The functions jump and branch were the most difficult to implement. The final time was spent at help-texts and drag and drop objects adjustments.

We had only one computer during the most part of master thesis. Therefore we decided to concentrate us on the implementation and wait to write the report. This has lead to that much time after the implementation has been spent at report writing. Afterwards it can be discussed if we spent too much time at the implementation. We think that it was the most important and it had to be finished.

6.2 More Functions

In Section 5.2 the difference with the function-specification is mentioned and the functions are described in more detail. There are still some functions that remain to be installed in the future, such as:

- The cursor has to change appearance to a hair-cross.
- The cursor does not have to cover the wire between elements where a drag and drop object has to be placed.
- The menus have not been modified according to the function specifications.
- The destination of the branch and the jump is decided with double-click in the implementation.
- An Undo-function that will undo the last command. It should work in several steps.
- Keyboard accelerators for the most used commands.

6.3 Other Modifications

- One idea is that the top wire of the drag and drop object should attach to the SFC net when the starting point of the branch has been decided. The length of the wire is adjusted when the drag and drop object is moved. The branch is attached to the SFC net when ending point of the branch is decided.
- Parts of the SFC net should be saved as a file in a subsequence library. The files can then be loaded and then be pasted in the SFC net. It could for instance be a drill process in a factory, which can be used as a module for the complete control program.

References

SattLine language Development version 0.31-n
SattControl AB
Malmö, Sweden 1991
Order no: 493-0298-11

Programming Microsoft Visual C++ Fifth Edition
Microsoft Press
Washington, USA 1998
ISBN: 1-57231-857-0

Computer Systems for Automation and Control, Second Edition
Gustaf Olsson and Gianguido Piani
Industrial Electrical Engineering and Automation
Lund Institute of Technology, Sweden 1998

Committee Draft IEC 61131-3, 2nd Ed.
Programmable Controllers Programming Languages
ABB Satt AB
Sweden 1998-11-18