

**AN INTEGRATED SIMULATION AND CONTROL
IMPLEMENTATION ENVIRONMENT FOR
HVDC SYSTEMS**

by

Michael Hua Xie

A thesis
presented to the University of Manitoba
in partial fulfilment of the
requirement for the degree of

Master of Science

in Department of Electrical and Computer Engineering
University of Manitoba
Winnipeg, Manitoba

(c) May 1997



**National Library
of Canada**

**Acquisitions and
Bibliographic Services**

**395 Wellington Street
Ottawa ON K1A 0N4
Canada**

**Bibliothèque nationale
du Canada**

**Acquisitions et
services bibliographiques**

**395, rue Wellington
Ottawa ON K1A 0N4
Canada**

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-23556-4

**THE UNIVERSITY OF MANITOBA
FACULTY OF GRADUATE STUDIES

COPYRIGHT PERMISSION PAGE**

**AN INTEGRATED SIMULATION AND CONTROL IMPLEMENTATION
ENVIRONMENT FOR HVDC SYSTEMS**

BY

MICHAEL HUA XIE

**A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University
of Manitoba in partial fulfillment of the requirements of the degree
of
MASTER OF SCIENCE**

Michael Hua Xie 1997 (c)

**Permission has been granted to the Library of The University of Manitoba to lend or sell
copies of this thesis/practicum, to the National Library of Canada to microfilm this thesis
and to lend or sell copies of the film, and to Dissertations Abstracts International to publish
an abstract of this thesis/practicum.**

**The author reserves other publication rights, and neither this thesis/practicum nor
extensive extracts from it may be printed or otherwise reproduced without the author's
written permission.**

Acknowledgements

Foremost, the author wishes to convey his sincere gratitude to Professor Ani. Gole for all his counsel, guidance, patience and especially his encouragement throughout the course of this thesis and the Master's program here at the University of Manitoba.

The author would also like to thank Mr. Dennis Brandt from Brandt Consultant Inc., Mr. George Wild, Mr. Hans Messna, Mr. Armin Moosburger and many Siemens engineers, for providing the opportunity to work on such an interesting project and for providing financial and technical support to conduct various experiments.

Next, sincere thanks goes to everybody at Power Tower for providing assistance.

Abstract

A control system for the control of HVDC, SVC and other FACTS devices can be developed, tested and debugged using an electromagnetic transients simulation program. The thesis presents a procedure for the direct conversion of this developed emtp-type model into the control language for a real world implementation. The procedure is validated by comparing the simulated emtp-type output with that from the real- world control system automatically derived from this code. The procedure has been implemented on the PSCAD/EMTDC emtp-type program and produces control software that can be loaded on to the SIMADYN-D control system. The benefits of this capability are to achieve faster design of complex controls with greater accuracy and reduced engineering costs.

Keywords: HVDC, SVC, FACTS, Digital Control Systems, Electromagnetic Transients Simulation

Table of Contents

Acknowledgements i

Abstract ii

Table of Contents iii

1. Introduction 1

- 1.1 SYMADYN-D control system 2
- 1.2 PSCAD/EMTDC Simulation program 6
- 1.3 Objectives 9

2. Comparison of SIMADYN-D STRUC-L and PSCAD DRAFT files 12

- 2.1 The Analysis of SIMADYN-D STRUC-L Language 15
 - 2.1.1 *The Overview 16*
 - 2.1.2 *The Syntax 17*
 - 2.1.3 *The Information Part 19*
 - 2.1.4 *The Connection Part 20*
- 2.2 The PSCAD DRAFT Module 21
 - 2.2.1 *Draft Graphical User Interface 21*
 - 2.2.2 *The Syntax of Draft File 23*
- 2.3 Comparison of STRUC-L and DRAFT file 27
- 2.4 Tasks 31

3. Illustration of Algorithms 33

- 3.1 The Object Model 34
 - 3.1.1 *Node Map 37*
 - 3.1.2 *The Pin Sequence 38*
 - 3.1.3 *Library List, Modifier Map and Special Info. List 40*
- 3.2 Functional Model of the Program 41
 - 3.2.1 *Top Level 42*
 - 3.2.2 *The Second Level 44*
 - 3.2.3 *The Collapsing Algorithm 46*

4. Component Library 49

- 4.1 The Component Library 50
- 4.2 The Creation of Component Library 56
- 4.3 Windows GUI for Library Creation 58

5. Validation of the PSCAD to STRUC-L Compilation Software 61

- 5.1 Test No. 1, Wave Form Generator 63
- 5.2 Test No. 2, Part of an HVDC Inverter Side Controller 74

6. Conclusions 78

- 6.1 Recommendations for Further Work 80

Appendix I: Draft file for Test No. 1 83

Appendix II: STRUC-L file for Test No.1 99

Appendix III: Draft File for Test No.2 104

Appendix IV: STRUC-L File for Test No.2 153

Appendix V: Header Files 158

References 173

CHAPTER 1

Introduction

Simulation using electromagnetic transient programs (generally called emtp) is gaining wider application in studies concerning the performance of Power-Electronics and Controls. Such studies include High Voltage DC Transmission Static VAR Compensation Systems and Flexible AC Transmission Systems (FACTS). Several earlier authors have developed detailed models for the simulation of HVDC and SVC (Static VAR Compensator) Control Systems [1,2,3]. One reason for this trend is the increasing confidence in the models for typical control blocks available in the real system. It is indeed possible to have an exact representation of the manufacturers' control blocks in an emtp-type program. One such implementation is that of the Siemens SIMADYN-D control library which has been exactly represented in the emtp-type program PSCAD/EMTDC [7].

Present day methodology requires that after the successful completion of the control and protection simulation studies, the optimized control topology as well as the controller parameters have then to be included in the actual hardware implementation. This requires great care, particularly when the control systems are very large and have an extremely large number of gains, limits, time constants and so on. It would therefore be more convenient if the actual hardware installed in the real plant be directly set from the emtp-type program. This is possible today, because of the use of digital controls which are programmable via software. This thesis will present a prototype system that has been developed which allows the generation of software for the Siemens SIMADYN-D Control System from PSCAD/EMTDC. Thus, when the desired control system design has crystallized after the completion of several simulation studies, the designed control system can be directly implemented in the real controls.

1.1 SYMADYN-D control system

The SIMADYN D control system [5] is a multi-processor system for fast closed-loop control and arithmetic operations, open-loop control and monitoring and for signalling and logging. The programmable SIMADYN D control sys-

tem is used by Siemens for all HVDC, SVC and FACTS applications [6] where it is utilized both for control and protection functions. The whole system can be divided into hardware and software parts.

At the hardware level, the multi-processor control system is made up of various plug-in boards optionally configured in a rack for HVDC and SVC type applications. Several different processor boards can be used depending the application and peripheral interface requirements with up to eight processor boards in one sub-rack. Each processor board has its own program and data memory and executes its allocated tasks independent of other processors. The program memory sub-module inserted in each processor board contains the system and application software. The parallel processors communicate via a local bus through a communication buffer board. Input/output boards interface the processors to external systems and signals with normalized input and output levels. Interface sub-modules which can be inserted in the processor board provide a standardized serial interface (20 mA, V.24 or EIA 485) for data exchange with other equipment and systems. Separate Communication boards are also used for serial data exchange if required.

As for the software part, the programming of control or protection functions, including all control parameter settings, is carried out using a high level programming language called STRUC-L, with function blocks available from an extensive library of control functions, which will be described and analysed in more detail in the following chapter. The library consists of control blocks, converter specific blocks, arithmetic blocks, logic and switch blocks, and input/output blocks. Execution time step of these function blocks from 100 ms and up are assigned according to the response time requirements of the particular control loop or protection application. Up to five different sampling times can be assigned to each processor. The valve firing system ("trigger set") and extinction angle measurement are specially created function blocks for HVDC applications with corresponding specific high-speed processors that ensure firing pulse and extinction angle accuracy of $\pm 0.1^\circ$. The software also includes the communication functions to handle data transfer within the SIMADYN D system and with external systems. Diagnostic software identifies errors/faults in the SIMADYN D system and provides external signals to allow the appropriate contingency actions such as switch-over to a redundant control channel. The software is self documenting and the control structure is directly entered using a graphical user interface (called

STRUC-G) as in Fig. 1-1. On compiling, STRUC-L code is produced as shown in Fig. 1-2.

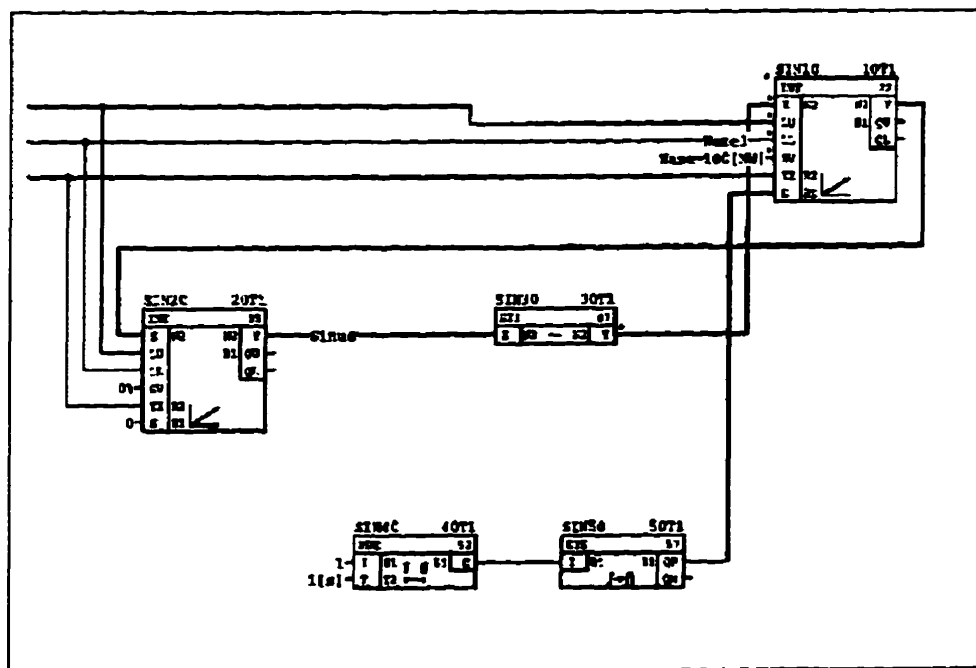


Fig. 1-1 STRUC-G Representation

```

^ 1 FP-SINUS.P16
8 W.2.1 FP-DIA UNC 08.07.96 08:54
8 F8SLI8 950401W20 A:
8 B: C:
880005563
2 CT1 45 = 'test "Inhalt/Content"
3 CT2 45 = "
4 CT3 45 = "
5 HLD 35 = " "Anlagenkennz./Higher lev.design"
6 DLS 15 = " "Zeichn.Nr./Draw. No List suffix"
7 DGS 15 = " "Graphic suffix"
8 DPS 15 = " "Besteller/Purchaser suffix"
9 DES 25 = 'gast "Bearbeiter/Designer"
10 ORD 45 = " "Urspr./Original document"
11 MD3 45 = " / / /10-21/96/ "Modification"
12 MD2 45 = " / / /
13 MD1 45 = " / / /
14 *
15 TX=T1
16 SIN10 : INT .POS=01.01.01
8800400018 2001G
17 X N2 < SIN30.Y
18 LU N2 < $LUP
19 LL N2 < $LLOW
20 SV N2 < 100%
21 TI R2 < $TSIN
22 S B1 < SIN50.QP
23 Y N2 >
24 QU B1 >
25 QL B1 >
26 *
27
28 SIN20 : INT .POS=01.01.01
8800400018 2001G
29 X N2 < SIN10.Y
30 LU N2 < $LUP
31 LL N2 < $LLOW
32 SV N2 < 0%
33 TI R2 < $TSIN
34 S B1 < 0

```

Fig. 1-2 STRUC-L listing

1.2 PSCAD/EMTDC Simulation program

PSCAD is a graphical front-end for the emtp-type simulation program EMTDC. It allows the user to make schematic drawing of control circuits using

one of its modules called DRAFT. In DRAFT, the complete system (network as well as controls) can be divided into many sub-systems. This allows one to represent the controls in a modular fashion, with each subsystem representing one specific functionality. Every sub-system is a drawing board on which many PSCAD components can be drawn as in Fig. 1-3. Each PSCAD component has a definition file defining its graphical appearance and underlying FORTRAN simulation code. The data for the component can be entered by filling in the pop-up menu that appears when the component is clicked on with the mouse.

In addition to the vendor-supplied components, users can also compose their own components [4]. Using this approach a comprehensive library has been developed and tested for the SIMADYN D system [7]. Fig. 1-3 shows one subsystem of the PSCAD DRAFT module, in which a SIMADYN-D wave form generator is simulated.

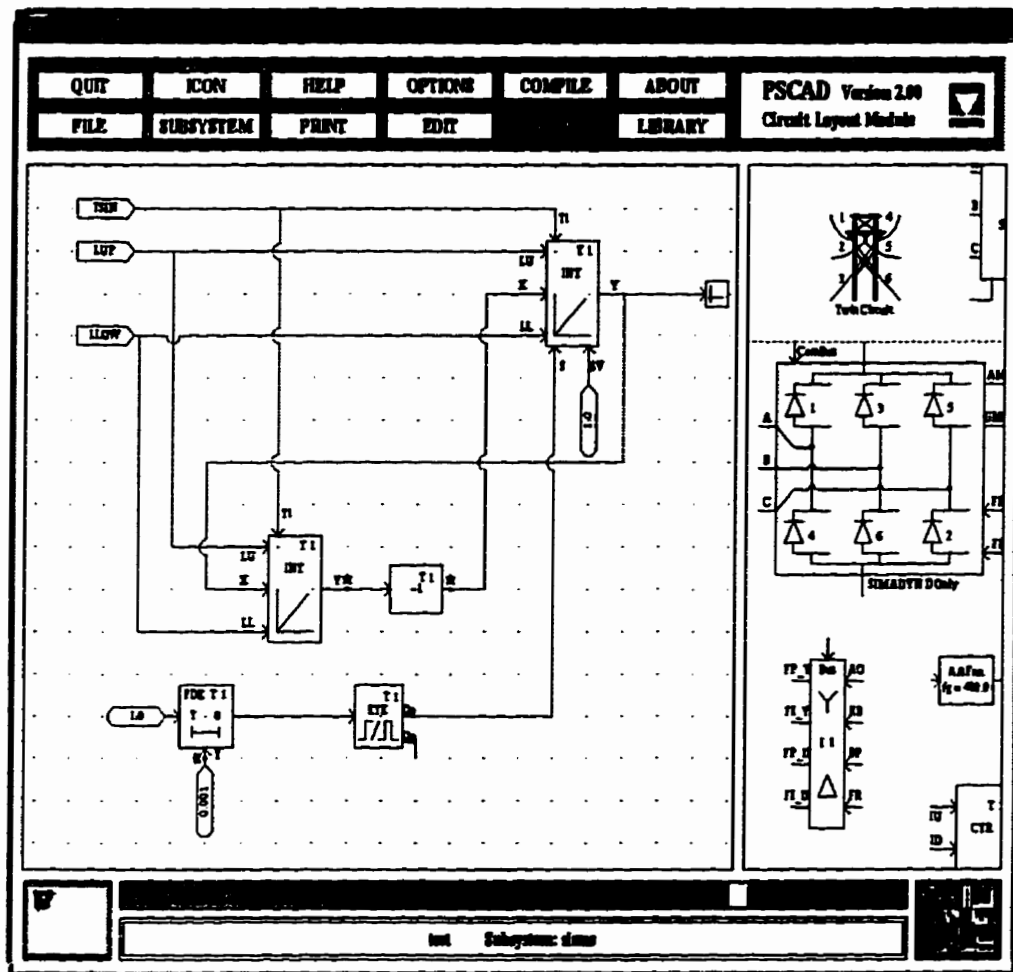


Fig. 1-3 PSCAD Draft Layout

The circuit developed in DRAFT is compiled and run through an operator console type interface called RUNTIME. This interface allows the user to

make changes to the run while it is in execution, in much the same manner as a operator at a system control console[4]. Gains and set-point can be varied on line, switches opened and closed, system quantities can be read on meters or plotted on graphs and so on.

1.3 Objective

Since the real control system contains many implementation details, it is hard to predict the system behaviour before the control system is built. On the other hand, simulation programs like PSCAD use differential equation methods to simulate the system behaviour, which include the modeling of the physical electrical network in addition to the control system. Thus, the objective of this thesis is to provide a computer program that can automatically convert the simulation scheme into real control code, which can be uploaded into control hardware later.

The actual implementation concept is shown in Fig. 1-4. The SIMADYN-D system is programmed using a language called STRUC-L in which the controls are defined. This language is then compiled into assembly within SIMADYN-D and the resulting assembly code used in the DSP based control hardware. For ease of data entry, SIMADYN -D provides a graphical interface (STRUC-G) to STRUC-L. The simulation software PSCAD/EMTDC, on the other hand, also

has a graphical interface (DRAFT) to generate the data files and the FORTRAN code necessary for running the actual EMTDC program. The run itself is fully interactive via the RUNTIME module, in a manner similar to analog simulators[4].

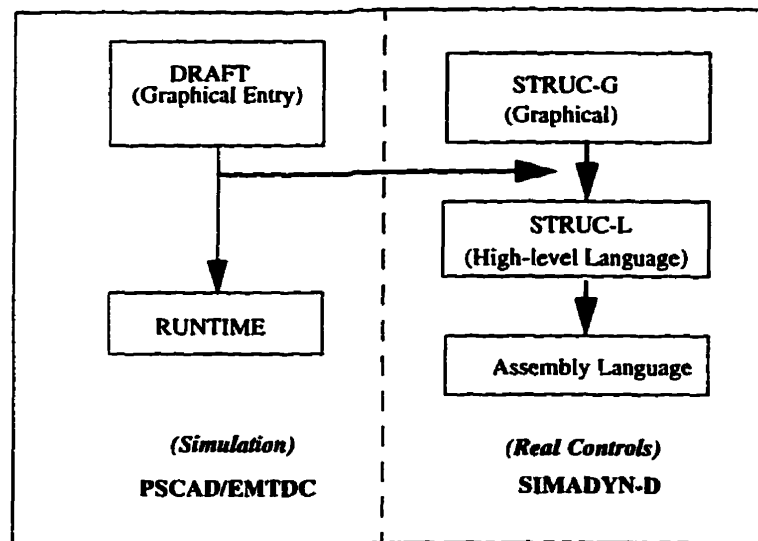


Fig. 1-4 Interfacing PSCAD with SIMADYN-D

The link between the real controls and the simulation software is made between DRAFT and STRUC-L. The DRAFT program already contains detailed models of the blocks of the SIMADYN-D library. These blocks, when compiled, generate FORTRAN code for PSCAD/EMTDC. An additional control switch can

easily be incorporated into these blocks, so that instead of generating FORTRAN code, they generate STRUC-L. The connection between blocks is already defined by the connections shown on the DRAFT palette and is the same for the simulation as well as the real implementation. In a similar manner, STRUC-L code could also be read from the real controls and implemented in PSCAD/EMTDC, although this feature is not yet implemented.

This thesis will show a simple implementation to interface PSCAD with SIMADYN-D using basic control building blocks.

***Comparison of
SIMADYN-D STRUC-L and PSCAD
DRAFT files***

In Chapter 1, the basic idea of integrating control generation with simulation software has already been explained in brief. In order to realize this integration, the following 3 aspects of the control system and its simulation representation need to be analysed, (1) determining what information is contained in the simulation system and in what format is this information presented, (2) determining what information is contained in the real control system and its format, (3) determining the relationship of the above two systems, namely determining what information is common for both systems and what information is extra for each system. This relationship is shown in Fig. 2-1, where the intersection of the two circles is the part of the control system that can be generated directly from the simulation schemes.

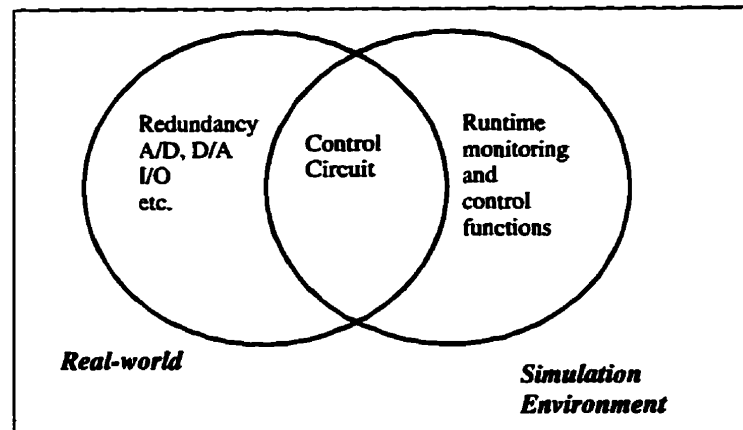


Fig. 2-1 Control Representation in Simulation and in the Real-world

In general, the relationship between the real-world controls system and its simulation representation can be summarized below:

- (1) As for the overall system behaviour and the control circuit structure, the simulation model is an exact representation of the real-world model. All the connections, key components and their parameters are accurately represented in the simulation scheme.
- (2) In the real-world implementation, there are many additional details, such as analog to digital conversion interfaces,

redundant modules and input/output functions, which are simplified in simulation. Since this additional information is not included in the control scheme, it will have to be added to the real system at a later stage.

- (3) In the simulation system, there are a few functions mainly used to monitor or control the simulation process. There is no need to include this part of the simulation scheme into the real-world control system. They could simply be ignored.

Thus, a border can be drawn within the simulation system, to distinguish those functions that are to be compiled into real controls, from those which are not to be compiled.

Since the PSCAD/EMTDC simulation software provides a very convenient subsystem model, the most effective method to define this border within the simulation is to partition sub-systems according to their functions. In the simulation process, attention will be paid to categorize the components, so that those components that do not need to be used in the real control system will be put into separate subsystems, later on these subsystems will be marked and filtered out in

the compilation process, so that only those subsystems that are relevant will be converted into real control code and downloaded to the control board.

2.1 The Analysis of SIMADYN-D STRUC-L Language

In this section, the content and the syntax of the STRUC-L language will be analysed. Figure 2-2 shows a piece of sample STRUC-L code for a saw-tooth generator package, which is used for discussion purpose in the latter part of the thesis.

```

1 PP-SAW,P16
EV4.2.1 PP-DIA UMC 08.07.96 08:54
& FBELIB 950401V420      A:
&B:                          C:
&&0009239
2CT1 4S = 'test'           "Inhalt/Content"
3CT2 4S = '                '
4CT3 4S = '                '
5HLD 3S = '                ' "Anlagenkennz./Higher lev.design"
6DLS 1S = '                ' "Zeichn.Nr./Draw. No List suffix"
7DGS 1S = '                ' "                Graphic suffix"
8DPS 1S = '                ' "    Besteller/Purchaser suffix"
9DBS 2S = 'gast            ' "Bearbeiter/Designer"
10ORD 4S = '                ' "Urspr/Original document"
11MD3 4S = '                /                /10-02/96/ "Modification"
12MD2 4S = '                /                /                /
13MD1 4S = '                /                /                /
14+
15 TX=T1
16 SAW10      : INT      ,POS=01.01.01
&&004D0018 2001G
17X  N2 < $WSAW
18LU  N2 < 100%
19LL  N2 < -100%
20SV  N2 < 0%
21TI  R2 < $TSAW, 'Speed'
22S  B1 < 0
23Y  N2 > ,SCAL=1000 [MM]
24QU  B1 >
25QL  B1 >
26+
27
28 SAW20      : RSS      ,POS=01.01.01
&&0016000C 2004K
29S  B1 < SAW10.QU
30R  B1 < SAW10.QL
31Q  B1 > $CDOWN
32QN  B1 >
33+
34
35 CM,POS=01
36 "Saw tooth generator"
37 END
38

```

Fig. 2-2 An example STRUC-L code listing

2.1.1 The Overview

Each complete SYMADYN-D control system consists of several “function packages”. Each function package concentrates on a specific task such as control, protection or communication. The total number of function packages allowed varies depending on the processor types of the underlying hardware.

Every function package exists in the format of a single file in the DOS/Windows environment. Before the SIMADYN-D compiler compiles the file, it has a suffix of “ufp”, which means “uncompiled function package”. After the compilation, an “.ofp” file will be generated. In this thesis, the “.ufp” files will be used as the entry point to the STRUC-L language, since they contain all the necessary circuit information of the control system, but do not have too many machine dependent implementation details.

The underlying file structure is identical for both STRUC-L and STRUC-G environment. In particular, the information included in the STRUC-L data also contains the graphical and the interconnection data. Thus when a STRUC-L file is loaded into STRUC-G, a fully graphical rendering of the system is obtained.

2.1.2 *The Syntax*

The STRUC-L language has a very strict syntax requirement. Any syntax error, as little as a missing space character can result in "System Error", which ultimately leads to the failure of compilation. This is the reason why special attention has to be paid to study the file syntax and formats.

Additionally, a "checksum" utility program has been provided by Siemens AG. This program has to be run before a ".ufp" file is loaded into the SIMADYN-D environment. The program checks the ".ufp" file with a special checksum algorithm and generates a checksum code, then adds the code in the ".ufp" file, so that the correctness of the ".ufp" file format is ensured before it is loaded into SIMADYN-D system.

Each STRUC-L file can be divided into two parts, the information part and the connection part. The first part, the information part, occupies line 1 through line 14. The second part occupies the rest of the file.

As shown in Fig. 2-2, each line of the STRUC-L file begins with a line number. The line number occupies the right most digits of a four-digit space. Also,

a special character needs to be written at the end of the file to indicate the termination of STRUC-L. This special character is ASCII 23 (Decimal).

If a piece of information is too long to fit in one line, it can be wrapped to the following line, provided an ampersand symbol is put in column 4 to indicate line continuation. As shown in Fig. 2-2, those lines start with double ampersand symbols are generated by the checksum program.

2.1.3 *The Information Part*

The information part of the STRUC-L specifies the general information for this function package, the content of this part is shown in Table 2-1.

| Line Number | Description |
|--------------------|--|
| 1 | File Name, save time, library name, library version. |
| 2 | Content description line 1 |
| 3 | Content description line 2 |
| 4 | Content description line 3 |
| 5 | Higher level designer. |
| 6 | Drawing Number suffix |
| 7 | Graphics suffix |
| 8 | Purchase suffix |

| Line Number | Description |
|--------------------|-----------------------------------|
| 9 | Designer |
| 10 | Original document |
| 11 | Modification note and date line 1 |
| 12 | Modification note and date line 2 |
| 13 | Modification note and date line 3 |

Table 2-1 STRUC-L file information

2.1.4 The Connection Part

The connection part of the STRUC-L specifies the function block parameters and their connections within this function package. Each function block in the package is listed in this part, by the sequence of their sampling times. Within the same sampling time, the function block that has the lower execution sequence number is listed first.

For each function block listing, the first line specifies the application name and the function block type. Optionally, a position string can be put at the end of the line to indicate where in the STRUC-G representation this function

block will appear. The second line of the function block listing is some code generated by the checksum program. This code is unique for each distinctive function block.

After the check-sum code, each connector of the function block is listed. In each line, the connector name, connector type and the connection destination is listed accordingly. Optionally, the comments and some other types of modification labels can be put at the line end. The function block listing ends with a plus sign occupying one line. An “End” sign needs to be put at the end of the whole function package listing file.

2.2 The PSCAD DRAFT Module

2.2.1 Draft Graphical User Interface

PSCAD is a graphical front end for the simulation software EMTDC. There are several interfacing points between PSCAD and EMTDC. One such point is the Draft file.

Whenever the user wants to do a simulation for a certain case, he would first use the PSCAD front end to compose a simulation scheme, because PSCAD provides a very user-friendly graphical interface on the computer terminal to help users laying out all the system components. Fig. 2-3 shows an example system in the PSCAD Draft module.

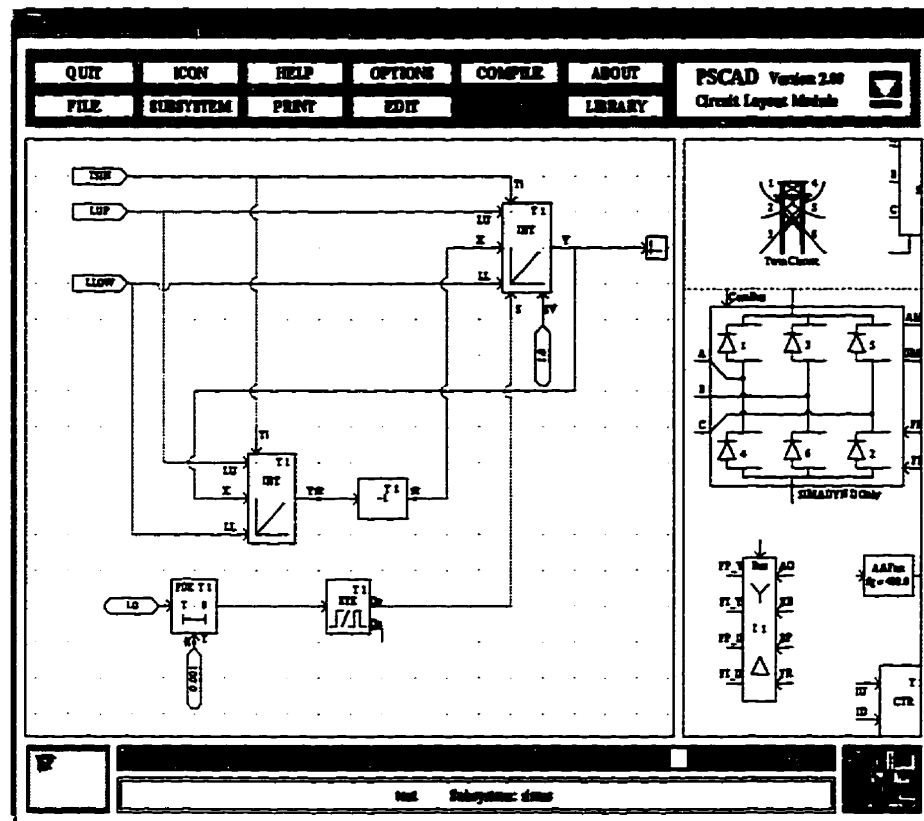


Fig. 2-3 PSCAD Draft module

After the system is laid out on the PSCAD Draft module, the graphical information will be collected and saved into the Draft file. Later on, when the “compile” option in PSCAD menu is chosen, the Draft files are compiled into EMTDC data format required for simulation. Our approach here is to take the

PSCAD Draft file, which contains all the component layout information about the system, and then generate STRUC-L code for the real control system.

As mentioned before, the PSCAD Draft module allows the user to define several sub-systems in a simulation case, those subsystems that do not need to be compiled into STRUC-L code are marked as “non-structl” subsystems and filtered out during compilation. For the rest of the subsystems, each of them will be mapped into a single function package file.

2.2.2 The Syntax of Draft File

An example of PSCAD Draft file (generated via the DRAFT GUI) is shown in Fig. 2-4.

```
XDRAFT Version 4.2.2
EMTDC
TITLE: test
CREATED: May 03, 1996 (gast)
LAST-MODIFIED: October 21, 1996 (michael)
TIME-STEP: 0.001
FINISH-TIME: 50
PRINT-STEP: 0.01
RTDS-RACK: 0
RTDS REAL-TIME: Yes
3 SUBSYSTEMS
SUBSYSTEM-TITLE: Para
SUBSYSTEM-COMMENT: Adjustment parameters.
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
46 COMPONENTS
export
656 208 2 0 1
Name:TSAW
sumjcts
496 208 0 0 8
T_Na:T
T_No:1
DPath:1
B:0
AName:SAW20
ProSeq:10
FbCom:Ramp control
StgPos:
WIRE
400 208 0 0 0
-32 -4 64 4
WIRE
432 240 0 0 0
-64 -4 64 4
.....
```

Fig. 2-4 A Sample Draft File

The Draft file is organized according to sub-systems. The first 11 lines form the general information part. This part contains the following information: PSCAD version, circuit title, creator, modifier, creation/modification time, simulation time step and interval as well as the number of sub-systems.

Following the general information part is a listing for each sub-system. Within each sub-system listing, the first 4 lines constitute the sub-system information part, which includes sub-system title, subsystem comments, the print/plot mode and the number of components in this subsystem. After the subsystem information listing, each component in the subsystem is listed individually.

Most of the PSCAD components listed in the Draft file take the same format. The first line for the component listing is the component name. The second line is position information. There are five integers in this line, each of which has a special meaning for the component position, as illustrated in Table 2-2.

| Number | Name | Meaning |
|--------|---------|-------------------------------------|
| 1 | X-coord | X coordinate of the insertion point |
| 2 | Y-coord | Y coordinate of the insertion point |

| Number | Name | Meaning |
|---------------|-------------|--|
| 3 | Mirror | Mirror state: 0=non mirrored, 1=mirrored |
| 4 | Orientation | Orientation of the component: 0=no rotation, 1=90 degree clock-wise, 2=180 degree, 3=90 degree clock-wise. |
| 5 | NOP | Number of Parameters. |

Table 2-2 Position Information for PSCAD component

Following the position information line, each parameter and its value is listed. A colon separates the parameter name from the value, and each parameter takes one line.

The above format applies to all the PSCAD components with the exception of the "wire" component. Since the "wire" component in PSCAD is stretchable, it is necessary to specify the starting and ending position of any specific wire, in addition to the insertion point. So for the wire component, the first line is the same as above, while the second line has 4 integer numbers, which are the x and y coordinates of the starting and ending point for the line segment(x0, y0, x1, y1). Since the orientation of the wire is already determined by the first line

of text, only two numbers out of these four are actually useful, either x0 and x1, or y0 and y1.

2.3 Comparison of STRUC-L and DRAFT file Structures

We now compare the STRUC-L and PSCAD Draft file with a view to developing an automatic translator, the following observations can be made:

- (1) Both the STRUC-L language and the DRAFT file describe what the control system is composed of and how its components are configured through parameters. In PSCAD, the components take a more general form in order to ease the simulation work. For example, the “Logic Gate” component in PSCAD can be configured into an AND/OR/NAND/NOR gate, which are represented separately in the STRUC-L language. This relationship can be best represented by Fig. 2-5 below:

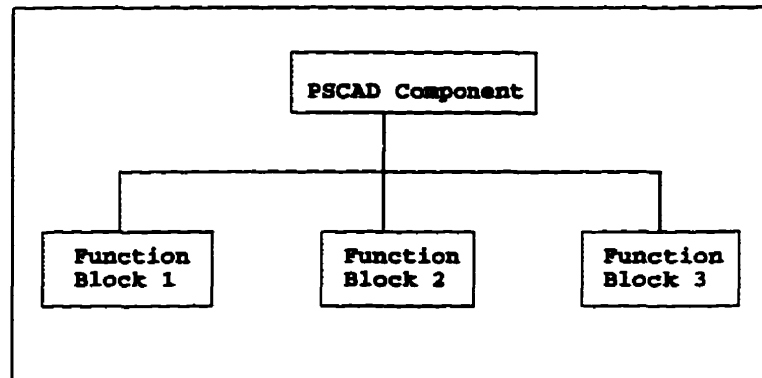


Fig. 2-5 PSCAD component vs. STRUC-L Function Block

(2) In STRUC-L, each input connector has to be defined. However, in PSCAD Draft file this is not necessary. Default values are assigned to some of the connectors if there is no explicit specification. The relationship between PSCAD connectors and STRUC-L connectors is illustrated in Fig. 2-6. Thus, some method will be needed to put the default values into the STRUC-L file. For example, if a default value is given in a DRAFT file, this value has to be retrieved and clearly specified in the STRUC-L file.

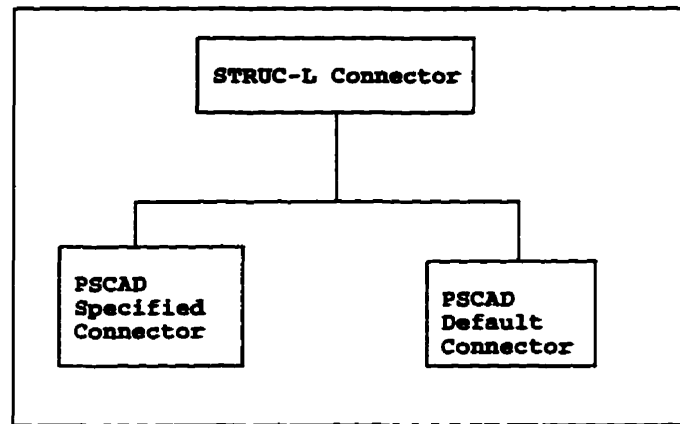


Fig. 2-6 STRUC-L and PSCAD Connectors

(3) In STRUC-L the connections are specified at each connector, while in Draft file, the connections are not directly specified. In Draft file, the electrical wire is also treated as an independent component and the exact co-ordinates are recorded for each component. Thus, some conversion will be needed to extract the connection information from the DRAFT file and convert it into STRUC-L format.

(4) There are many special components in the Draft file for monitoring and interacting with simulation, i.e. sliders, plot

selection symbols, etc. These component can be ignored when converting PSCAD scheme into STRUC-L.

(5) The Draft file header contains some general information about designer, date/time, modification, etc. This information can be used directly in the STRUC-L file information section. A mechanism must be developed to extract this information and convert it into the STRUC-L format. Also the sampling time information will be needed for the SIMA-DYN-D master program. This information will be extracted and printed out separately.

(6) Each PSCAD sub-system can be mapped to one STRUC-L function package and non compile subsystems will be marked and filtered out in the compiler.

The above six points outline the main methods to compile PSCAD Draft file into STRUC-L. In the following chapter, illustrations will be made to show how these methods are implemented in a computer program.

2.4 Tasks

In order to write a computer program to convert PSCAD Draft file directly into SIMADYN-D control code, following tasks need to be conducted:

- (1) define a border which includes the control components to be converted to SIMADYN-D (note that the electrical networks are not eligible for conversion);
- (2) It is sometimes the case that a particular PSCAD component can be mapped into several variants of the SIMADYN-D components; the mapping method needs to be described and this is done via a separate Windows95 program designed in this thesis. The actual conversion is carried out via a unix based program also designed by the author.
- (3) The PSCAD connections are defined by the user, whereas SIMADYN-D represents connections on the block. A unix based program was developed to carry out this process.

(5) The final results of above procedures is that a STRUC-L file is generated directly from PSCAD-Draft.

The following chapter will discuss the implementation of the various computer programs required to achieve the task objectives outlined above.

Illustration of Algorithms

In the previous chapter, the differences between the SIMA-DYN-D system and PSCAD systems has been analysed. Now in this chapter, the computer program that converts Draft directly into STRUC-L language will be developed and explained. This conversion process is referred to as the “compile” process hereafter, and the computer program is referred to as PSCAD to SIMADYN-D compiler program.

The actual algorithms and data structures employed will be studied in detail. But before proceeding further, it should be mentioned that the software in this project has been designed using the Object-Oriented methodology. Thus in this thesis, the main focus will be kept on how to solve the problem, rather than the coding details.

The notation used in this thesis basically follows the OMT notation[8].

The OMT methodology is a set of object-oriented rules and methods. Using such a methodology helps to analyse the problem in the early stage of a computer software design project. Each aspect of the problem will be analysed in an implementation-independent notation, so that after a few design cycles, the problem itself is studied thoroughly and the coding becomes relatively easy. All we need to do in the coding stage is to convert the notation of OMT design into the actual code.

The OMT methodology provides three models to analyse different aspects of a problem, in this thesis, the object model and functional model will be the main focus, since the dynamic model is trivial for any kind of compiler-type software.

3.1 The Object Model

The object model breaks a problem into several objects, each object contains its own methods and attributes. The public methods of an object define the external operation that could be performed on this object, the attributes are usually hidden from outside for the purpose of data encapsulation. Objects can be

abstracted to classes. A class is a generalization of the objects that have the same methods and attributes. In Fig. 3-1, the class diagram of this project is shown.

Each rectangle represents a class and the links between the classes represent an association between the them. The links can be easily implemented as pointers in C++ language. In the following sections, the attributes of each class will be described in detail.

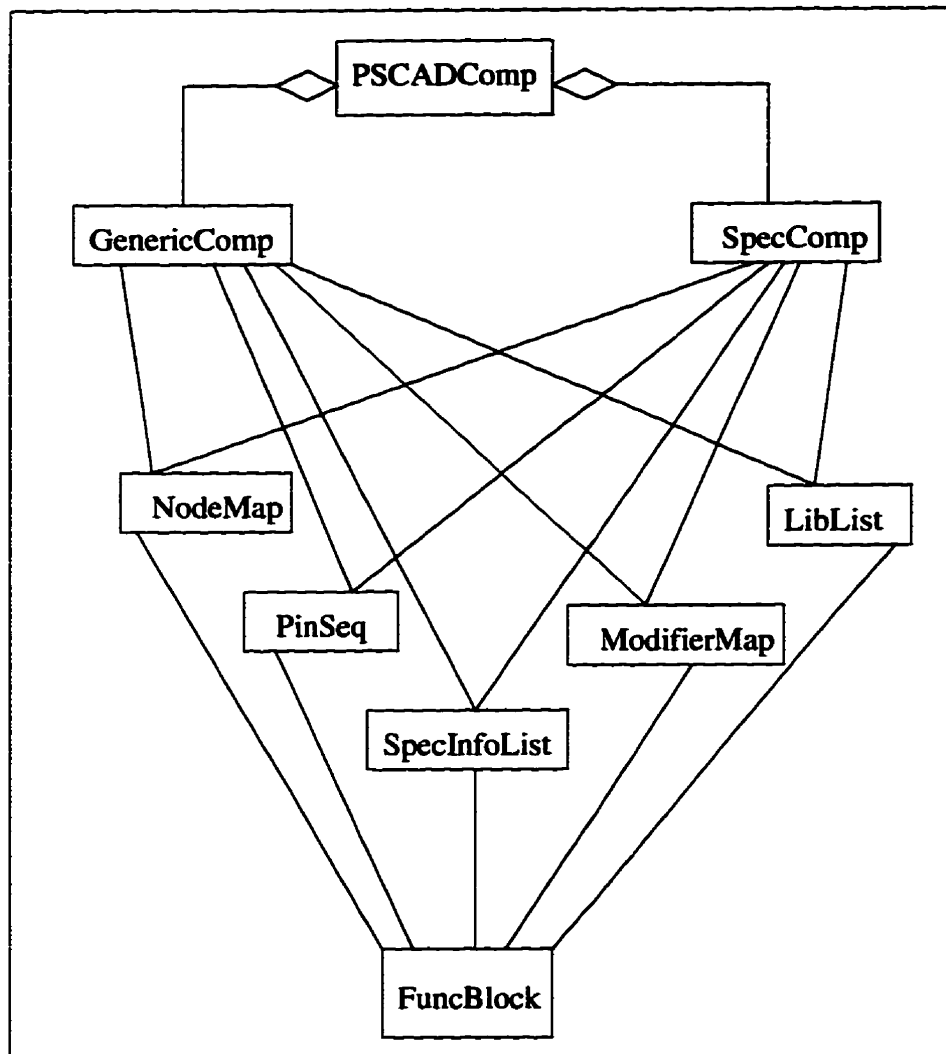


Fig. 3-1 Class Diagram of the PSCAD to STRUC-L compiler

In the class diagram shown in Fig. 3-1. The class “GenericComp” and “SpecComp” are both inherited from base class “PSCADComp” class. These classes will process the position and orientation information contained in the PSCAD Draft file, and put them into the objects which are instantiated from the following 5 classes: “NodeMap”, “PinSeq”, “LibList”, “SpecInfoList” and “ModifierMap”. The funcBlock class will then use the above 5 classes to generate the STRUC-L file. Next we are going to take a closer look at those 5 classes.

3.1.1 Node Map

The basic data structure of the Node Map is a dictionary, with the node name as its key and the node description as its information. The node name is generated automatically when the compiler program scans through the Draft file, the node name is unique for each node. The node description part contains all the useful information about this node. An explanation of the node description part is shown in Table 3-1.

| Name | Meaning |
|-------------|--|
| point set | collection of all points connected for this node |
| node type | type of node, ie.co-ordinate, constant in or external. |
| app. name | application name parameter |
| con. name | connector name |

| Name | Meaning |
|------------------|---|
| const. type | type of constant (integer, float or others) if node type = constant input |
| value | value of the constant |
| scale | scaling factor of the constant |
| unit | unit |
| ext. in name | external input connector name |
| ext. out name | external output connector name |
| node name base | the base part of the node name |
| node name add-on | the number part of the node name |
| node name | the node name string |
| collapsed2node | the node name that this node is connected to |
| pin pool | a list of connected pins |
| err flag | error flag |

Table 3-1 Node Information

The “NodeMap” class forms a connection between PSCAD simulation scheme and the STRUC-L language. It records all the nodes and their connected points in a data structure, so that later on, all the connected nodes can be grouped together by comparing the points each node contains. This is the method to convert the component-oriented format in PSCAD to connection-oriented format used in STRUC-L.

3.1.2 The Pin Sequence

The Pin Sequence is the second intermediate data structure used in the compiling process. It is also derived from the dictionary structure, but with a different key and information set.

The key of the Pin Sequence class is the Sampling time, and the information part of the Pin Sequence is the Component Information Sequence, which is another data dictionary data structure. The key of the Component Information Sequence is the Execution Sequence, and the information part is the Component Information, which is further explained in Table 3-2.

| Name | Meaning |
|-------------|--|
| AppName | Application name of the component. |
| fType | Type of corresponding function block |
| fbCom | comments about this function block |
| secCode | checksum code for the corresponding function block |
| pos | function block position (optional) |
| PinList | A list of the connected pins. |
| exSeq | Execution sequence of the component. |
| SampTime | Sampling time for this component |

Table 3-2 The Component Information Sequence

The structure of Pin Sequence is very similar to the that of STRUC-L language. After all the nodes have been processed in the compiler, the Pin

Sequence class will be filled with all the necessary information about the simulated circuit. Thus, the main portion of the connection part of the STRUC-L can be generated by writing out information in this class, and check back with the other 4 intermediate classes.

3.1.3 Library List, Modifier Map and Special Info. List

The other 3 intermediate classes are temporary storage classes for some extra information, they are generated when the Draft file is scanned by the program and are checked later in the compilation process.

The Library list is a linked-list structure and stores the name of the User's Library in the STRUC-L file.

The Modifier Map is derived from the dictionary structure, the key of the modifier map is the co-ordinate of the modifier and the information part is a detailed listing about the modifier, including scaling, limitation, unit and connector side comments.

The Special Info List is a class derived from linked list, it contains the general information of the whole system, including circuit creator, modifier, crea-

tion/modification date, circuit title, subsystem title, number of subsystems, number of component for each subsystems, sampling time and the subsystem comments.

3.2 Functional Model of the Program

In OMT methodology, the functional model describes computations within a software system. The functional model is the third leg of the modelling tripod, in addition to the object model and the dynamic model. The functional model shows how output values in a computation are derived from input values [8]. The functional model in this thesis is divided into two levels. The top level is shown in Fig. 3-2. There are 3 actors (data terminators) in this project, the “PSCAD Draft”, the “Master Program Information” and the “STRUC-L Function Package”. According to OMT notation, these actors are shown as rectangles. The “PSCAD Draft” represents the original Draft file. the “Master Program Information” represents the information contained in the Draft file that is to be used in the STRUC-L Master program. It mainly contains the sampling times for different processors. The actor “STRUC-L Function Package” represents the STRUC-L file

generated as the output of the program. There are 2 data stores shown between two horizontal bars, which are library and the Function Package Header Information. The ellipses represent actions and the arrows represent data flows.

3.2.1 Top Level

In the top level of the functional model shown in Fig. 3-2. The labelled operations in Fig.3-2 are summarized below:

- (1) Extract general Information and store some information into Function Package header Information Data Store;
- (2) Extract some data from the subsystem information part of the draft file and also store them in the Function Package Header Information Data Store;
- (3) Process the component information in the Draft file and look up the library to put data into Function Block Data Store;

(4) Combine the Function Package Header information and the function block information and output the STRUC-L function package.

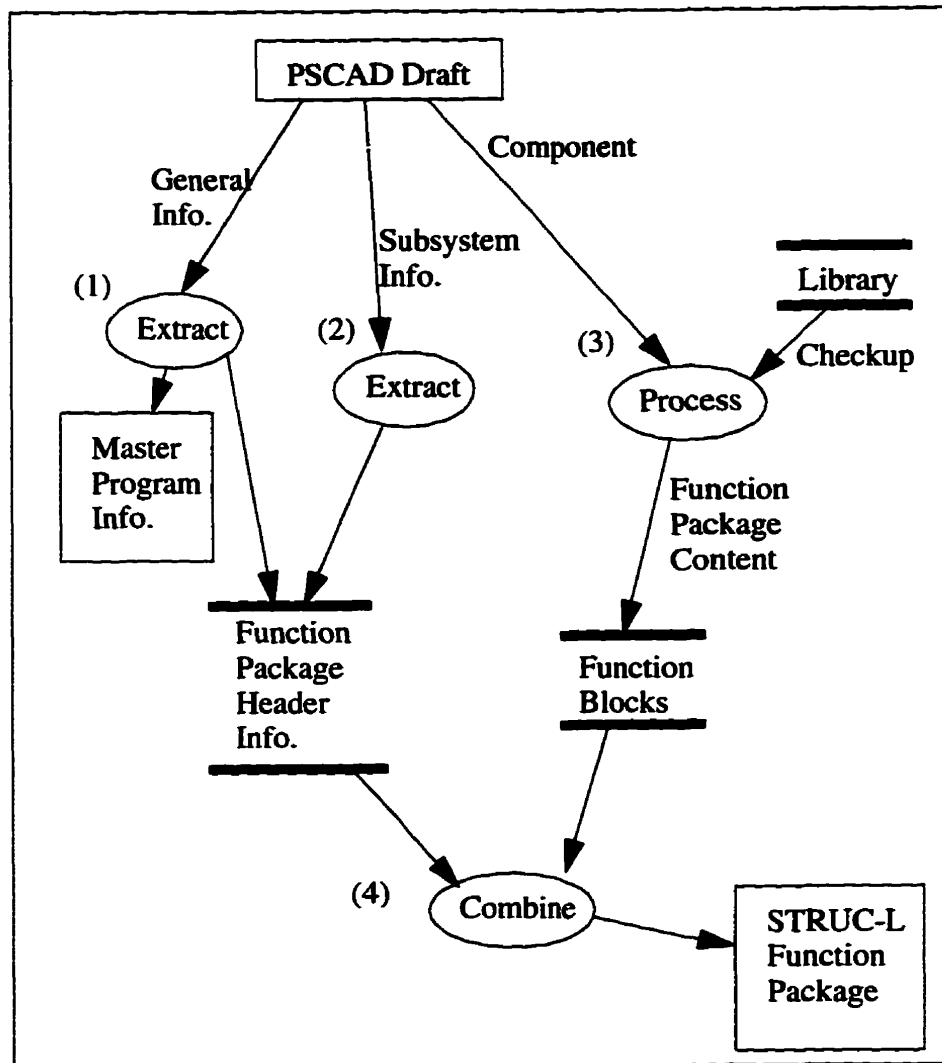


Fig. 3-2 Function Model of the Program

3.2.2 *The Second Level*

The second level of the functional model is shown in Fig. 3-3. This is an expansion of the “process” action in Fig. 3-2. The following is a brief summary of the operations for this level:

- (1) The components are divided into special components and ordinary components. For special components, such as “modifier”, “wire” and “jumper”, a dedicated method is used to transform the connection information and put them into classes “modifier map” and “node map”. For ordinary components, a generic method is defined to look up the library and transform the connection information into the “node map” and the “pin sequence”.

- (2) After all the components in the Draft file have been processed, the “node map” performs a connection checking procedure, in which all the connected nodes are linked together through a linked list. This connection checking procedure will be explained in further detail in the next section.

(3) Finally, the STRUC-L format is generated jointly by the linked node map, pin sequence and the modifier map.

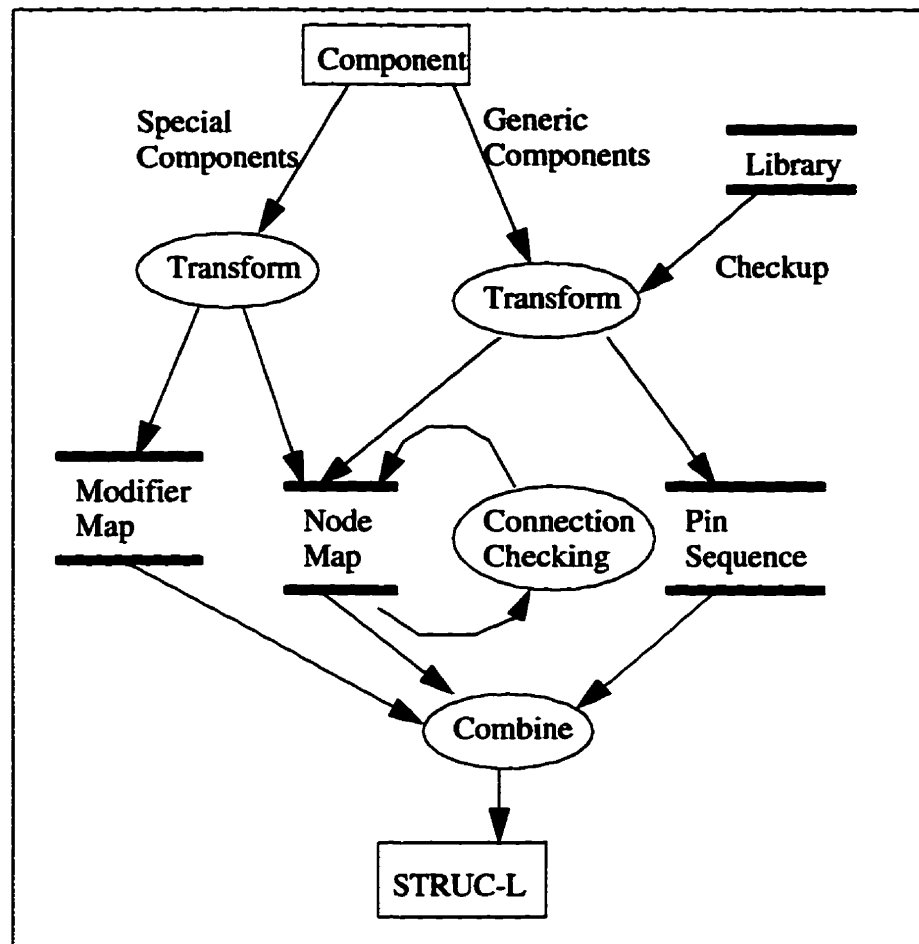


Fig. 3-3 Functional Model, Second Level

3.2.3 The Connection Check-up Procedure for the Node Map

The connection checkup procedure is performed by the node map class. Before the checkup, the information of each node is read directly from the PSCAD Draft file. These nodes are not related to each other. The connection checkup procedure performs a co-ordinate checking for every node, and links any connected nodes together by pointers.

The details of the connection checking procedure are illustrated by a flow chart in Fig. 3-3. The procedure basically contains two loops, the outer loop and the inner loop. The outer loop checks every node in the node map by sequence, while the inner loop checks all other nodes to evaluate connections to the node of the outer loop. If there is a connection, a pointer is put into the node of the outer loop, and all relevant information is collapsed to the node of the inner loop. So at the end, by following the node pointer, all the connected nodes in PSCAD Draft can be grouped together to generate the STRUC-L formatted output.

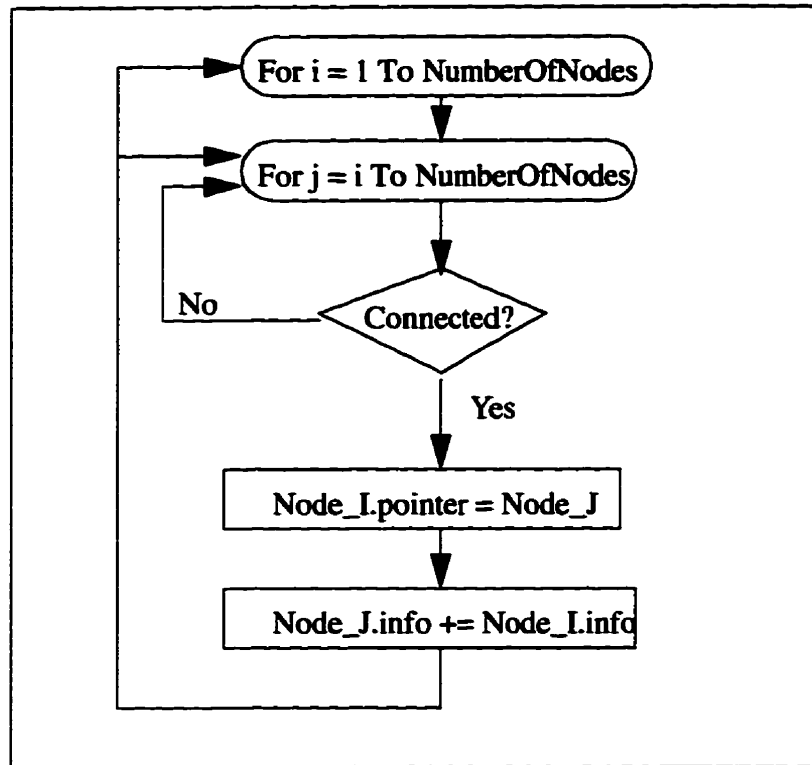


Fig. 3-4 Flow Chart of the Connection Check-up Procedure

Component Library

In the functional model shown in the previous Chapter (Fig. 3-2), it has been mentioned that a special method is defined for each special PSCAD component, to process the information about the component position, orientation and connectors, and to store the results into the intermediate classes. However, for most of the PSCAD components, the information about position, orientation and connectors is presented in a standard format in PSCAD Draft. Thus, a generic method is defined for these PSCAD components, and a component library is utilized to store component specific information.

In this chapter, the generic method and the structure of the component library will be examined. In addition, a special Windows program

has been developed to help the creation, modification and maintenance of this library. The usage of this program will also be demonstrated.

4.1 The Component Library

The basic function of this component library is to map each PSCAD component to a corresponding SIMADYN-D function block, as well as to decide on the parameter values of that function block according to the component state.

The relationship between the PSCAD component and SIMADYN-D function blocks as well as their connectors have been explained in previous chapters. To summarize, (1) one PSCAD component corresponds to one or more SIMADYN-D function blocks, depending on a critical component parameter value, (2) one SIMADYN-D function block connection could be represented in one or more styles in PSCAD Draft module.

The component library is stored in UNIX files using an indexed sequential structure. The benefit of using this structure is (1) to speed up the look-up process; (2) to keep the algorithms relatively simple, since the basic I/O func-

tions of UNIX files has already been well developed and provided by the operating system. There are altogether two tables defined in this library, both of which are stored as fixed record data files, the first table is the Function Block Name Look-up Table. The second table is the Connector Look-up Table. In the first table, there are two pointers that point to the starting and ending address of all connectors for each component. By following these two address, the connectors for the particular function blocks can be easily located in the second table. The definitions of the two tables are illustrated in Table 4-1 and Table 4-2.

| Table Entry Name | Meaning |
|-------------------------|--|
| CompName | PSCAD component name |
| Condi | conditional mapping? |
| CritPara | PSCAD component critical parameter name |
| CritVal | PSCAD component critical parameter value |
| FB | corresponding function block name |
| AddrStart | Starting address in Connector Look-up Table |
| AddrEnd | Ending address in Connector Look-up Table |
| SecCode | Checksum code for this function block |
| Lib | Library to which this function block belongs |

Table 4-1 Function Block Name Look-up Table

| Table Entry Name | Meaning |
|------------------|---|
| FB | Function Block name |
| Seq | Sequence of this connector in function block definition |
| ConName | SIMADYN-D connector name |
| ConType | connector type |
| InOrOut | Input or output |
| Qmark | Question mark on this connector? |
| Condi | Conditional mapping? |
| CritPara | PSCAD component critical parameter name |
| CritVal | PSCAD component critical parameter value |
| Type | Type of connection (COORD, CONSTIN or VOIDOUT) |
| xCoord | X-coordinate (if type = COORD) |
| yCoord | Y-coordinate (if type = COORD) |
| ConstVal | Constant value (if type = constin) |

Table 4-2 Connector Look-up Table

The whole look-up process is summarised below:

- (1) After a PSCAD component is read from the input file, it is first compared with all the special component names. If it is a special component, this component will be processed in the method that has been defined for it specifically, otherwise, this component is an ordinary component, the library look-up method is called, go to step 2;

- (2) Look up the “Function Block Name Look-up Table”, for the particular PSCAD component name, if there is a match, go to the next step. If the component name cannot be found in the table, this component is not defined in the library, an error message will be generated and the program will exit.
- (3) Look up the “Condi” value for this component, if it is FALSE, this PSCAD component will be mapped into one specific STRUC-L function block unconditionally. Thus, the conversion on function block level is finished, we can jump to Step 5 for connector mappings. On the other hand, if the “Condi” value is TRUE, this PSCAD component could be mapped on to one of several different STRUC-L function blocks, depending on a critical parameter and its value. To decide the conditional mapping, go to the next step.
- (4) Read the “CritPara” and “CritVal” from the library, then check the parameter value for “CritPara” of this PSCAD component, if it doesn’t match the previous value in the

library, go to the next record in the library, until a match is found. From the record that contains the matched critical parameters, the corresponding function block name can be decided. If no match is found for this PSCAD component name, this PSCAD component is not configured in the library, an error message will be generated and the program will exit.

- (5) From the record in the “Function Block Name Look-up Table”, read the “AddrStart” and the “AddrEnd” values. These two values mark the starting and ending address of that particular function block in the “Connector Look-up Table”. A FOR loop is used to go through all the records between the starting and ending and a connector checkup process is performed.
- (6) The connector checkup process is very similar to that of the Function Block Name. If “Condi” is FALSE, the connector is unconditionally mapped, otherwise, the mapping exists only when the specified condition is satisfied. The condition

is specified in the “CritPara” and the “CritVal” in the same fashion as that of the “Function Block Name Look-up Table”.

Since both PSCAD and SIMADYN-D are software systems that are under continuous development, additional features may appear in the new version of these software systems, it is likely that the component name, connector type and other details will change with time. Making use of the component library gives a good solution to the evolving software issue. If there is any change in the new version of PSCAD or SIMADYN-D that might affect the compilation from PSCAD to STRUC-L, the compiling program does not need to be changed, all we need to do is to construct a new library. A lot of time and software maintenance effort will be saved by this method, and backward compatibility will be maintained.

In addition, there are currently several SIMADYN-D processor types in use with the practical system. The selection of a processor type mainly depends on both historical compatibility and the complexity of the control project. It is very likely that these different processor types will co-exist for a period of time. The component library also provides a means to generate code for different proc-

essor types from one simulation case, by compiling the PSCAD Draft file using different component libraries, thus gives the maximum flexibility in choosing hardware to realize the control concept.

4.2 The Creation of Component Library

As mentioned before, the component library is implemented as fixed record files on a UNIX system. In order to generate such a library, two text files need to be created first by any kind of UNIX text editors. These two text files could be entered manually, they are just listings of the tables defined in Table 4-1 and Table 4-2. After these two text files are created, a C program “writedata” will read these two files and write them into two new binary files in fixed record format.

Fig. 4-1 is an example of the first text file, which listed the function block name. This text file must have a file extension name of “.nam”. Each value occupies a single line in the file.

```
subjects
1
B
0
ADD2
1
3
&&0011000A 2003B
abc
subjects
1
B
1
SUB
4
6
&&0011000A 2003C
abc
nswan
0
a
a
NSW
7
10
&&0012000B 2404U
abc
```

Fig. 4-1 An example the Function Block Name Table Text File

Fig. 4-2 is an example of the second text file, which is a listing of the connectors. This file must have a file extension name of “.con”. In this file, the details about each connector are listed according to the table defined in Table 4-2. Each value occupies a single line in the file.

```
subjects
1
B
0
ADD2
1
3
&&0011000A 2003B
abc
subjects
1
B
1
SUB
4
6
&&0011000A 2003C
abc
nswsn
0
a
a
NSW
7
10
&&0012000B 2404U
abc
ints
0
a
a
INT
11
26
&&004D001B 2001G
abc
```

Fig. 4-1 An Example of the Connector Text File

4.3 Windows GUI for Library Creation

To further facilitate the creation of the Component Library, a Windows GUI program has been developed. The GUI window is shown in Fig. 4-3.

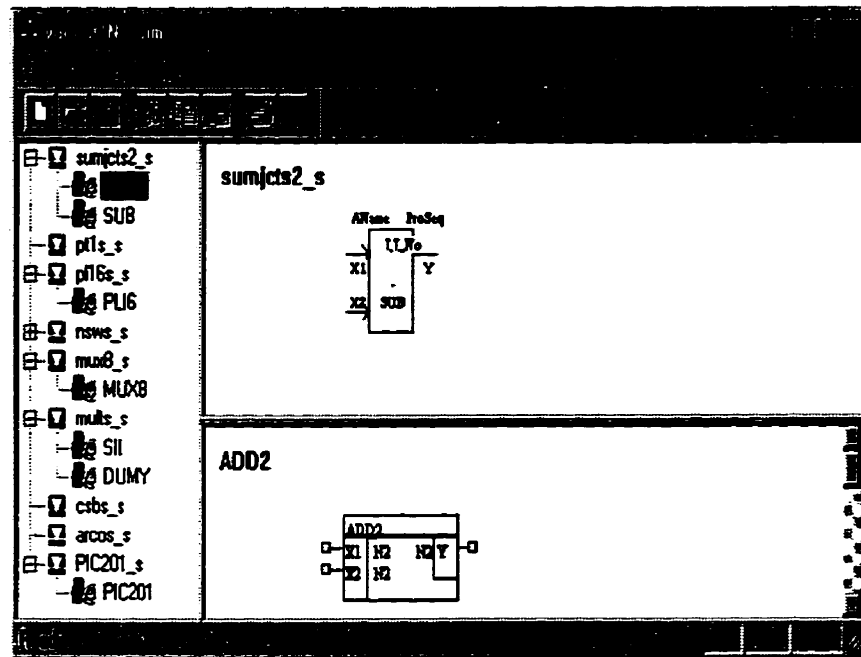


Fig. 4-3 Windows GUI to help Creating Component Library

In the above Windows GUI program, the left-hand side is a tree structure that shows the relationship between each PSCAD component and the corresponding SIMADYN-D function blocks. On the right hand side, the graphical

representation of the selected PSCAD component and SIMADYN-D function block are presented. By double-clicking on the small rectangle at each SYMA-DYN-D function block connector in the lower right window, a dialogue Window will pop up to help the configuration of the connector level mapping. The configuration procedure is very similar to that of the component level mapping. This windows program greatly facilitates the creation and modification of component libraries by presenting a graphical interface. The text files that we mentioned before will be generated as the result of this program.

Validation of the PSCAD to STRUC-L Compilation Software

The previous chapters described a compilation program designed to convert a PSCAD file directly to the real world control system SIMADYN-D. In this chapter, several test cases will be presented to validate the practical value and the usability of this program.

The basic idea of verification is to load the compilation result into the SIMADYN-D system. However, due to the complexity and expenses involved in making real control boards, it is not necessary to conduct all verification cases completely to the hardware level. The verification can be done in three different levels, as summarized below:

(1) **Level 1: Format Verification.** Load the result into the STRUC-L editor. If the result can be successfully loaded into the STRUC-L editor, the syntax of the generated file is verified, since a mis-formatted file can not be loaded into the STRUC-L editor. As a further test, the generated STRUC-L file can also be loaded into STRUC-G environment, so that a graphic representation of the circuit can be obtained. Since the compiler program in STRUC-G has an auto-placement function, the loaded circuit will be laid out automatically. This graphic layout should be modified later to improve readability.

(2) **Level 2: Sanity Check.** Compile the STRUC-L file using the SIMADYN-D internal compiler. The SIMADYN-D internal compiler checks the connection and the validates all the function blocks, then generates compiled function packages (".ofp" files) from the corresponding un-compiled files (".ufp" files). If the STRUC-L file can be successfully compiled by the SIMADYN-D internal compiler, the circuit specified in the STRUC-L file can be downloaded to the

SIMADYN-D control board. For some small scale circuits, this level of test is enough, since the behaviour of the digital circuit can be easily figured out from the circuit schematic drawings.

(3) **Level 3: Hardware Testing.** Hardware testing. The generated “.ofp” files are loaded into SIMADYN-D control board and the real-world performance of the circuit is measured. This is the final stage of testing and successful results can prove the correctness of the program developed in this thesis. The hardware testing is much more time-consuming and costly than the level 2 testing.

In this chapter, two tests will be presented, one of which is tested at the hardware level (level 3), the other is tested at level 2.

5.1 Test No. 1, Wave Form Generator

The first test is a wave form generator. The schematic drawing in PSCAD software of the circuit is shown in Fig. 5-1a, Fig. 5-1b and Fig. 5-1c. The corresponding Draft file is attached as appendix 1. The subsystem “para” sets several parameters for the system. The subsystem “sinus” (Fig. 5-1b) is a sine wave generator, it uses two integrators and one reverser to generate a sine wave. The subsystem “SAW” is a saw-tooth wave generator. It utilizes the integrator to generate the ramp signal and resets the integrator when the limit is reached, so that a repeating saw-tooth wave is generated at the output.

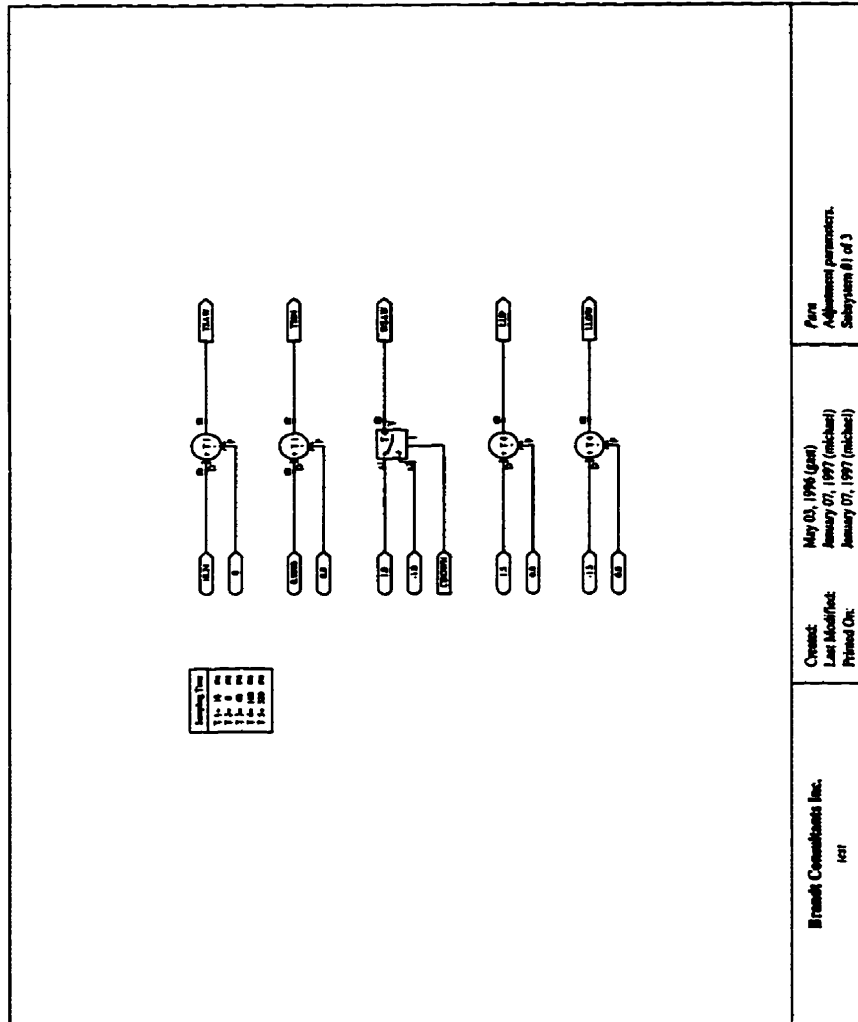


Fig. 5-1a Subsystem "para" of the Wave Form Generator

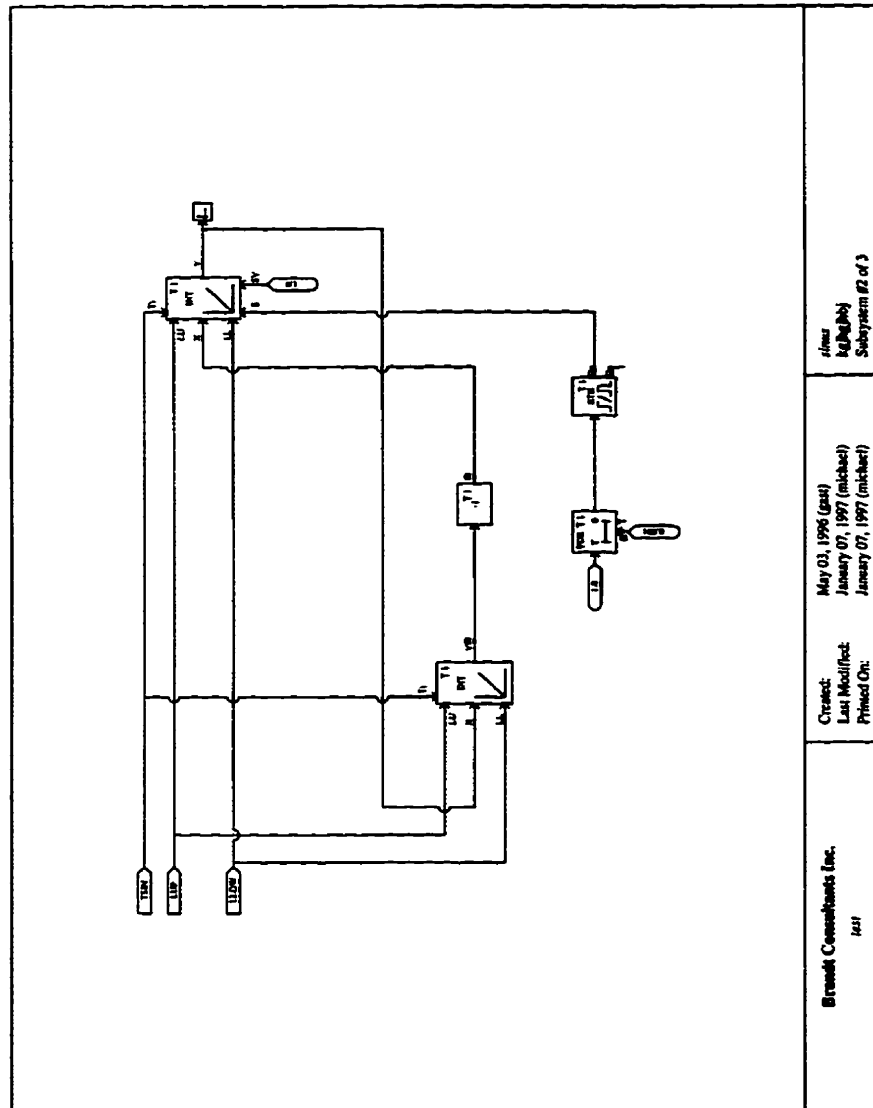


Fig. 5-1b Subsystem "sinus" of the Wave Form Generator.

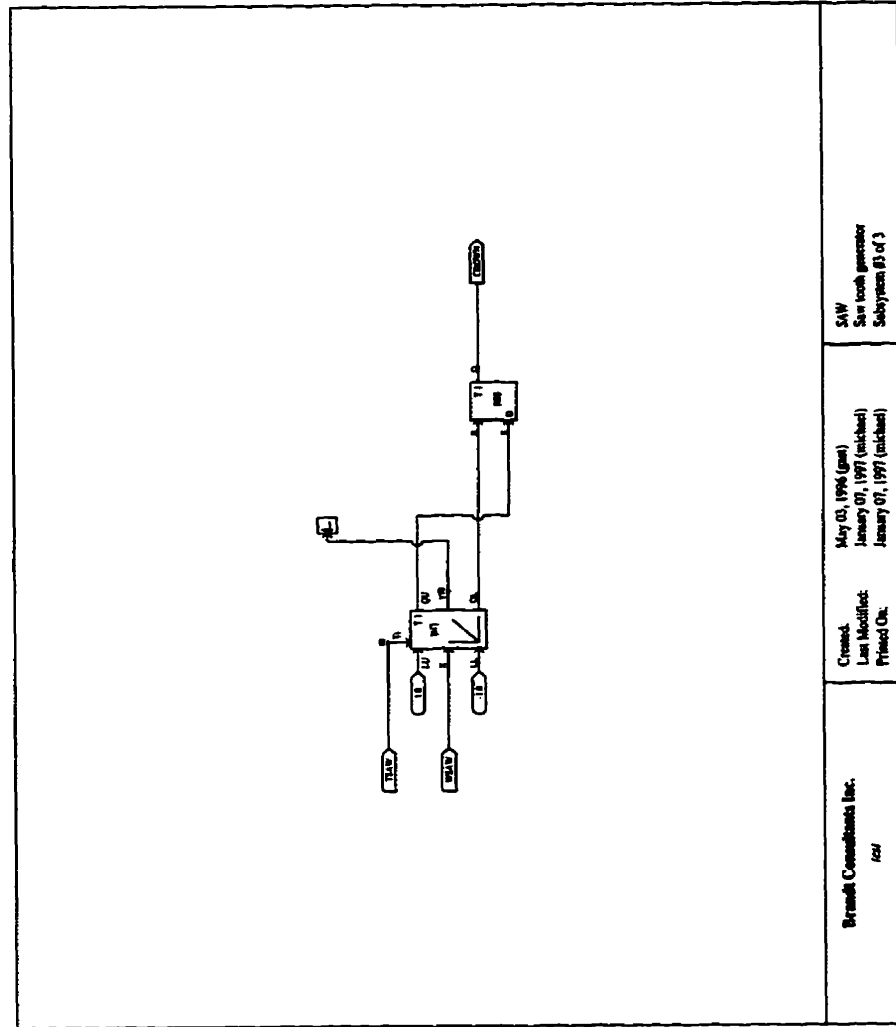


Fig. 5-1c Subsystem "SAW" of the Wave Form Generator

After all the components are laid out on PSCAD Draft module as shown in Fig. 5-1, the corresponding Draft file is processed using the program demonstrated in thesis, the result are three separate STRUC-L files, each of which is a stand alone function package. These STRUC-L files are attached in Appendix 2. The simulation software can simulate the behaviour of such a circuit and the result is plotted in Fig. 5-2. The generated real world control should have the same result as this one.

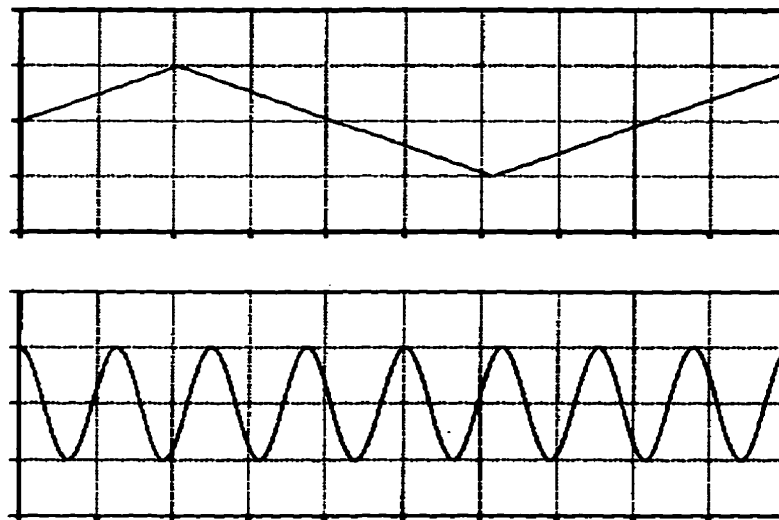


Fig. 5-2 The simulation result.

The next step for verification is to load the generated code into STRUC-L and STRUC-G environment. The graphical representation of the circuit in STRUC-G is obtained, which is shown in Fig.5-3a, Fig. 5-3b and Fig. 5-3c.

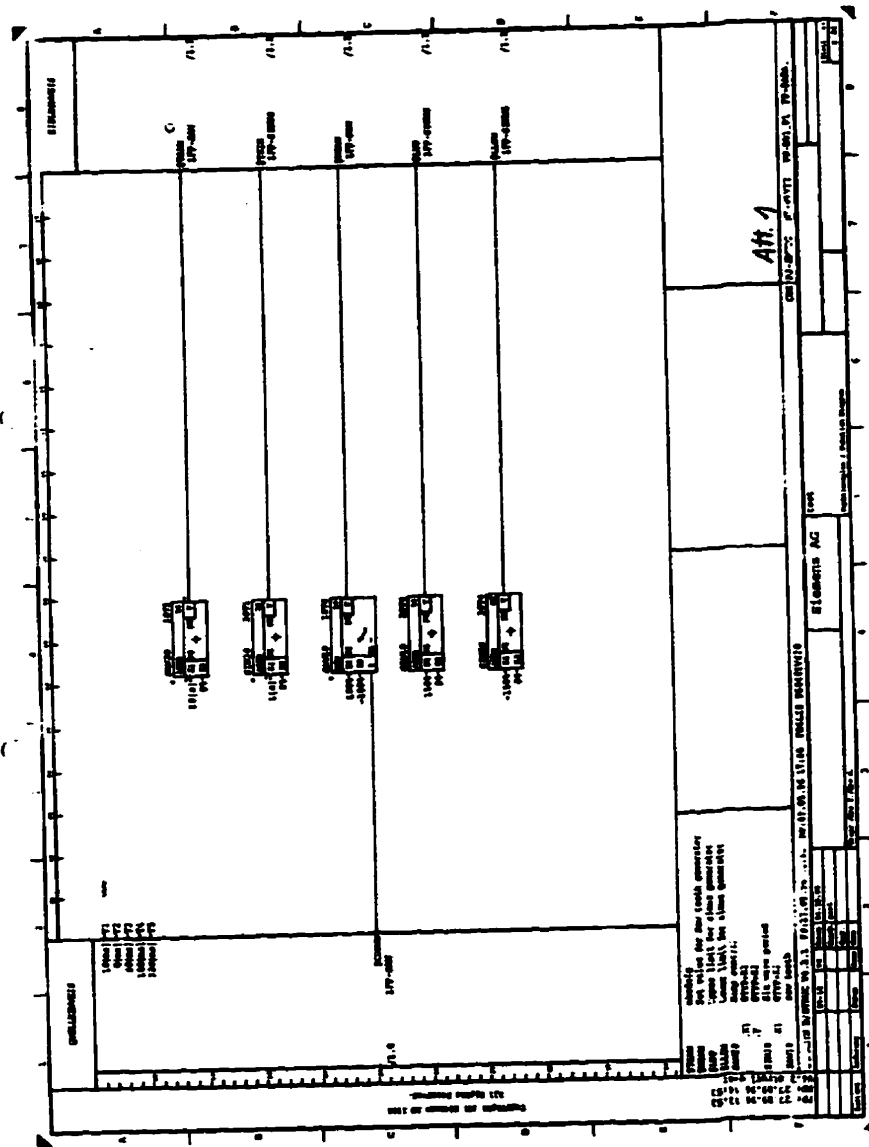


Fig. 5-3a Function Package FP-PARA in STRUC-G

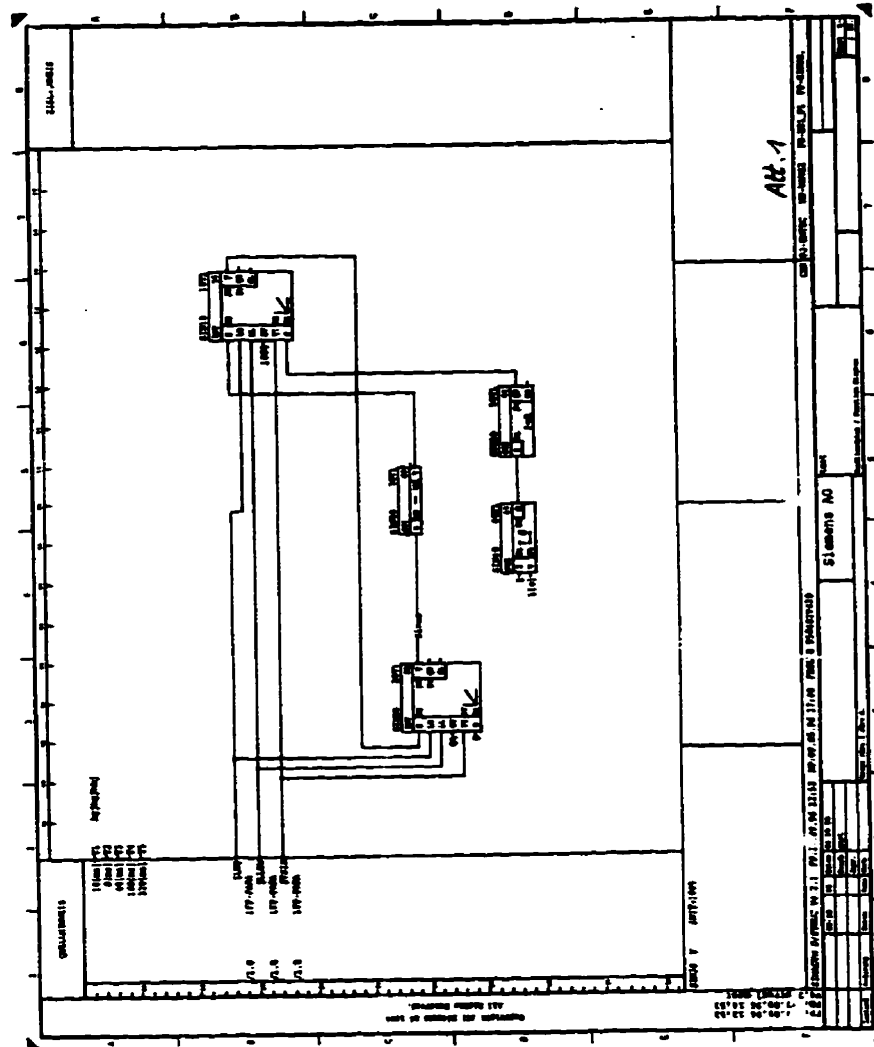


Fig. 5-3b Function Package FP-SINUS in STRUC-G

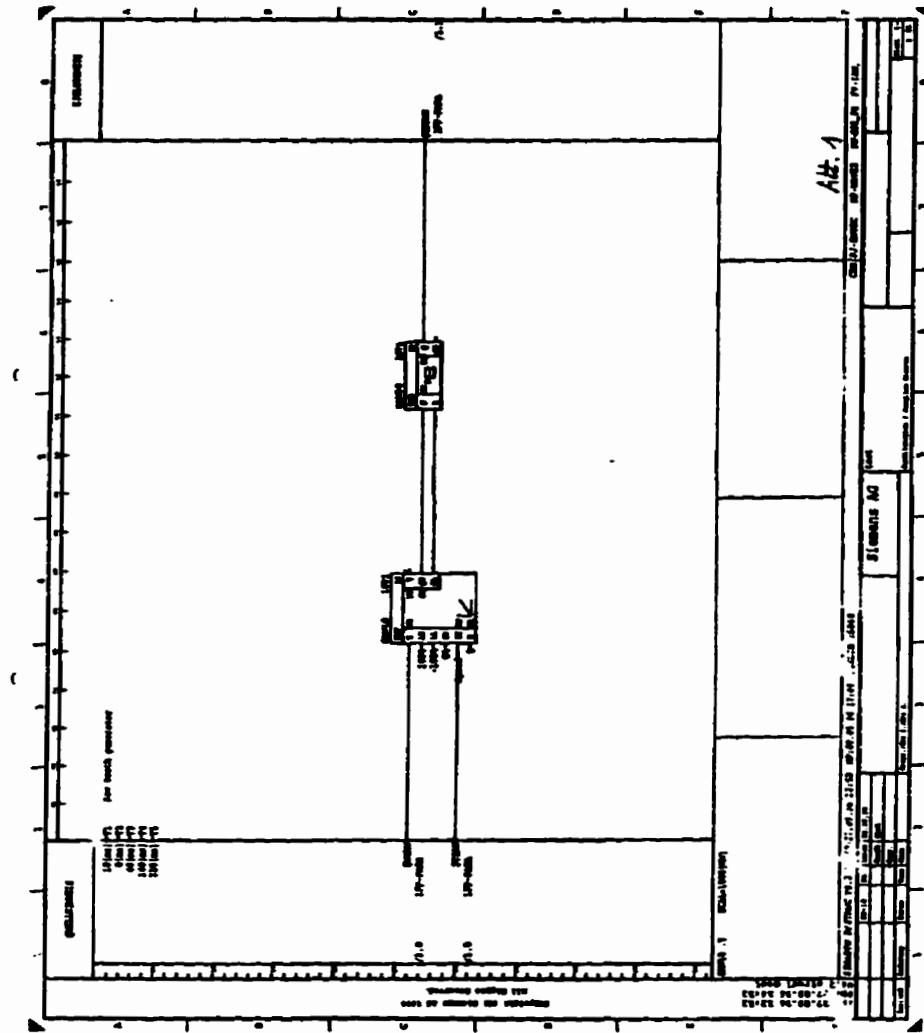


Fig. 5-3c Function Package FP-SAW in STRUC-G

Next, use the SIMADYN-D internal compiler to compile the generated STRUC-L file. This step has been gone through without any error. Then, the compiled files were sent to SIEMENS AG for hardware testing. The real world hardware output was obtained, which is shown in Fig. 5-4

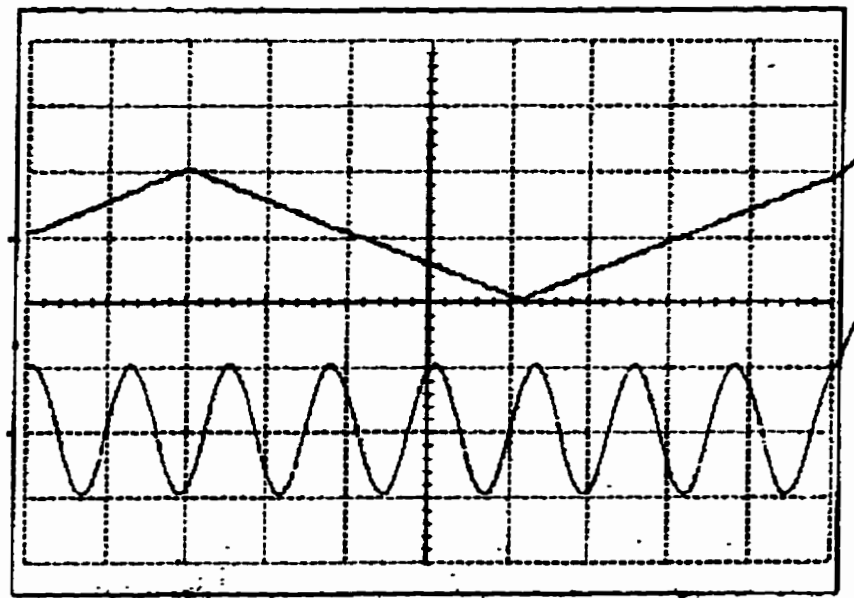


Fig. 5-4 The real control output plots

Comparing Fig. 5-4 with Fig. 5-2, the real world result matches the simulation results very well. These results demonstrate that the idea of integrating control generation with simulation is feasible and the computer program developed in this thesis is correct and working with simple cases.

5.2 Test No. 2, Part of an HVDC Inverter Side Controller

The second test case is taken from part of an HVDC system inverter side controller. This case embodies several of the features required in commercial applications of SIMADYN-D. The circuit is shown in Fig. 5-4. The basic function of the circuit is to use a selector (CSB.C) to select an input from several sources according to the system state parameter, and then feed the selector output to a PI controller. The corresponding Draft file is attached as Appendix 3.

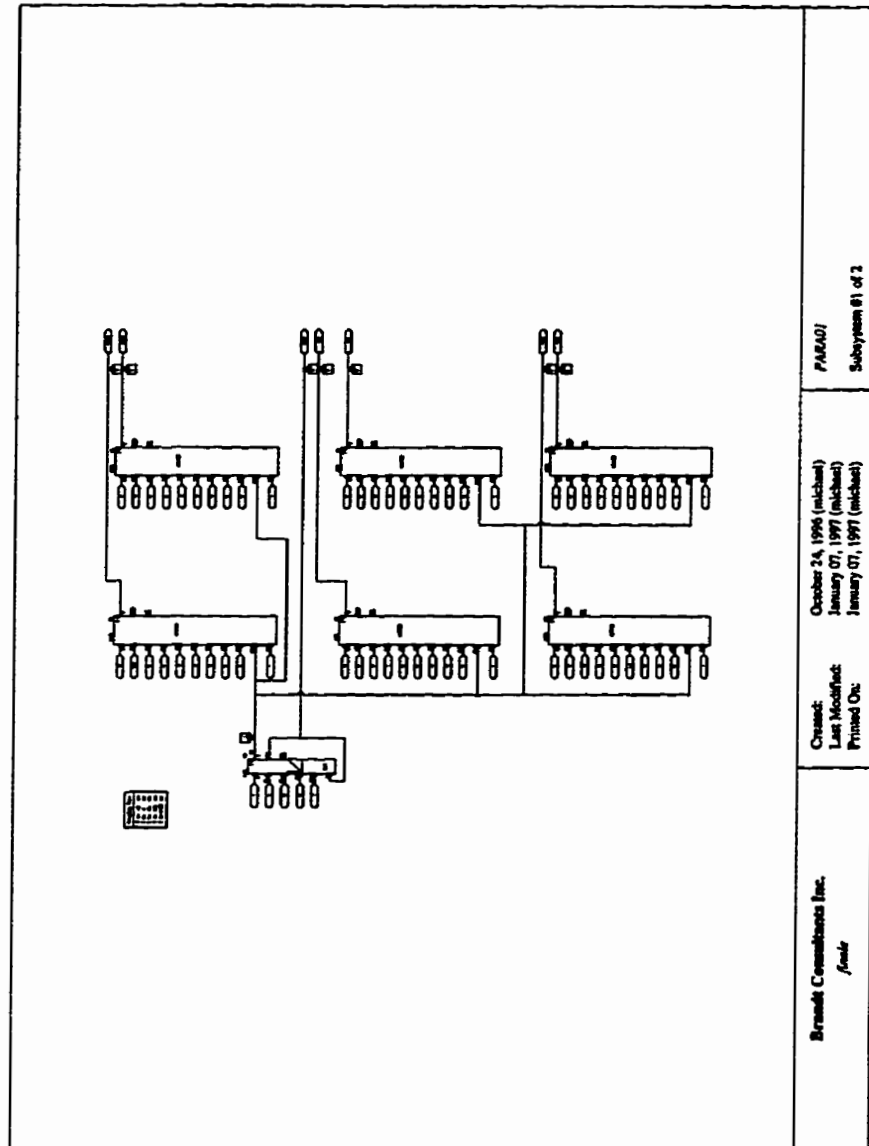


Fig. 5-4 Subsystem PARA01 of the Second Test

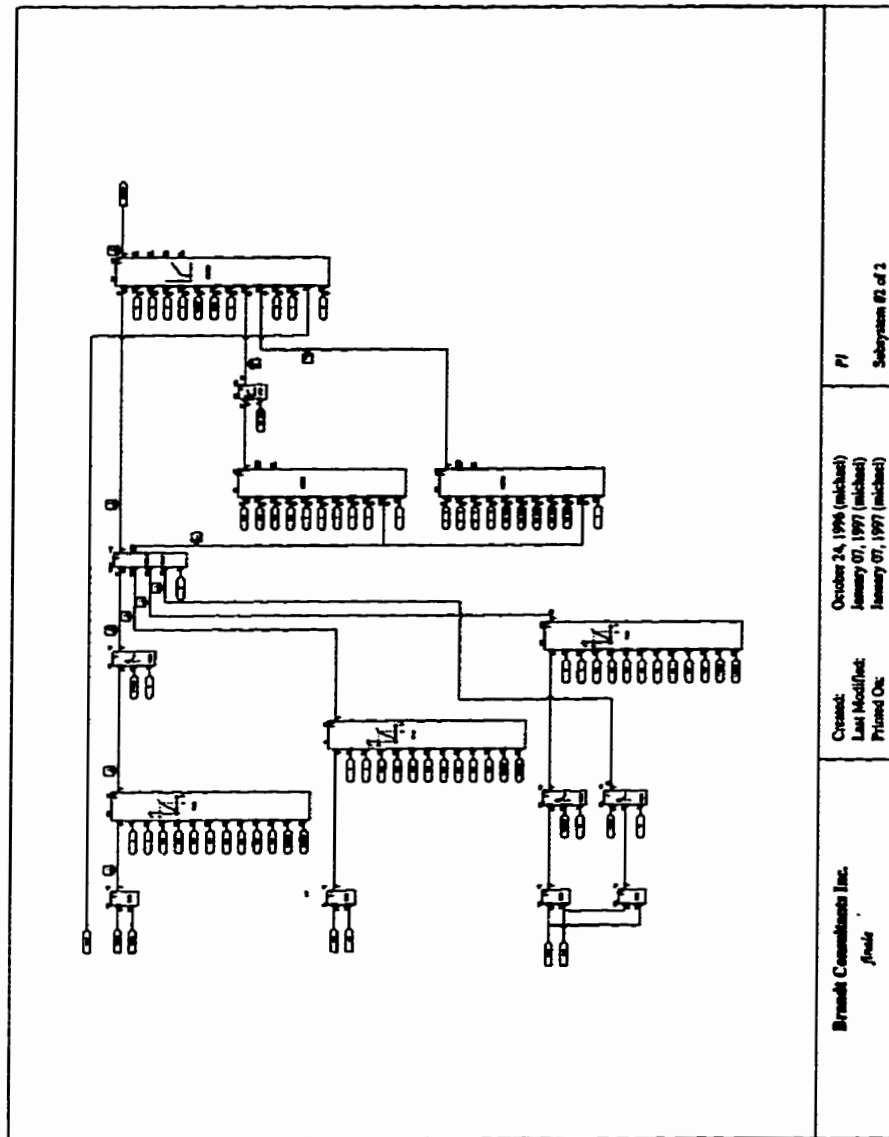


Fig. 5-4b Subsystem "PI" of the Second Test

For this second test, the verification work has been completed to the test level 2 mentioned before (Sanity Check). The STRUC-L code was generated and checked by the SIMADYN-D internal compiler, the compilation has been successful. Since the hardware level test has to be carried out in Siemens AG and involves many extra resources, no results have been obtained so far.

Conclusions

It is possible to develop an integrated environment for simulation and control development. In this environment, the proposed control system can be thoroughly analysed through digital simulation, and the crystallized design readily converted to the software code for the real-world controller.

The features available for control modelling in PSCAD/EMTDC can be exploited to develop this technique. An off-line assignment program can make the task of assigning SIMADYN D code to the PSCAD icon fairly straightforward.

The use of the integrated development/simulation environment will speed up the design and implementation of complex controls and reduce the engineering costs.

The following is a summary of the computer program that converts the simulated control scheme directly into control code:

- The simulation case is divided into several subsystems, those which do not need to be included in the real world controls system can be marked and filtered out by the program.
- The program performs a component to component mapping from the simulation scheme to the real world control system. The special components such as wire and jumper are handled specially by separate pieces of code. The ordinary components are handled by checking out a component library.
- The component library specifies the relationship between each simulation component and real world control component, as well as the relationship between simulation parameters and the real world control parameters.

- **A few test cases have been conducted and satisfactory results has been obtained. These test cases validate the idea of integrating control generation with simulation software and also prove that the computer program developed in this thesis works with simple control cases.**

6.1 Recommendations for Further Work

Firstly, any non-trivial real control system will contains a large number of function blocks. In order to utilize the concept developed in this thesis, a very comprehensive library needs to be built and tested.

Another aspect of an integrated simulation and control implementation environment is the engineering cycle for designing and implementing the control system. It is quite common in the practical cases that some change needs to be done for the code on a control board. In such cases, it is highly desirable that the changes could be simulated first and the changes can be exactly copied into the real control board. Thus, to have a link from real control system back to the simulation software would be very beneficial for such a purpose. The concept of this link is shown in Fig. 6-1.

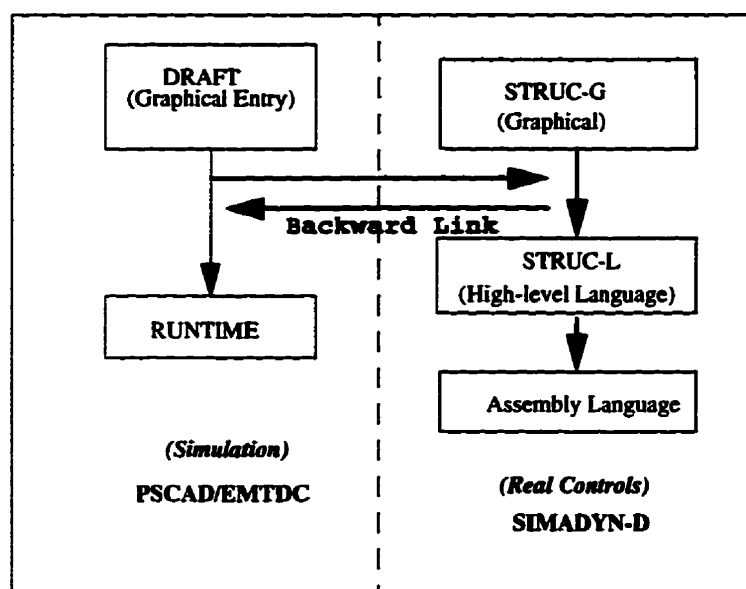


Fig. 6-1 Link from Real-world control back to Simulation Software

XDRAFT Version 4.2.2
EMTDC
TITLE: test
CREATED: May 03, 1996 (gast)
LAST-MODIFIED: October 21, 1996 (michael)
TIME-STEP: 0.001
FINISH-TIME: 50
PRINT-STEP: 0.01
RTDS-RACK: 0
RTDS REAL-TIME: Yes
3 SUBSYSTEMS
SUBSYSTEM-TITLE: Para
SUBSYSTEM-COMMENT: Adjustment parameters.
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
46 COMPONENTS
export
656 208 2 0 1
Name:TSAW
sumjcts
496 208 0 0 8
T_Na:T
T_No:1
DPath:1
B:0
AName:SAW20
ProSeq:10
FbCom:Ramp control
StgPos:
WIRE
400 208 0 0 0
-32 -4 64 4
WIRE
432 240 0 0 0
-64 -4 64 4
WIRE
560 208 0 0 0
-32 -4 64 4
WIRE
400 624 0 0 0
-32 -4 64 4
WIRE
400 656 0 0 0
-32 -4 64 4
import

336 688 0 0 1
Name:CDOWN
WIRE
432 688 0 0 0
-64 -4 64 4
export
656 624 2 0 1
Name:WSAW
WIRE
560 624 0 0 0
-32 -4 64 4
export
656 1008 2 0 1
Name:LLOW
sumjcts
496 1008 0 0 8
T_Na:T
T_No:4
DPath:l
B:0
AName:SIN20
ProSeq:30
FbCom:
StgPos:
WIRE
432 1040 0 0 0
-64 -4 64 4
WIRE
400 1008 0 0 0
-32 -4 64 4
WIRE
560 1008 0 0 0
-32 -4 64 4
sampletables
176 112 0 0 6
T:T
T1:10
T2:0
T3:40
T4:160
T5:320
export
656 400 2 0 1
Name:TSIN

sumjct
496 400 0 0 8
T_Na:T
T_No:1
DPath:1
B:0
AName:SIN30
ProSeq:20
FbCom:Sin wave period
StgPos:02.03.07
WIRE
432 432 0 0 0
-64 -4 64 4
WIRE
560 400 0 0 0
-32 -4 64 4
export
656 848 2 0 1
Name:LUP
sumjct
496 848 0 0 8
T_Na:T
T_No:4
DPath:1
B:0
AName:SIN10
ProSeq:20
FbCom:
StgPos:
WIRE
432 880 0 0 0
-64 -4 64 4
WIRE
400 848 0 0 0
-32 -4 64 4
WIRE
560 848 0 0 0
-32 -4 64 4
const
336 240 0 0 1
Value:0
const
336 208 0 0 1
Value:10.24

const
336 432 0 0 1
Value:0.0
const
336 624 0 0 1
Value:1.0
const
336 656 0 0 1
Value:-1.0
const
336 848 0 0 1
Value:1.5
const
336 880 0 0 1
Value:0.0
const
336 1040 0 0 1
Value:0.0
const
336 1008 0 0 1
Value:-1.5
nswsn
496 624 0 0 6
T_Na:T
T_No:4
AName:SAW10
ProSeq:10
FbCom:saw tooth
StgPos:09.08.97
WIRE
496 656 1 0 0
-4 0 4 32
modifier
464 208 0 0 18
Name:
Com:
EnSw:1
Tp1:N2
Tp2:R2
LnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4

Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
const
336 400 0 0 1
Value:0.9990
WIRE
400 400 0 0 0
-32 -4 64 4
modifier
464 400 0 0 18
Name:
Com:
EnSw:1
Tp1:N2
Tp2:R2
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
528 208 0 0 18
Name:
Com:abcdefg
EnSw:1
Tp1:N2
Tp2:R2
EnScal:0
Scal:1
EnUnit:1

Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
528 400 0 0 18
Name:
Com:
EnSw:0
Tp1:N2
Tp2:R2
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
528 624 0 0 18
Name:
Com:Set value for Saw tooth generator
EnSw:0
Tp1:N2
Tp2:R2
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4

EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
528 848 0 0 18
Name:
Com:Upper limit for sinus generator
EnSw:0
Tp1:N2
Tp2:R2
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
528 1008 0 0 18
Name:
Com:Lower limit for sinus generator
EnSw:0
Tp1:r
Tp2:r
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:

EnMin:0
Min:
SUBSYSTEM-TITLE: sinus
SUBSYSTEM-COMMENT: kgjhgjhbj
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
45 COMPONENTS
import
208 144 0 0 1
Name:LUP
ints
432 464 0 0 17
ELU:1
ELL:1
QU:0
QL:0
S:0
ESV:0
ETi:1
T_Na:T
T_No:1
LU:1.0
LL:-1.0
SV:0.0
Ti:0.01
AName:SIN20
ProSeq:20
FbCom:
StgPos:
WIRE
528 144 0 0 0
-288 -4 320 4
ints
880 176 0 0 17
ELU:1
ELL:1
QU:0
QL:0
S:1
ESV:1
ETi:1
T_Na:T
T_No:1
LU:1.0
LL:-1.0

SV:0.0
Ti:0.01
AName:SIN10
ProSeq:10
FbCom:
StgPos:
WIRE
272 272 1 0 0
-4 -128 4 160
WIRE
336 432 0 0 0
-64 -4 64 4
import
208 208 0 0 1
Name:LLOW
jumper
272 208 2 0 0
WIRE
560 208 0 0 0
-256 -4 288 4
WIRE
240 336 1 0 0
-4 -128 4 160
WIRE
304 496 0 0 0
-64 -4 96 4
WIRE
624 368 0 0 0
-320 -4 352 4
jumper
304 432 3 0 0
WIRE
304 368 1 0 0
-4 0 4 32
WIRE
336 464 0 0 0
-32 -4 64 4
WIRE
976 272 1 0 0
-4 -96 4 96
WIRE
944 176 0 0 0
0 -4 32 4
WIRE

560 464 0 0 0
-64 -4 64 4
WIRE
752 464 0 0 0
-64 -4 64 4
jumper
816 368 3 0 0
WIRE
816 432 1 0 0
-4 -32 4 32
jumper
816 208 3 0 0
WIRE
816 272 1 0 0
-4 -32 4 64
WIRE
816 176 0 0 0
0 -4 32 4
WIRE
560 112 0 0 0
-320 -4 320 4
import
208 112 0 0 1
Name:TSIN
jumper
432 368 3 0 0
jumper
432 208 3 0 0
jumper
432 144 3 0 0
WIRE
432 272 1 0 0
-4 -32 4 64
WIRE
848 688 0 0 0
-32 -4 32 4
jumper
880 368 3 0 0
WIRE
880 272 1 0 0
-4 -32 4 64
pdefs
624 688 0 0 7
T_Na:T

T_No:1
Type:0
AName:SIN40
ProSeq:40
FbCom:
StgPos:
etes
784 688 0 0 6
T_Na:T
T_No:1
AName:SIN50
ProSeq:50
FbCom:
StgPos:
WIRE
688 688 0 0 0
-32 -4 64 4
pgb
976 176 0 0 6
DL:SINUS
Group:
Scale:1.0
Max:2.0
Min:-2.0
Units:
gains
656 464 0 0 8
T_Na:T
T_No:1
G:-1
COM:Gain
AName:SIN30
ProSeq:30
FbCom:
StgPos:
const
560 688 0 0 1
Value:1.0
const
624 752 3 0 1
Value:0.001
const
912 272 3 0 1
Value:1.0

modifier
496 464 0 0 18
Name:Sinus
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
688 464 0 0 18
Name:
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:1
Init:100%
EnMax:0
Max:
EnMin:0
Min:
modifier
624 720 3 0 18
Name:
Com:

EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
WIRE
880 528 1 0 0
-4 -128 4 160
SUBSYSTEM-TITLE: SAW
SUBSYSTEM-COMMENT: Saw tooth generator
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
21 COMPONENTS
WIRE
240 272 0 0 0
-64 -4 64 4
import
144 336 0 0 1
Name:WSAW
import
144 272 0 0 1
Name:TSAW
WIRE
208 336 0 0 0
-32 -4 64 4
ints
304 336 0 0 17
ELU:1
ELL:1
QU:1
QL:1
S:0
ESV:0
ETi:1

APPENDIX 1 : DRAFT Files for Test No. 1

T_Na:T
T_No:1
LU:1.0
LL:-1.0
SV:0.0
Ti:0.01
AName:SAW10
ProSeq:10
FbCom:
StgPos:
rsffs
592 400 0 0 7
T_Na:T
T_No:1
Type:0
AName:SAW20
ProSeq:20
FbCom:
StgPos:
WIRE
400 304 0 0 0
-32 -4 64 4
WIRE
464 368 0 0 0
-96 -4 96 4
WIRE
496 400 0 0 0
-32 -4 64 4
jumper
464 368 3 0 0
WIRE
464 304 1 0 0
-4 0 4 32
export
784 368 2 0 1
Name:CDOWN
WIRE
688 368 0 0 0
-32 -4 64 4
pgb
432 208 0 0 6
DL:SAW
Group:
Scale:1.0

Max:2.0
Min:-2.0
Units:
WIRE
400 336 0 0 0
-32 -4 32 4
jumper
432 304 3 0 0
WIRE
432 240 1 0 0
-4 -32 4 32
const
240 304 0 0 1
Value:1.0
const
240 368 0 0 1
Value:-1.0
modifier
368 336 0 0 18
Name:
Com:
EnSw:0
Tp1:
Tp2:
EnScal:1
Scal:1000
EnUnit:1
Unit0:4
Unit1:4
Unit2:4
Unit3:3
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
304 272 0 0 18
Name:Speed
Com:
EnSw:0
Tp1:
Tp2:

**EnScal:0
Scal:1000
EnUnit:1
Unit0:4
Unit1:4
Unit2:4
Unit3:3
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
SEQUENCER DATA [916,380]
Subsystem #1: Repeat
Subsystem #2: Repeat
Subsystem #3: Repeat**

```

1 FP-PARA,P16
&V4.2.1 FP-DIA UNC 08.07.96 08:54
& FBSLIB 950401V420 A:
&B: C:
&&0005563
2CT1 4S = 'test "Inhalt/Content"
3CT2 4S = '
4CT3 4S = '
5HLD 3S = ' "Anlagenkennz./Higher lev.design"
6DLS 1S = ' "Zeichn.Nr./Draw. No List suffix"
7DGS 1S = ' " Graphic suffix"
8DPS 1S = ' " Besteller/Purchaser suffix"
9DES 2S = 'gast "Bearbeiter/Designer "
10ORD 4S = ' "Urspr/Original document"
11MD3 4S = ' / /10-21/96/ "Modification"
12MD2 4S = ' / / / '
13MD1 4S = ' / / / '
14+
15 TX=T1
16 SAW20 : ADD2 ,POS=01.01.01 "Ramp control"
&&0011000A 2003B
17X1 N2 < @TYP=R2,10[s]
18X2 N2 < 0%
19Y N2 > @TYP=R2,$TSAW "abcdefg"
20+
21
22 SIN30 : ADD2 ,POS=02.03.07 "Sin wave period"
&&0011000A 2003B
23X1 N2 < @TYP=R2,0[s]
24X2 N2 < 0%
25Y N2 > $TSIN
26+
27
28 TX=T4
29 SAW10 : NSW ,POS=09.08.97 "saw tooth"
&&0012000E 2404U
30X1 N2 < 100%
31X2 N2 < -100%
32I B1 < $CDOWN
33Y N2 > $WSAW "Set value for Saw tooth generator"
34+
35
36 SIN10 : ADD2 ,POS=01.01.01

```

APPENDIX 2 : STRUC-L files for Test No. 1

```

&&0011000A 2003B
37X1 N2 < 150%
38X2 N2 < 0%
39Y N2 > $LUP          "Upper limit for sinus generator"
40+
41
42 SIN20 : ADD2 ,POS=01.01.01
&&0011000A 2003B
43X1 N2 < -150%
44X2 N2 < 0%
45Y N2 > $LLOW        "Lower limit for sinus generator"
46+
47
48 CM,POS=01
49 "Adjustment parameters."
50 END
51
????????????????????????????????????????????????????????????????????????????????????

```

```

1 FP-SINUS.P16
&V4.2.1 FP-DIA UNC 08.07.96 08:54
& FBSLIB 950401V420 A:
&B:          C:
&&0005563
2CT1 4S = 'test          ""Inhalt/Content""
3CT2 4S = '              '
4CT3 4S = '              '
5HLD 3S = '              ""Anlagenkennz./Higher lev.design""
6DLS 1S = ' '            ""Zeichn.Nr./Draw. No List suffix""
7DGS 1S = ' '            " . . Graphic suffix"
8DPS 1S = ' '            "  Besteller/Purchaser suffix"
9DES 2S = 'gast         '   ""Bearbeiter/Designer  ""
10ORD 4S = '            ""Urspr/Original document""
11MD3 4S = ' /          /10-21/96/ ""Modification""
12MD2 4S = ' /          / / '
13MD1 4S = ' /          / / '
14+
15 TX=T1
16 SIN10 : INT ,POS=01.01.01

```

```

&&004D0018 2001G
17X N2 < SIN30.Y
18LU N2 < $LUP
19LL N2 < $LLOW
20SV N2 < 100%
21TI R2 < $TSIN
22S B1 < SIN50.QP
23Y N2 >
24QU B1 >
25QL B1 >
26+
27
28 SIN20 : INT ,POS=01.01.01
&&004D0018 2001G
29X N2 < SIN10.Y
30LU N2 < $LUP
31LL N2 < $LLOW
32SV N2 < 0%
33TI R2 < $TSIN
34S B1 < 0
35Y N2 > ,Sinus'
36QU B1 >
37QL B1 >
38+
39
40 SIN30 : SII ,POS=01.01.01
&&000D0008 2003A
41X N2 < SIN20.Y
42Y N2 > ,INIT=100%
43+
44
45 SIN40 : PDE ,POS=01.01.01
&&00220010 2404N
46I B1 < 1
47T T2 < 0[s]
48Q B1 >
49+
50
51 SIN50 : ETE ,POS=01.01.01
&&0018000E 2404I
52I B1 < SIN40.Q
53QP B1 >
54QN B1 >
55+

```

56
 57 CM,POS=01
 58 "kgjhghbj"
 59 END
 60
 ???

1 FP-SAW,P16
 &V4.2.1 FP-DIA UNC 08.07.96 08:54
 & FBSLIB 950401V420 A:
 &B: C:
 &&0005563
 2CT1 4S = 'test' "Inhalt/Content"
 3CT2 4S = '
 4CT3 4S = '
 5HLD 3S = ' "Anlagenkennz./Higher lev.design"
 6DLS 1S = ' "Zeichn.Nr./Draw. No List suffix"
 7DGS 1S = ' " Graphic suffix"
 8DPS 1S = ' " Besteller/Purchaser suffix"
 9DES 2S = 'gast "Bearbeiter/Designer "
 10ORD 4S = ' "Urspr/Original document"
 11MD3 4S = ' / /10-21/96/ "Modification"
 12MD2 4S = ' / / / '
 13MD1 4S = ' / / / '
 14+
 15 TX=T1
 16 SAW10 : INT ,POS=01.01.01
 &&004D0018 2001G
 17X N2 < \$WSAW
 18LU N2 < 100%
 19LL N2 < -100%
 20SV N2 < 0%
 21TI R2 < \$TSAW,'Speed'
 22S B1 < 0
 23Y N2 > ,SCAL=1000[MW]
 24QU B1 >
 25QL B1 >
 26+
 27

28 SAW20 : RSS ,POS=01.01.01
&&001600C 2004K
29S B1 < SAW10.QU
30R B1 < SAW10.QL
31Q B1 > \$CDOWN
32QN B1 >
33+
34
35 CM,POS=01
36 "Saw tooth generator"
37 END
38
??

APPENDIX 3 : DRAFT files for Test No. 2

XDRAFT Version 4.2.2
EMTDC
TITLE: finale
CREATED: October 24, 1996 (michael)
LAST-MODIFIED: October 24, 1996 (michael)
TIME-STEP: 0.001
FINISH-TIME: 20
PRINT-STEP: 0.001
RTDS-RACK: 0
RTDS REAL-TIME: Yes
2 SUBSYSTEMS
SUBSYSTEM-TITLE:
SUBSYSTEM-COMMENT:
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
125 COMPONENTS
sampletables
144 144 0 0 6
T:T
T1:10
T2:0
T3:40
T4:160
T5:5120
WIRE
368 368 0 0 0
-96 -4 128 4
ints_s
240 368 0 0 15
ELU:1
ELL:1
QU:1
QL:1
S:1
ESV:1
ETi:1
AName:A10
ProSeq:10
T_Na:T
T_No:4
LU:9
LL:0
SV:0.0
Ti:0.16

APPENDIX 3 : DRAFT files for Test No. 2

const
176 368 0 0 1
Value:1
const
176 400 0 0 1
Value:9
const
176 432 0 0 1
Value:0
const
176 464 0 0 1
Value:0
const
176 496 0 0 1
Value:2
WIRE
272 400 0 0 0
0 -4 32 4
WIRE
304 464 1 0 0
-4 -64 4 96
WIRE
240 560 0 0 0
-32 -4 64 4
WIRE
208 528 1 0 0
-4 0 4 32
mux8_s
528 1008 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:U10
ProSeq:60
T_Na:T

T_No:4
const
464 1008 0 0 1
Value:0
const
464 1040 0 0 1
Value:0
const
464 1072 0 0 1
Value:0
const
464 1104 0 0 1
Value:0
const
464 1136 0 0 1
Value:0
const
464 1168 0 0 1
Value:1.5
const
464 1200 0 0 1
Value:0
const
464 1232 0 0 1
Value:0
const
464 1264 0 0 1
Value:0
const
464 1328 0 0 1
Value:1
WIRE
432 1296 0 0 0
-32 -4 64 4
WIRE
400 880 1 0 0
-4 -384 4 416
mux8_s
528 560 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1

X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:110
ProSeq:40
T_Na:T
T_No:4
const
464 560 0 0 1
Value:0
const
464 592 0 0 1
Value:0
const
464 624 0 0 1
Value:0
const
464 656 0 0 1
Value:1
const
464 688 0 0 1
Value:0
const
464 720 0 0 1
Value:0
const
464 752 0 0 1
Value:0
const
464 784 0 0 1
Value:0
const
464 816 0 0 1
Value:0
const
464 880 0 0 1
Value:1
WIRE
432 848 0 0 0
-32 -4 64 4

jumper
400 464 1 0 0
WIRE
400 400 1 0 0
-4 -32 4 32
WIRE
432 400 1 0 0
-4 -32 4 32
WIRE
720 464 0 0 0
-416 -4 448 4
WIRE
592 432 0 0 0
-160 -4 160 4
mux8_s
912 1008 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:G10
ProSeq:20
T_Na:T
T_No:4
const
848 1008 0 0 1
Value:0
const
848 1040 0 0 1
Value:0
const
848 1072 0 0 1
Value:0
const
848 1104 0 0 1
Value:0

```
const
848 1136 0 0 1
Value:0
const
848 1168 0 0 1
Value:0
const
848 1200 0 0 1
Value:0
const
848 1232 0 0 1
Value:0
const
848 1264 0 0 1
Value:0
const
848 1328 0 0 1
Value:1
mux8_s
912 560 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:G10
ProSeq:20
T_Na:T
T_No:4
const
848 560 0 0 1
Value:0
const
848 592 0 0 1
Value:0
const
848 624 0 0 1
```

APPENDIX 3 : DRAFT files for Test No. 2

Value:0
const
848 656 0 0 1
Value:0
const
848 688 0 0 1
Value:0
const
848 720 0 0 1
Value:0
const
848 752 0 0 1
Value:0
const
848 784 0 0 1
Value:0
const
848 816 0 0 1
Value:0
const
848 880 0 0 1
Value:1
mux8_s
912 80 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:G10
ProSeq:20
T_Na:T
T_No:4
const
848 80 0 0 1
Value:0
const

848 112 0 0 1
Value:0
const
848 144 0 0 1
Value:0
const
848 176 0 0 1
Value:0
const
848 208 0 0 1
Value:0
const
848 240 0 0 1
Value:0
const
848 272 0 0 1
Value:0
const
848 304 0 0 1
Value:0
const
848 336 0 0 1
Value:0
const
848 400 0 0 1
Value:1
WIRE
752 400 1 0 0
-4 -32 4 32
WIRE
816 368 0 0 0
-64 -4 64 4
WIRE
624 80 0 0 0
-32 -4 32 4
WIRE
656 48 1 0 0
-4 0 4 32
WIRE
912 48 0 0 0
-256 -4 256 4
WIRE
624 560 0 0 0
-32 -4 32 4

WIRE
656 528 1 0 0
-4 -32 4 32
WIRE
912 496 0 0 0
-256 -4 256 4
WIRE
1072 560 0 0 0
-96 -4 96 4
WIRE
1072 800 0 0 0
-96 -4 96 4
WIRE
816 848 0 0 0
-32 -4 64 4
WIRE
592 944 0 0 0
-192 -4 192 4
WIRE
656 1008 0 0 0
-64 -4 32 4
WIRE
688 976 1 0 0
-4 0 4 32
WIRE
912 976 0 0 0
-224 -4 256 4
WIRE
784 880 1 0 0
-4 -32 4 64
jumper
784 976 1 0 0
WIRE
784 1136 1 0 0
-4 -128 4 160
WIRE
816 1296 0 0 0
-32 -4 64 4
WIRE
1072 1008 0 0 0
-96 -4 96 4
export
1200 1008 2 0 1
Name:xud

export
1200 976 2 0 1
Name:wud
export
1200 560 2 0 1
Name:xid
export
1200 496 2 0 1
Name:wid
export
1200 464 2 0 1
Name:reset
export
1200 80 2 0 1
Name:xgam
export
1200 48 2 0 1
Name:wgam
mux8_s
528 80 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:G10
ProSeq:20
T_Na:T
T_No:4
const
464 80 0 0 1
Value:0
const
464 112 0 0 1
Value:0.5
const
464 144 0 0 1

Value:0
const
464 176 0 0 1
Value:0
const
464 208 0 0 1
Value:0
const
464 240 0 0 1
Value:0
const
464 272 0 0 1
Value:0
const
464 304 0 0 1
Value:0
const
464 336 0 0 1
Value:0
const
464 400 0 0 1
Value:1
modifier
208 400 0 0 18
Name:
Com:
EnSw:1
Tpl:
Tp2:O2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
208 432 0 0 18

Name:
Com:
EnSw:1
Tp1:
Tp2:O2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
272 368 0 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:O2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
pgb
1136 48 1 0 6
DL:m1
Group:MUX
Scale:1.0
Max:2.0

Min:-2.0
Units:
pgb
1136 80 1 0 6
DL:m2
Group:MUX
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 464 1 0 6
DL:reset
Group:
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 496 1 0 6
DL:m3
Group:MUX
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 560 1 0 6
DL:m4
Group:MUX
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 976 1 0 6
DL:m5
Group:MUX
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 1008 1 0 6

DL:m6
Group:MUX
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
304 368 3 0 6
DL:tr
Group:
Scale:1.0
Max:2.0
Min:-2.0
Units:
modifier
208 496 0 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:O2
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
SUBSYSTEM-TITLE: PI
SUBSYSTEM-COMMENT:
SUBSYSTEM-PRINTMODE: AUTO PLOTMODE: AUTO
184 COMPONENTS
pl16s_s
432 272 0 0 28
A1N:1
B1N:1
A2N:1
B2N:1

APPENDIX 3 : DRAFT files for Test No. 2

A3N:1
B3N:1
A4N:1
B4N:1
A5N:1
B5N:1
A6N:1
B6N:1
AName:G2
ProSeq:20
T_Na:T
T_No:1
A1:-200
B1:-20
A2:-100
B2:-15
A3:-50
B3:0
A4:0
B4:20
A5:100
B5:100
A6:300
B6:150
WIRE
336 272 0 0 0
-64 -4 64 4
sumjcts2_s
240 720 0 0 6
AName:I1
ProSeq:40
T_Na:T
T_No:1
DPath:1
Type:1
WIRE
176 720 0 0 0
0 -4 32 4
WIRE
176 752 0 0 0
0 -4 32 4
import
144 720 0 0 1
Name:wid

import
144 752 0 0 1
Name:xid
pl16s_s
592 720 0 0 28
A1N:1
B1N:1
A2N:1
B2N:1
A3N:1
B3N:1
A4N:1
B4N:1
A5N:1
B5N:1
A6N:1
B6N:1
AName:I2
ProSeq:50
T_Na:T
T_No:1
A1:-200
B1:-20
A2:-100
B2:-15
A3:-50
B3:0
A4:0
B4:20
A5:100
B5:100
A6:300
B6:150
WIRE
400 720 0 0 0
-128 -4 160 4
WIRE
752 720 0 0 0
-96 -4 96 4
nsws_s
784 272 0 0 4
AName:G3
ProSeq:30
T_Na:T

T_No:1
WIRE
624 272 0 0 0
-128 -4 128 4
const
720 304 0 0 1
Value:1.9999
const
720 336 0 0 1
Value:0
WIRE
880 272 0 0 0
-64 -4 96 4
pl16s_s
816 1168 0 0 28
A1N:1
B1N:1
A2N:1
B2N:1
A3N:1
B3N:1
A4N:1
B4N:1
A5N:1
B5N:1
A6N:1
B6N:1
AName:U5
ProSeq:100
T_Na:T
T_No:1
A1:-200
B1:-20
A2:-100
B2:-15
A3:-50
B3:0
A4:0
B4:20
A5:100
B5:100
A6:300
B6:150
sumjcts2_s

APPENDIX 3 : DRAFT files for Test No. 2

240 1168 0 0 6
AName:U1
ProSeq:60
T_Na:T
T_No:l
DPath:l
Type:l
WIRE
176 1168 0 0 0
-32 -4 32 4
WIRE
176 1200 0 0 0
-32 -4 32 4
import
112 1168 0 0 1
Name:wud
import
112 1200 0 0 1
Name:xud
sumjcts2_s
240 1328 0 0 6
AName:U2
ProSeq:70
T_Na:T
T_No:l
DPath:l
Type:l
WIRE
176 1360 0 0 0
0 -4 32 4
WIRE
208 1264 1 0 0
-4 -64 4 64
WIRE
176 1296 1 0 0
-4 -64 4 64
jumper
176 1200 3 0 0
nsws_s
464 1168 0 0 4
AName:U3
ProSeq:80
T_Na:T
T_No:l

nsws_s
464 1296 0 0 4
AName:U4
ProSeq:90
T_Na:T
T_No:1
WIRE
336 1168 0 0 0
-64 -4 96 4
WIRE
336 1328 0 0 0
-64 -4 96 4
WIRE
624 1168 0 0 0
-128 -4 160 4
GROUP
752 1392 0 0 0
12
const
752 1200 0 0 1
Value:-2
const
752 1232 0 0 1
Value:-2
const
752 1264 0 0 1
Value:-0.2
const
752 1296 0 0 1
Value:-0.2
const
752 1328 0 0 1
Value:-0.1
const
752 1360 0 0 1
Value:-0.1
const
752 1392 0 0 1
Value:0.1
const
752 1424 0 0 1
Value:0.1
const
752 1456 0 0 1

APPENDIX 3 : DRAFT files for Test No. 2

Value:0.2
const
752 1488 0 0 1
Value:0.2
const
752 1520 0 0 1
Value:1.9999
const
752 1552 0 0 1
Value:1.9999
const
528 752 0 0 1
Value:-2
const
528 784 0 0 1
Value:-2
const
528 816 0 0 1
Value:-0.2
const
528 848 0 0 1
Value:-0.2
const
528 880 0 0 1
Value:-0.1
const
528 912 0 0 1
Value:-0.1
const
528 944 0 0 1
Value:0.1
const
528 976 0 0 1
Value:0.1
const
528 1008 0 0 1
Value:0.2
const
528 1040 0 0 1
Value:0.2
const
528 1072 0 0 1
Value:1.9999
const

528 1104 0 0 1
Value:1.9999
const
400 1200 0 0 1
Value:1.9999
consti
400 1232 0 0 1
Value:0
consti
400 1360 0 0 1
Value:0
const
400 1296 0 0 1
Value:-1.9999
WIRE
592 1296 0 0 0
-96 -4 96 4
WIRE
688 1232 1 0 0
-4 -32 4 64
WIRE
688 1040 1 0 0
-4 -64 4 96
jumper
688 1168 1 0 0
WIRE
752 976 0 0 0
-64 -4 96 4
csbs_s
1008 272 0 0 5
AName:Select
ProSeq:110
T_Na:T
T_No:1
M:0
WIRE
848 496 1 0 0
-4 -192 4 224
WIRE
912 304 0 0 0
-64 -4 64 4
WIRE
880 752 1 0 0
-4 -416 4 416

WIRE
912 336 0 0 0
-32 -4 64 4
jumper
880 976 2 0 0
WIRE
912 656 1 0 0
-4 -288 4 320
WIRE
944 368 0 0 0
-32 -4 32 4
consti
944 400 0 0 1
Value:0
WIRE
1328 272 0 0 0
-288 -4 288 4
mux8_s
1168 528 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1
X8:1
N:1
CCI:1
CCS:1
QF:1
AName:O1
ProSeq:120
T_Na:T
T_No:1
mux8_s
1168 944 0 0 16
X1:1
X2:1
X3:1
X4:1
X5:1
X6:1
X7:1

X8:1
N:1
CCI:1
CCS:1
QF:1
AName:O3
ProSeq:140
T_Na:T
T_No:1
ptls_s
1392 528 0 0 6
AName:O2
ProSeq:130
T_Na:T
T_No:1
ET:1
T:.01
PIC201_s
1648 272 0 0 18
EN:1
ESV:1
S:1
HI:1
W2:1
X1:1
X2:1
WP:1
IC:1
YE:1
YI:1
QU:1
QL:1
AName:O4
ProSeq:150
T_Na:T
T_No:1
SV:0.
const
1584 304 0 0 1
Value:0
const
1584 336 0 0 1
Value:0
const

1584 368 0 0 1
Value:0
const
1584 400 0 0 1
Value:0
const
1584 432 0 0 1
Value:0.996
const
1584 464 0 0 1
Value:-0.94
const
1584 496 0 0 1
Value:0
WIRE
1040 752 1 0 0
-4 -448 4 480
WIRE
1072 1232 0 0 0
-32 -4 64 4
WIRE
1072 816 0 0 0
-32 -4 64 4
const
1104 848 0 0 1
Value:1
const
1104 1264 0 0 1
Value:1
const
1104 944 0 0 1
Value:4
const
1104 976 0 0 1
Value:3
const
1104 1008 0 0 1
Value:2
const
1104 1040 0 0 1
Value:1
const
1104 1072 0 0 1
Value:0.01

const
1104 1104 0 0 1
Value:0.01
const
1104 1136 0 0 1
Value:0.01
const
1104 1168 0 0 1
Value:0.01
const
1104 1200 0 0 1
Value:0.01
const
1104 528 0 0 1
Value:0.4
const
1104 560 0 0 1
Value:0.3
const
1104 592 0 0 1
Value:0.2
const
1104 624 0 0 1
Value:0.1
const
1104 656 0 0 1
Value:1
const
1104 688 0 0 1
Value:1
const
1104 720 0 0 1
Value:1
const
1104 752 0 0 1
Value:1
const
1104 784 0 0 1
Value:1
WIRE
1296 528 0 0 0
-64 -4 64 4
const
1328 560 0 0 1

Value:0.01
WIRE
1520 528 0 0 0
-96 -4 96 4
WIRE
1552 560 0 0 0
-64 -4 64 4
WIRE
1488 752 1 0 0
-4 -192 4 192
WIRE
1360 944 0 0 0
-128 -4 128 4
const
1584 592 0 0 1
Value:0
const
1584 624 0 0 1
Value:1
const
1584 688 0 0 1
Value:0
WIRE
1552 656 0 0 0
-32 -4 64 4
jumper2
1520 560 3 0 0
WIRE
1520 624 1 0 0
-4 -32 4 32
jumper
1520 272 3 0 0
WIRE
1520 400 1 0 0
-4 -96 4 96
WIRE
1520 208 1 0 0
-4 0 4 32
WIRE
848 208 0 0 0
-672 -4 672 4
sumjcts2_s
240 272 0 0 6
AName:G1

ProSeq:10
T_Na:T
T_No:1
DPath:1
Type:1
WIRE
176 272 0 0 0
0 -4 32 4
WIRE
176 304 0 0 0
0 -4 32 4
import
144 272 0 0 1
Name:wgam
import
144 304 0 0 1
Name:xgam
import
144 208 0 0 1
Name:reset
pgb
1712 272 0 0 6
DL:output
Group:INT
Scale:1
Max:60
Min:0
Units:
pgb
944 368 3 0 6
DL:s4
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
912 336 3 0 6
DL:s3
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:

pgb
880 304 3 0 6
DL:s2
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
848 272 3 0 6
DL:s1
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1136 272 3 0 6
DL:so
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
const
368 304 0 0 1
Value:-2
const
368 336 0 0 1
Value:-2
const
368 368 0 0 1
Value:-0.2
const
368 400 0 0 1
Value:-0.2
const
368 432 0 0 1
Value:-0.1
const
368 464 0 0 1
Value:-0.1
const
368 496 0 0 1

Value:0.1
const
368 528 0 0 1
Value:0.1
const
368 560 0 0 1
Value:0.2
const
368 592 0 0 1
Value:0.2
const
368 624 0 0 1
Value:1.9999
const
368 656 0 0 1
Value:1.9999
pgb
528 272 3 0 6
DL:tt2
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
304 272 3 0 6
DL:tt1
Group:SELECT
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1040 432 0 0 6
DL:SelWord
Group:
Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1456 528 1 0 6
DL:kp
Group:INT

Scale:1.0
Max:2.0
Min:-2.0
Units:
pgb
1488 656 2 0 6
DL:tn
Group:INT
Scale:1.0
Max:2.0
Min:-2.0
Units:
modifier
496 1296 0 0 18
Name:dUdRec
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
880 1168 1 0 18
Name:dUdInv
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4

Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
976 272 0 0 18
Name:dGama
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
976 304 0 0 18
Name:dId
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:

EnMax:0
Max:
EnMin:0
Min:
modifier
976 336 0 0 18
Name:dUdInv
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
976 368 0 0 18
Name:dUdRec
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:

modifier
1136 528 0 0 18
Name:
Com:KP gama
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 560 2 0 18
Name:
Com:KP Id
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 592 2 0 18
Name:
Com:Kp Ud

EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 624 2 0 18
Name:
Com:KP Ud Inverter
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 656 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0

Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 688 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 720 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4

Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 752 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 784 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:

EnMax:0
Max:
EnMin:0
Min:
modifier
1136 816 2 0 18
Name:
Com:switch over by max selection
EnSw:0
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 944 0 0 18
Name:
Com:TN gama
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:

modifier
1136 976 2 0 18
Name:
Com:TN Id
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1008 2 0 18
Name:
Com:TN Ud rectifier
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1040 2 0 18
Name:
Com:TN Ud inverter

**EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1072 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1104 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:R2
EnScal:0**

**Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1136 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1168 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4**

Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1200 2 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:R2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1136 1232 2 0 18
Name:
Com:switch over by max selection
EnSw:0
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:

EnMax:0
Max:
EnMin:0
Min:
modifier
1360 528 0 0 18
Name:
Com:
EnSw:1
Tp1:
Tp2:E2
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1360 560 2 0 18
Name:
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:1
Unit0:2
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:

modifier
1424 528 0 0 18
Name:KP
Com:
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 272 0 0 18
Name:
Com:set point 1
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 304 2 0 18
Name:
Com:set point 2

EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 336 2 0 18
Name:
Com:actual value 1
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 368 2 0 18
Name:
Com:actual value 2
EnSw:0
Tp1:
Tp2:
EnScal:0

Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 400 2 0 18
Name:
Com:pre-control value
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 432 2 0 18
Name:
Com:upper limit value
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4

Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 464 2 0 18
Name:
Com:lower limit value
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 496 2 0 18
Name:
Com:setting value
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:

EnMax:0
Max:
EnMin:0
Min:
modifier
1616 528 2 0 18
Name:
Com:propotional coefficient
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 560 2 0 18
Name:
Com:reset time
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:

modifier
1616 592 2 0 18
Name:
Com:I-controller
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 624 2 0 18
Name:
Com:conrtooler enable
EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
modifier
1616 688 2 0 18
Name:
Com:hold integrater

EnSw:0
Tp1:
Tp2:
EnScal:0
Scal:1
EnUnit:0
Unit0:0
Unit1:4
Unit2:4
Unit3:4
EnInit:0
Init:
EnMax:0
Max:
EnMin:0
Min:
SEQUENCER DATA [916,260]
Subsystem #1: Repeat
Subsystem #2: Repeat


```

1 FP-PARA.P16
&V4.2.1 FP-DIA UNC 08.07.96 08:54
& FBSLIB 950401V420 A:
&B: C:
&&0005563
2CT1 4S = 'test "Inhalt/Content"
3CT2 4S = ' '
4CT3 4S = ' '
5HLD 3S = ' "Anlagenkennz./Higher lev.design"
6DLS 1S = ' "Zeichn.Nr./Draw. No List suffix"
7DGS 1S = ' "Graphic suffix"
8DPS 1S = ' "Besteller/Purchaser suffix"
9DES 2S = 'gast "Bearbeiter/Designer "
10ORD 4S = ' "Urspr/Original document"
11MD3 4S = ' / /10-21/96/ "Modification"
12MD2 4S = ' / / / '
13MD1 4S = ' / / / '
14+
15 TX=T1
16 SAW20 : ADD2 ,POS=01.01.01 "Ramp control"
&&0011000A 2003B
17X1 N2 < @TYP=R2,10[s]
18X2 N2 < 0%
19Y N2 > @TYP=R2,$TSAW "abcdefg"
20+
21
22 SIN30 : ADD2 ,POS=02.03.07 "Sin wave period"
&&0011000A 2003B
23X1 N2 < @TYP=R2,0[s]
24X2 N2 < 0%
25Y N2 > $TSIN
26+
27
28 TX=T4
29 SAW10 : NSW ,POS=09.08.97 "saw tooth"
&&0012000E 2404U
30X1 N2 < 100%
31X2 N2 < -100%
32I B1 < $CDOWN
33Y N2 > $WSAW "Set value for Saw tooth generator"
34+
35
36 SIN10 : ADD2 ,POS=01.01.01

```

APPENDIX 4 : STRUC-L files for Test No. 2

```

&&0011000A 2003B
37X1 N2 < 150%
38X2 N2 < 0%
39Y N2 > $LUP          "Upper limit for sinus generator"
40+
41
42 SIN20 : ADD2 ,POS=01.01.01
&&0011000A 2003B
43X1 N2 < -150%
44X2 N2 < 0%
45Y N2 > $LLOW        "Lower limit for sinus generator"
46+
47
48 CM,POS=01
49 "Adjustment parameters."
50 END
51
????????????????????????????????????????????????????????????????????????????????????

```

```

1 FP-SINUS,P16
&V4.2.1 FP-DIA UNC 08.07.96 08:54
& FBSLIB 950401V420 A:
&B:          C:
&&0005563
2CT1 4S = 'test          ""Inhal/Content""
3CT2 4S = '              '
4CT3 4S = '              '
5HLD 3S = '              ""Anlagenkennz./Higher lev.design""
6DLS 1S = ' '            ""Zzeichn.Nr./Draw. No List suffix""
7DGS 1S = ' '            " . . . Graphic suffix"
8DPS 1S = ' '            "  Besteller/Purchaser suffix"
9DES 2S = 'gast         "Bearbeiter/Designer "
10ORD 4S = '            ""Urspr/Original document""
11MD3 4S = ' /          /10-21/96/ ""Modification""
12MD2 4S = ' /          / / '
13MD1 4S = ' /          / / '
14+
15 TX=T1
16 SIN10 : INT ,POS=01.01.01

```

```

&&004D0018 2001G
17X N2 < SIN30.Y
18LU N2 < $LUP
19LL N2 < $LLOW
20SV N2 < 100%
21TI R2 < $TSIN
22S B1 < SIN50.QP
23Y N2 >
24QU B1 >
25QL B1 >
26+
27
28 SIN20 :INT ,POS=01.01.01
&&004D0018 2001G
29X N2 < SIN10.Y
30LU N2 < $LUP
31LL N2 < $LLOW
32SV N2 < 0%
33TI R2 < $TSIN
34S B1 < 0
35Y N2 > ,Sinus'
36QU B1 >
37QL B1 >
38+
39
40 SIN30 :SIH ,POS=01.01.01
&&000D0008 2003A
41X N2 < SIN20.Y
42Y N2 > ,INIT=100%
43+
44
45 SIN40 :PDE ,POS=01.01.01
&&00220010 2404N
46I B1 < 1
47T T2 < 0[s]
48Q B1 >
49+
50
51 SIN50 :ETE ,POS=01.01.01
&&0018000E 2404I
52I B1 < SIN40.Q
53QP B1 >
54QN B1 >
55+

```

56
 57 CM,POS=01
 58 "kgjhghbj"
 59 END
 60
 ???

1 FP-SAW,P16
 &V4.2.1 FP-DIA UNC 08.07.96 08:54
 & FBSLIB 950401V420 A:
 &B: C:
 &&0005563
 2CT1 4S = 'test "Inhalt/Content"
 3CT2 4S = '
 4CT3 4S = '
 5HLD 3S = ' "Anlagenkennz./Higher lev.design"
 6DLS 1S = ' "Zeichn.Nr./Draw. No List suffix"
 7DGS 1S = ' "Graphic suffix"
 8DPS 1S = ' "Besteller/Purchaser suffix"
 9DES 2S = 'gast "Bearbeiter/Designer "
 10ORD 4S = ' "Urspr/Original document"
 11MD3 4S = ' / /10-21/96/ "Modification"
 12MD2 4S = ' / / / '
 13MD1 4S = ' / / / '
 14+
 15 TX=T1
 16 SAW10 : INT ,POS=01.01.01
 &&004D0018 2001G
 17X N2 < \$WSAW
 18LU N2 < 100%
 19LL N2 < -100%
 20SV N2 < 0%
 21TI R2 < \$TSAW,'Speed'
 22S B1 < 0
 23Y N2 > ,SCAL=1000[MW]
 24QU B1 >
 25QL B1 >
 26+
 27

APPENDIX 4 : STRUC-L files for Test No. 2

```
28 SAW20 : RSS ,POS=01.01.01
&&0016000C 2004K
29S B1 < SAW10.QU
30R B1 < SAW10.QL
31Q B1 > $CDOWN
32QN B1 >
33+
34
35 CM,POS=01
36 "Saw tooth generator"
37 END
38
????????????????????????????????????????????????????????????????
```

```

////////////////////////////////////
////
// file:global.h//
// Description:Definition of global classes, data structures and//
//functions//
// Created:Wednesday, Jul, 3, 1996//
// Author:Michael H. Xie//
// Email:mhxie@ee.umanitoba.ca//
////
////////////////////////////////////

#ifndef GLOBAL_H
#define GLOBAL_H

#include <iostream.h>
#include <fstream.h>
#include <libc.h>
#include <ctype.h>

#include <LEDA/string.h>
#include <LEDA/set.h>
#include <LEDA/point.h>
#include <LEDA/sortseq.h>
#include <LEDA/list.h>

const int    GRID = 32;
const intLONGEST_FILE_NAME = 100;
const string NOT_COMPILE_STRING = "NOT_STRUC";
const stringNODE_NAME_BASE_STRING = "_Node_";
const stringmpFileName = "mpinfo.txt";
const stringstdLibName = "FBSLIB 950401V420";
const stringfakedChecksum = "&&0005563";
const stringdefaultPos = "01.01.01";

enum BOOL{ FALSE=0, TRUE=1 };
enum CONTYPE{ COORD=2, CONSTIN=3, VOIDOUT=4, SAMPLINGTIME=5
};
enum NODE_TYPE { OUTPUT_PIN=6, CONST_IN=7, EX_IN=8, EX_OUT=9,
VOID=10 };
enum IN_OUT_PUT { IN=11, OUT=12 };
enum CONST_TYPE { INT_CONST=13, FLOAT_CONST=14,
TYPE_NOT_SURE_CONST=15 };

```

```

////////////////////////////////////
// inline functions

inline int min(int i1, int i2) { return ((i1 < i2) ? i1:i2); }
inline int max(int i1, int i2) { return ((i1 < i2) ? i2:i1); }

// error function
inline int error(char* msg)
{
    cerr << msg << endl;
    exit(10);
}

// string to integer & floating point number

inline int stoi(const string & s0)
{
    intbits, temp;
    char *number;

    bits = s0.length();
    number = new char[bits+1];
    for(int i=0; i<bits; i++)
        number[i] = s0[i];
    number[bits] = '\0';
    temp = atoi(number);
    delete []number;
    return temp;
}

inline float stof(const string & s0)
{
    intbits;
    float temp;
    char *number;

    bits = s0.length();
    number = new char[bits+1];
    for(int i=0; i<bits; i++)
        number[i] = s0[i];
    number[bits] = '\0';
    temp = atof(number);
    delete []number;
    return temp;
}

```

```

}

////////////////////////////////////
// fill space chars after the string to make it desired length
// cut it if it is too long

inline string strFixLen(string s0, int len)
{
string tempStr = s0;
if(s0.length() > len) {
return s0.head(len);
}

for(int i=s0.length(); i<len; i++)
tempStr += " ";
return tempStr;
}

string readCompType(istream&);
BOOL pSetIntersect(set<point>, set<point>);
set<point> pSetUnion(set<point>, set<point>);

////////////////////////////////////
// global data structures

struct PinInfo {
stringfbName;
string pinName;
string connectorType;
IN_OUT_PUT inOrOut;
BOOLqMark;
pointpinPos;// for checking modifier map
CONTYPEconnection;
stringnodeName;// if is a COORD
stringconstIn;// if is a CONSTIN (format guaranteed in dataBase)
void print();
};

struct SpecInfoListStruc {
string content1;
string content2;
string content3;
}

```



```

string creator;
string modifier;
string modifyDate;
string createDate;
string time;
int numOfSubSys;
string subSysName;
string truncatedSubSysName;
string subSysComments;
int numOfComp;
string T1;
string T2;
string T3;
string T4;
string T5;
void print();
};

struct ModifierStruc {
string sigName; // without single quote
string sigCom;
string typeSwitch; // the complete type switch string
string switchTo; // the final connector type
bool enScal;
string scale;
string unit;
string limiting;
string initCondition;
void print();
};

struct CompInfoCl {
// constructors
public:
CompInfoCl ();

// member variables, public by default
string appName;
string fType;
string fbName;
string fbCom; // F.B. comments
string secCode;
string pos; // for load back

```

```

intexSeq;
string sampTime;
list<PinInfo>pinList;
void print();
};

typedef sortseq<int, CompInfoCI> CompSameSTimeCI;

struct NodeInfoCI {
// constructors
public:
NodeInfoCI() { nodeType=VOID; constantType=TYPE_NOT_SURE_CONST; err-
Flag=FALSE; };
string getName() { return nodeName; };

// member variables
set<point> pointSet;
NODE_TYPE nodeType;
string  appName;// if nodeType = output pin
string  connName;
CONST_TYPE constantType;// if nodeType = constant
float  val;
string  scale;
string  unit;
string  exInName;// w/o $ sign
string  exOutName;
string  nodeNameBase;// _NODE_#_
int  nodeNameAddOn;// pin number
string  nodeName;// _NODE_#_pin number or other
string  collapsed2Node;
list<PinInfo>pinPool;// list of pins connected to this node
BOOL  errFlag;// Indicate any wrong connection between
// critical pins, (e.g. 2 outputs together
void print();

};

#endif

```

```

////////////////////////////////////
///
// File:dataBase.h//
// Description:declaration of lookup tables and their basic operations//
// Created:Wednesday, Jul, 3, 1996//
// Author:Michael Xie//
// Email:mhxie@ee.umanitoba.ca//
///
////////////////////////////////////

#ifndef DATABASE_H
#define DATABASE_H

#include "global.h"
#include <stdio.h>

class GeneralComp;

// structure of a record for function block name table
struct FBNameEntry {
charcompName[20];
enumBOOLcondi;
charcritPara[16];
charcritVal[35];
charfBName[11];
intaddrStart;
intaddrEnd;
charsecCode1[20];
charsecCode2[20];
charlib[23];
};

/* structure of a record for function block connector table */
struct FBConnEntry {
charfBName[11];
intseq;
charconnName[4];
charconnType[4];
enumIN_OUT_PUTinOrOut;
enumBOOLqMark;
enumBOOLcondi;
charcritPara[16];
charcritVal[35];
enumCONTYPEconnectType;
};

```

```

intxCoord;
intyCoord;
charconstVal[20];
};

```

```

class FBNameTableCI {

//constructors & destructors
public:
FBNameTableCI() {};
FBNameTableCI(string);

// basic operations
public:
BOOL lookUp(GeneralComp&, FBNameEntry&);

// variables
private:
stringfileName;
charfileNameChar[LONGEST_FILE_NAME];

};

```

```

class FBConnTableCI {

//constructors & destructors
public:
FBConnTableCI() {};
FBConnTableCI(string);

// basic operations
public:
BOOL lookUp(GeneralComp&, FBNameEntry&, list<FBConnEntry>&);

// variables
private:
stringfileName;
charfileNameChar[LONGEST_FILE_NAME];

};

```

```

class convertDBCI {
// constructors
public:

```

```

convertDBCI() {};
convertDBCI(string);

// operations
public:
BOOL dbLookUp(GeneralComp&, FBNameEntry&, list<FBConnEntry>&);

// variables
private:
string dbName;
FBNameTableCI nameTab;
FBConnTableCI connectorTab;
};

void FBNameEntryprint(FBNameEntry );
void FBConnEntryprint(FBConnEntry );

#endif

```

```

//////////////////////////////////////////////////////////////////
///
// File:generalComp.h//
// Description:definition of general user-defined componens//
// Created:Wednesday, Jun, 27, 1996//
// Author:Michael Xie//
// Email:mhxie@ee.umanitoba.ca//
///
//////////////////////////////////////////////////////////////////

#ifndef GENERAL_COMP_H
#define GENERAL_COMP_H

#include "global.h"
#include "PSCADcomp.h"

class GeneralComp : public PSCADcomp {

// basic operations
public:

```

```
int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameSTimeCl>&,
sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&, string);
```

```
virtual int GeneralComp::process(sortseq<string, NodeInfoCl>&, sortseq<string,
CompSameSTimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&)
{
    cout << "No method Process(nodeMap, compSeq, modifierMap, ";
    cout << "specInfoList, libList) defined for GeneralComp, a ";
    cout << "database name has to be specified\n";
    exit(3);
}
```

```
friend class FBNameTableCl;
friend class FBConnTableCl;
friend class convertDBCl;
```

```
};
```

```
#endif
```

```
////////////////////////////////////
///
// File:PSCADcomp.h//
// Description:declaration of PSCAD component concept, structure and//
//basic operations//
// Created:Wednesday, Jun, 27, 1996//
// Author:Michael Xie//
// Email:mhxie@ee.umanitoba.ca//
///
////////////////////////////////////
```

```
#ifndef PSCAD_COMP_H
#define PSCAD_COMP_H
```

```
#include "global.h"
```

```
class PSCADcomp {
```

```

    // variables
    protected:
    intcompCnt;// to make node name base
    charcompCntStr[5];// convert the count to string
    stringcompType;
    intinsertX, insertY;// insertion point coordinate
    int orientation;// 0: 0 deg.  1: 90 deg. clockwise
    // 2: 180 deg.  3: 270 deg. clockwise
    intmirror;// 0:No  1:Yes
    intnumOfParams;

    sortseq<string, string> params;

    // basic operations
    public:
    PSCADComp() {};
    void assignType(string);// assign the type to PSCADcomp
    virtual void readHead(istream&);// read the 9 numbers
    void readOthers(istream&);// read the params & values
    void putCount(int);// for generating node base
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>& ) = 0;
    print();

};

#endif

```

```

/////////////////////////////////////////////////////////////////
////
// File:SpecComp.h//
// Description:declaration of special componens, i.e. wire, jumper//
//pgb, dotted line, sampling table, ... everything needed//
//to be treated specially//
// Created:Wednesday, Jun, 27, 1996//
// Author:Michael H. Xie//
// Email:mhxie@ee.umanitoba.ca//
////
/////////////////////////////////////////////////////////////////

```

```

#ifndef SPECCOMP_H
#define SPECCOMP_H

#include "PSCADcomp.h"
#include "global.h"

/////////////////////////////////////////////////////////////////
// WIRES

class WIRE : public PSCADcomp {
    // constructors
    public:
    WIRE() {};

    // variables
    private:
    intcoefXStart, coefYStart;
    intcoefXEnd, coefYEnd, coefMirror;
    int startX, startY, endX, endY;
    intoffsetX1, offsetY1;
    intoffsetX2, offsetY2;
    intlowX, highX, lowY, highY;// for the "for" loop
    pointp3;
    NodeInfoClderNode;

    // operations
    public:
    virtual void readHead(istream&);
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
};

/////////////////////////////////////////////////////////////////
// jumpers

class jumper : public PSCADcomp {

    // constructors
    public:
    jumper() {};

    // variables

```



```

        private:
        pointp3, p4;
        NodeInfoClderNode;

        // operations

        public:
        virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
        };

        ////////////////////////////////////////////////////
        // floating point constants

        class constant: public PSCADcomp {

        private:
        point p3;
        NodeInfoClderNode;

        public:
        virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
        };

        ////////////////////////////////////////////////////
        // integer constants

        class consti : public PSCADcomp {

        private:
        point p3;
        NodeInfoClderNode;

        public:
        virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
        };

        ////////////////////////////////////////////////////
        // import signals

```

```

class import : public PSCADcomp {

    private:
    point p3;
    NodeInfoClderNode;

    public:
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
};

////////////////////////////////////
// export signals

class export : public PSCADcomp {

    private:
    NodeInfoClderNode;
    point p3;

    public:
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
};

////////////////////////////////////
// annotation boxes (do nothing)

class anotation : public PSCADcomp {

    public:
    virtual int process(sortseq<string, NodeInfoCl>& z1, sortseq<string, CompSameS-
TimeCl>& z2, sortseq<point, ModifierStruc>& z3, SpecInfoListStruc& z4, list<string>& z5 ) { };
};

////////////////////////////////////
// plotting graphics boxes (do nothing)

class pgb : public PSCADcomp {

```

```

        public:
        virtual int process(sortseq<string, NodeInfoCl>& z1, sortseq<string, CompSameS-
TimeCl>& z2, sortseq<point, ModifierStruc>& z3, SpecInfoListStruc& z4, list<string>& z5 ) {};
};

////////////////////////////////////
// sampling tables

class sampletables : public PSCADcomp {

    public:
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
};

////////////////////////////////////
// dashed lines (do nothing)

class DASHED_LINE : public PSCADcomp {

    public:
    virtual int process(sortseq<string, NodeInfoCl>& z1, sortseq<string, CompSameS-
TimeCl>& z2, sortseq<point, ModifierStruc>& z3, SpecInfoListStruc& z4, list<string>& z5 ) {};
};

////////////////////////////////////
// modifier

class modifier : public PSCADcomp {

    private:
    point p3;

    public:
    virtual int process(sortseq<string, NodeInfoCl>&, sortseq<string, CompSameS-
TimeCl>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);
};

////////////////////////////////////
// Wire Label, the "process" method takes care of the connection, it is
// responsible for putting wires together and put it into NodeMap

```

```
// correctly.  
  
class wirelabel : public PSCADcomp {  
  
    private:  
    point p3;  
    NodeInfoClderNode;  
  
    public:  
    virtual int process(sortseq<string, NodeInfoCI>&, sortseq<string, CompSameS-  
TimeCI>&, sortseq<point, ModifierStruc>&, SpecInfoListStruc&, list<string>&);  
};  
  
#endif
```

References

- [1] A. Hammad et al: "Controls Modelling and Verification for the Pacific Intertie HVdc 4-Terminal Scheme", *IEEE Transactions on Power Delivery*, Vol. 8, No. 1, January 1993.
- [2] P. Kuffel, K.L. Kent, G.B. Mazur and M.A. Weekes "Development and Validation of Detailed Controls Models of the Nelson River Bipole I HVdc System". *IEEE Transactions on Power Delivery*, Vol. 8, No. 1, January 1993.
- [3] G. Morin et al: "Modelling of the Hydro Quebec - New England HVdc System and Digital Controls with EMTP", *IEEE Transactions on Power Delivery*, Vol. 8, No. 2, April 1993.
- [4] A.M. Gole, O.B. Nayak, T.S. Sidhu, M.S. Sachdev, "A Graphical Electromagnetic Simulation Laboratory for Power Systems Engineering Programs", *IEEE Trans. on Power Systems*, May 1996
- [5] SIMADYN D Digital Control System, Standard Function Blocks, Siemens AG 1988.
- [6] D. Knittler and L.Huegelschaefer," Experience with a Digital Fully Redundant Control System for HVdc- Plants", *EPE, Firenze*, 1991.

- [7] K. Sadek, G.Wild, L. Huegelschaefer, A. Gole, X. Jiang, D. Brandt "Modelling of Digital HVDC Control Systems using a Graphical Electromagnetic Simulation Program", IPST, Sept. 1995, Lisbon, Portugal.
- [8] James Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, W. Lorensen, "Object Oriented Modeling and Design", Prentice-Hall Inc., 1991.
- [9] B.W.Kernighan, D.M.Ritchie, "The C programming language", Prentice Hal, 1988.
- [10] B.Stroustrup, "The C++ programming language", 2nd edition, Addison-Wesley publishing company, 1991.