AN ABSTRACT OF THE THESIS OF

Priya N. Werahera for the degree of Master of Science in Electrical and Computer Engineering presented on August 11, 1986.

Title: Supervisory Control of a Doubly-Fed Machine

Redacted for Privacy

Abstract approved: ______________________
Roy C. Rathja

An EMCON-D distributed process controller is evaluated for control and monitoring of an experimental doubly-fed machine system. A closed-loop control circuit is implemented to operate the doubly-fed machine with constant rotor current. A standard PID algorithm as well as an algorithm tailored to the doubly-fed machine are evaluated.

Supervisory Control of a Doubly-Fed Machine

by

Priya N. Werahera

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed August 11, 1986

Commencement June 1987

APPROVED:

Redacted for Privacy

Associate Professor of Electrical and Computer Engineering
in charge of major

Redacted for Privacy

Head of department of Electrical and Computer Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented August 11, 1986

ACKNOWLEDGEMENT

I wish to acknowledge first the invaluable assistance and advice of my advisor Prof. Roy Rathja. Professors Hian Lauw, Alan Wallace, Leland Jensen and Randall Jacobson also provided guidance for which I am most grateful. Bob Searfus, Mohammed Siala, Ken McCracken, Arthur Neeley and Bill Molnar provided technical assistance. Gangadhar Burrah and Jean Carrol helped with the typing of the final manuscript and Hadi assisted in the preparation of figures. Their kind assistance is much appreciated. Finally, sincere thanks go to my parents and three sisters for their loyal support and encouragement.

TABLE OF CONTENTS

1. Introduction 1

2. Process Controller 4

   2.1 Hardware Description 4

   2.2 Software Description 9

3. Doubly-fed Machine Control 10

   3.1 Constant Current Mode of Operation 13

   3.2 Control Circuit for Constant Current Operation 20

4. Data Acquisition 22

   4.1 Transducer Selection and Interface Design 22

   4.2 Database Preparation 23

   4.3 Data Recording And Display 24

5. Control Circuit Design and Algorithm Development 26

   5.1 Control Circuit Hardware 26

   5.2 Requirements of the Control Algorithm 28

   5.3 PID Algorithm 34

   5.4 Algorithm Development Using FORTRAN Access 36

6. Constant Current Operation 40

   6.1 PID Algorithm 40

   6.2 FORTRAN Access 42

   6.3 Improvement of the FORTRAN Access Algorithm 45

7. Summary and Conclusions 52

   7.1 Conclusions 52

   7.2 Future Applications 55

Bibliography 56

Appendix A: Ratings of the Electrical Machines 57

Appendix B: Schwarz AC to DC Converter 58

Appendix C: Current Sensor Interconnection 60

Appendix D: CAM Command File 63

Appendix E: GROUP Command File 66

Appendix F: SYSLOD Command File 69

Appendix G: LOGEN Program 72

Appendix H: Historical Trending Package 74

Appendix I: AC to DC Converter Interconnection 76

Appendix J: FORTRAN Program 1 79

Appendix K: FTNACCPRG Command File 81

Appendix L: FORTRAN Program 2 84

<u>LIST OF FIGURES</u>

Figure	Page
2.1 EMCON-D Process Control System	5
2.2 Host Computer	6
3.1 Variable Speed Drive Setup for the Doubly-fed Machine	14
3.2 Doubly-fed Machine with Converter on Rotor Side	17
3.3 Constant Current Operation of the Doubly-fed Machine (for variable speed drives)	19
5.1 Modified Control Circuit for Constant Current Operation	27
5.2 Non-uniform Step Sizes Vs Uniform Step Sizes for the Field Current of the Generator (a) Non-uniform Steps (b) Uniform Steps	31
5.3 Propergation Delays of the Control Circuit	33
5.4 Negative Feedback Control System	35
5.5 Flow Chart of the Control Algorithm (Program in pp 79)	38
6.1 Flow Chart of the modified Control Algorithm (Program in pp 84)	50
B.1 Main Circuit Diagram of the a.c. to d.c. converter (Operation Manual of a.c. to d.c. converter, Netherlands, May 1985)	59
C.1 Connection of the Sensor Output to the Analog Multiplexer	61
I.1 Analog Output from the Digital Multiplexer	77

LIST OF TABLES

Table		Page
I	Typical Values Assigned for P, I and D	41
II	Status of the System Before and After the Execution of the Control Algorithm	44
III	Time Taken to Correct the Error Before the Improvement of the Control Algorithm	45
IV	Time Taken to Correct the Error After the Improvement of the Control Algorithm	47

SUPERVISORY CONTROL OF A DOUBLY-FED MACHINE

1. INTRODUCTION

The traditional solution to the problem of maintaining stable operation of interconnected power generation facilities has been to control the speed of all generators to provide synchronous operation. Because of the recent interest in utilizing alternative sources such as wind and wave energy for electrical generation, there is a need to provide generators that operate successfully over a wide speed range [1].

One proposed technique for variable speed generation and for variable speed drives utilizes the doubly-fed machine [2]. To successfully operate and control such a machine in a synchronous power system, new and interesting control problems arise which must be solved.

Computers play an important role as controllers. A controller collects data, performs analyses and executes specified tasks to change or maintain the 'STATE'[1] of the system. Controllers can be grouped into two broad categories: local controllers and supervisory controllers. A supervisory controller decides the 'STATE' of a system based on given criteria. It is the local controller which changes the 'present state' to the 'new state' when

1. Definition: STATE
The STATE of a system is a set of variables which represent past knowledge of system activity in such a manner that knowing the input enables the system to determine the outputs and the next STATE.

instructed to do so by the supervisory controller. There can be one or more local controllers under the supervision of a single supervisory controller. Since computers can be easily programmed to decide the 'STATE' of a system, they are a very good choice for supervisory controllers [2].

This dissertation presents the application of a commercially available process controller to operate a doubly-fed machine. The objective of this research is to find to what extent the EMCON-D process controller, manufactured by EMC (Electronic Modules Corporation) Controls, Inc., can be used for effective control of the doubly-fed machine. The following areas are of concern:

(1) use of a Schwarz converter [3] to control rotor current,

(2) use of the EMCON-D Process Controller for data acquisition and

(3) use of the EMCON-D Process Controller to develop and execute control algorithms.

Chapter 2 describes the EMCON-D Process Controller including the hardware configuration, software tools and overall system capabilities. The doubly-fed machine is described in Chapter 3, along with the control strategy and its relevence. Chapter 4 gives an explanation of transducer selection for data acquisition, connection to the process control unit and the use of software.

Chapter 5 describes the doubly-fed machine control system. The objective of the control is to maintain constant rotor current independent of load, speed and other variations. Hardware components and algorithm development are discussed. A PID control algorithm selected from the EMC software library and a new algorithm developed specifically for supervisory control are also discussed. The results of the application of the control algorithm are presented in Chapter 6. Finally, Chapter 7 presents some conclusions of this research and possible future applications.

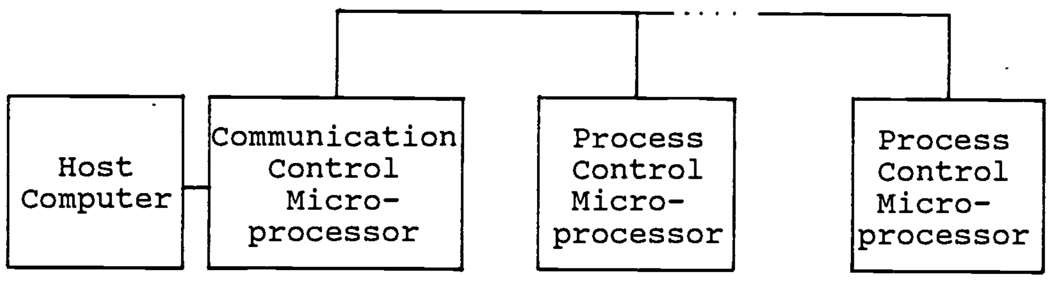
2. PROCESS CONTROLLER

Process control systems are normally used for data acquisition and control purposes. This chapter describes the hardware and software facilities of the EMCON-D Process Controller. Capabilities and limitations are also discussed. Further information can be found in the EMCON-D Hardware and Software Manuals [4-9].

The process controller is formed by a host computer, several microprocessors, industrial quality analog and digital I/O and associated software. Microprocessors are assigned to dedicated tasks such as communication or display control.

2.1 HARDWARE DESCRIPTION

Figure 2.1 shows the general arrangement of the EMCON-D Process Control System. Figure 2.2 shows a block diagram of the host computer. The host communicates with one or more Process Control Microprocessors (PCM) via the Communication Control Microprocessor (CCM). The CCM and PCM(s) are connected by a 56K bit per second data link as shown in Figure 2.2.


Host Computer
Communication Control Micro-processor
Process Control Micro-processor
Process Control Micro-processor

Operator
interaction:
1. System Terminal
2. Printer
3. Color Display

Process
interaction:
1. A/D Converters
2. D/A Converters
3. Pulse Counter Inputs
4. Pulse Train Outputs

Figure 2.1 EMCON-D Process Control System

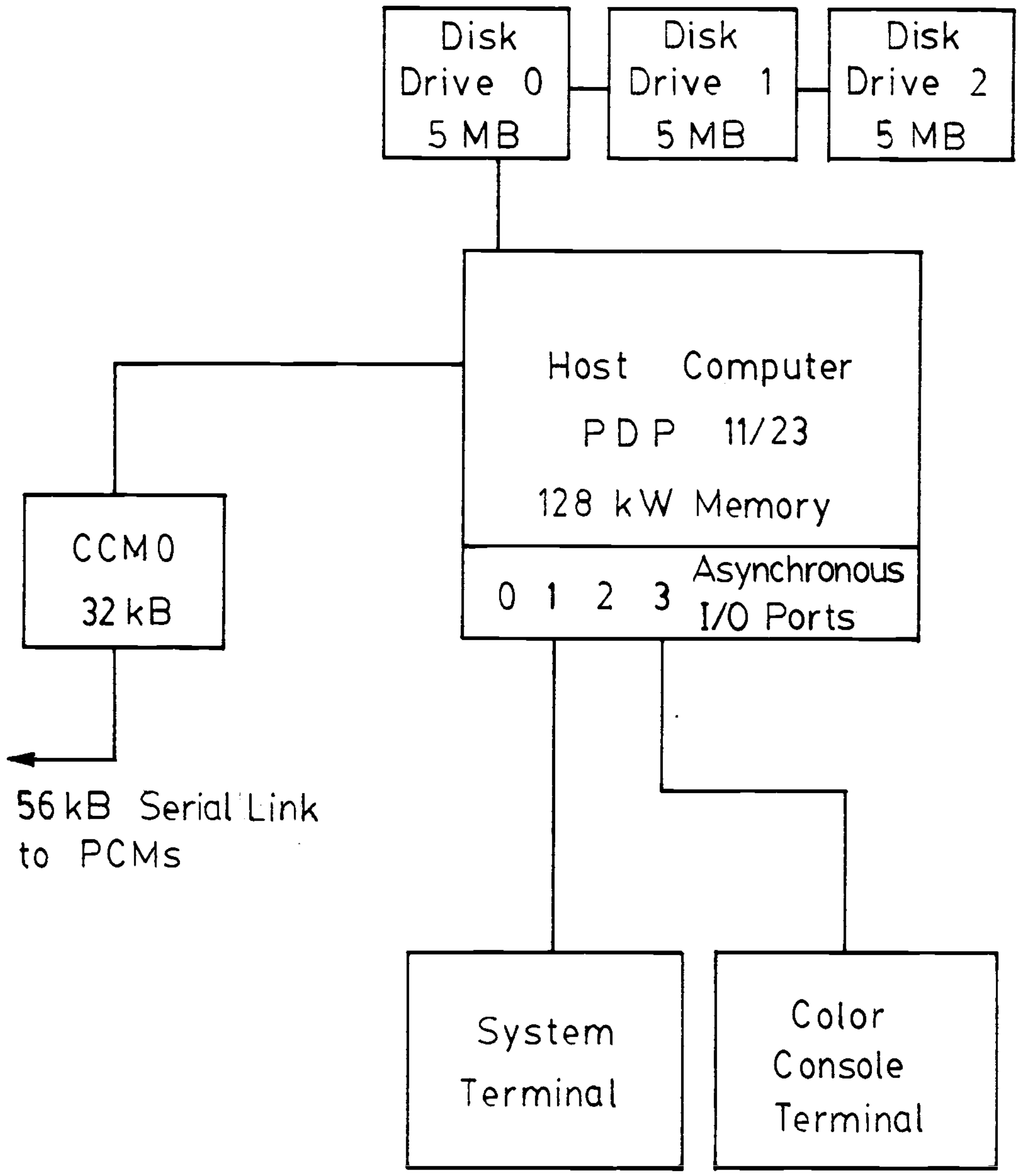


Figure 2.2 Host Computer

The basic hardware of the EMCON-D Process Control System consists of:

1. Host Computer.
2. Communication Control Microprocessor (CCM).
3. Process Control Microprocessor (PCM).
4. Display Control Microprocessor (DCM). (optional)
5. Input/Output Multiplexers.

The host computer is a Digital Equipment Corporation PDP 11/23 minicomputer with 128K-words of memory. The system disk contains the operating system (RSX-11M version 3.2), EMCON-D command files and other system programs. The host computer can control up to 16 PCMs (maximum)[2]. Host functions include database modifications, access to various points in the plant, alarm acknowledgements, initialization, and operator interaction.

The Communication Control Microprocessor is an Intel 8085 with 32K-Bytes of memory. The CCM performs three major functions:

1. Relaying of communication packets between the host and PCMs.
2. System monitoring for reliable operation.
3. Storage of the display portion of certain parameters of the system (as required by the color console). (optional)

2. This is a hardware maximum. The actual number of PCMs that can be installed in the system depends on the speed and capacity of the host computer. The PDP 11/23 can support only four PCMs in a system.

The CCM assists the DCM for color console terminal operation. The CCM provides:

1. The time of day upon demand
2. Access to the CCM database
3. Remote access to the PCM database

The Process Control Microprocessors are also Intel 8085s but with 64K-Bytes of memory. There is only one PCM installed in the system used. The PCM scans the analog, digital and pulse counter inputs and passes selected information to the host via the CCM. It controls the analog, digital and pulse train outputs according to a specified control algorithm provided by the host. The host computer loads the necessary database information to the PCMs during system initialization or restart.

The Display Control Microprocessor (DCM) is used for communication to the host and for control of the color console display. The DCM displays the current process control values on the color console terminal.

The multiplexers are under direct control of the PCM. There are three digital multiplexers and one analog multiplexer. One analog multiplexer can support up to 100 analog input variables. A digital multiplexer holds 16 digital input/output cards. These input/output cards can be a mixture of several types. The PCM scans all specified inputs at a fixed rate. The minimum scan period is one second while the maximum scan period is 255 seconds. The maximum number of analog points that can be scanned in a one second period is restricted to 30 for one PCM.

2.2 SOFTWARE DESCRIPTION

The EMCON-D software falls into three major categories:

1. Host computer operating system.
2. Host application software.
3. Microprocessor application software.

The system software is stored in four disks:

1. System Disk. (RSX-11M operating system, EMCON-D command files and EMCOND-D executable programs)
2. Host Source 1. (EMCON-D host program sources)
3. Host Source 2. (EMCON-D host program sources)
4. Micsysgen. (EMCON-D microprocessor development system and microprocessor program sources)

The host computer software is described in seven manuals [7] and EMCON-D software is described in three manuals [4-6]. The CAM, GROUP, SYSLOD command files, the LOGEN program and the Historical Trending Package are described in Chapter 4. The FTNACCPRG command file is described in Chapter 5.

3. DOUBLY-FED MACHINE CONTROL

The doubly-fed machine and a control strategy are described in this chapter. The use of a doubly-fed machine as a variable speed drive is the main concern. The implementation by installing a current controller either in the stator or the rotor is explained. This is followed by a description of the potentially unstable operation of the machine. Constant Current Mode of operation, which is one method of implementing a stable drive system is described.

A distributed approach can be used to control a doubly-fed machine. A supervisory controller is given the task of specifying the rotor excitation current. A local controller then maintains the rotor current at the value specified. A control circuit is described which is capable of controlling the rotor excitation current within the limitations of the supervisory controller sampling speed.

There are three types of a.c. motors which can be used as variable speed drives: induction motors, synchronous motors and doubly-fed motors. Induction motors draw high reactive power from the grid. To meet this demand, synchronous generators supplying the national grid must have high kVA ratings; otherwise, the generators would be overloaded. This high reactive power demand also has the undesirable result of producing a low power-factor.

Hence, overall system efficency is low [10].

When synchronous motors are used, problems may arise depending on the type of excitation employed. Demagnetization is a common problem in synchronous motors with permanent-magnet excitation [10]. In case of variable excitation, additional costly circuit components are required.

The doubly-fed machine's stator winding as well as the rotor (armature) winding can be fed from the same or different three phase sources. With the rotor completely short-circuited, the doubly-fed machine is an induction machine. Since it is possible to feed the rotor from a different a.c. source (different voltage and frequency), the doubly-fed machine can be made to operate at both 'sub-synchronous' and 'super-synchronous'[3] speeds. Hence, the doubly-fed machine can be considered as a generalization of a synchronous and an induction machine [1].

---

3. The synchronous speed Ns of a machine is given by;

$$Ns = 120 f / P$$

f = frequency of the three phase source
P = number of poles in the machine

For a 4 pole machine connected to a 60Hz source, the synchronous speed is 1800 rpm. If the machine speed is less than the synchronous speed, it is said to be operating at 'sub-synchronous' speed. If the machine speed is greater than the synchronous speed, it is said to be operating at 'super-synchronous' speed.

At the Delft University of Technology in Holland, Schwarz and Lauw initiated a research project for the integration of an ASDTIC (Analog Signal to Discrete Time Interval Conversion) controlled electronic power conversion and the electromechanical energy conversion process [11]. ASDTIC is in essence a signal modulation process, shown to be superior to conventional pulse width and/or frequency modulation techniques. ASDTIC control is important for efficiently controlling the energy flow through a converter incorporating series capacitor resonance circuits. Such a converter is known as a Schwarz Converter [11].

Studies show that the promising potential of the doubly-fed machine may be realized by the use of an ASDTIC controlled converter [2]. The intrinsic characteristics of the doubly-fed machine combined with an ASDTIC controlled converter may be an efficient combination for a.c. variable-speed drives.

3.1 CONSTANT CURRENT MODE OF OPERATION

Figure 3.1 shows a typical variable speed drive configuration for a doubly-fed machine. The ratings of the doubly-fed machine are given in Appendix A. The stator is connected to a busbar with relatively constant voltage and frequency. The rotor is connected to a three phase converter. The current and the frequency of the converter output can be controlled manually or remotely.

Sensors may be installed in the system to provide a number of parameters such as torque, speed, current, power and voltage to the supervisory controller. After manipulating these parameters, the supervisory controller determines the optimum current and frequency of the converter. The converter receives a control signal proportional to the desired values of the rotor current and the frequency. The converter then adjusts and maintains its output until further changes are requested by the supervisory controller. The following sections describe speed control methods and stability problems of the doubly-fed machine.

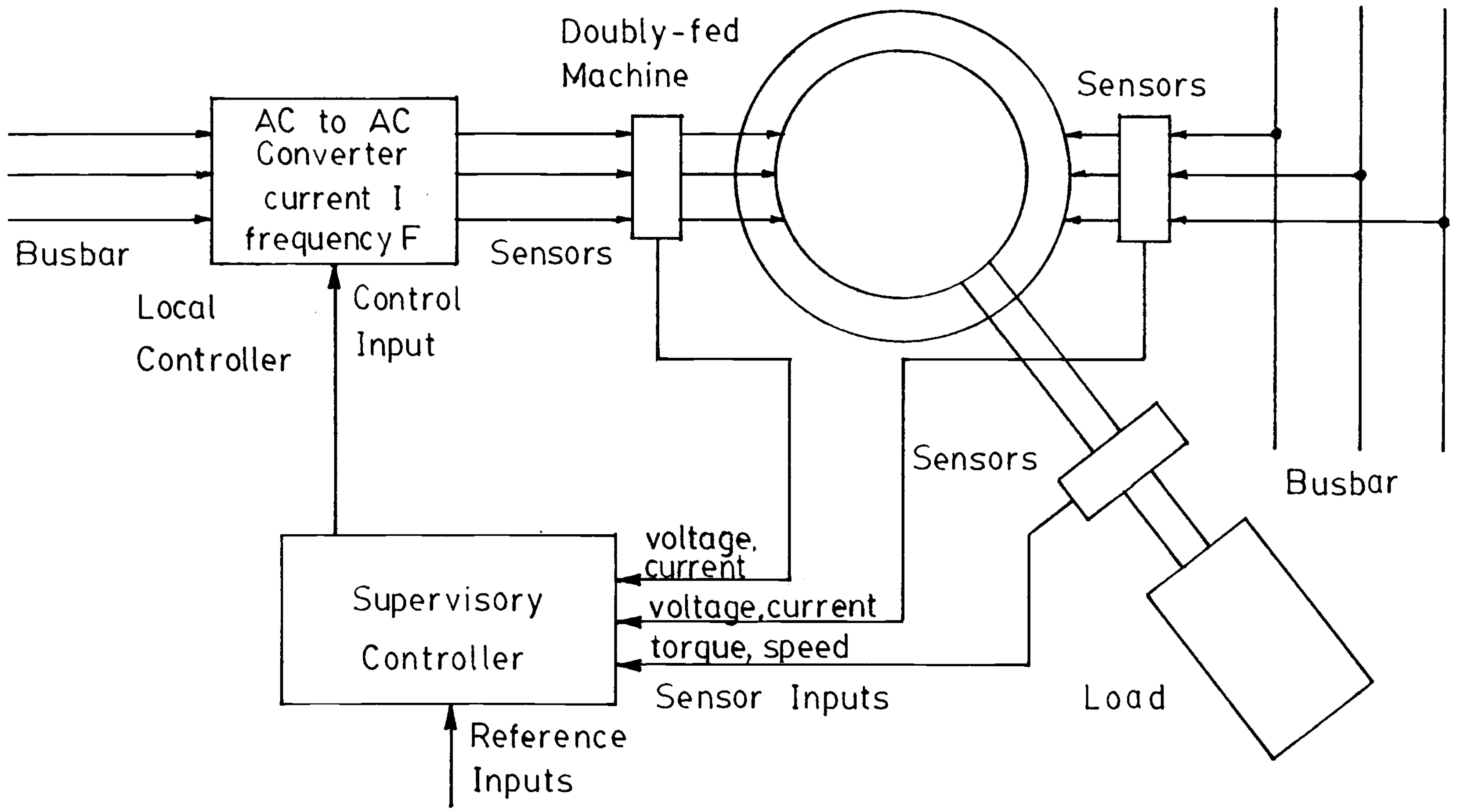


Figure 3.1 Variable Speed Drive Setup for the Doubly-fed Machine

3.1.1 <u>SPEED CONTROL</u>

Both the stator current and the rotor current contribute to the total air-gap flux in the doubly-fed machine. Hence, the speed of the machine is a function of the stator and rotor frequencies. By changing either frequency, the doubly-fed machine can operate as a variable speed drive. By inserting a converter between the stator and the busbar, it is possible to change the frequency applied to the stator. The power rating of the stator converter must be the same as the power rating of the doubly-fed machine. Such a controller would be very expensive.

The more promising approach is to change the rotor frequency as shown in Figure 3.1. For typical loads (fans, blowers) which satisfy a cubic relation with respect to the power - speed characteristics, the power rating of a rotor controller is much less than the stator controller (a ratio on the order of 1:10).

3.1.2 STABILIZATION

It is known that the doubly-fed machine has an unstable region of operation [12]. This region is above, but close to the synchronous speed of the machine. Two methods[4] have been proposed to stabilize the machine:

1. Speed feedback controlling the rotor voltage (voltage source control).
2. Speed feedback controlling the rotor current (current source control).

A major drawback of the first method is that the 'feedback gain' needs to be adjusted with changing load conditions and operating speeds. That is, as shown in Figure 3.2, the feedback gain K is selected for a particular speed N and load. The variable feedback is used to stabilize the doubly-fed machine.

The second method has improved stability properties. Current is more directly related to the flux of the machine than the voltage. Hence, it is much easier to stabilize the machine by controlling the rotor excitation current. Recent developments in power converters further encourage this approach [2].

The speed of the doubly-fed machine is a function of the stator and rotor frequencies [10]. The speed is

4. The two methods are mathematically proven to stabilize a simplified model of a doubly-fed machine. The model requires further work to improve the results for a practical doubly-fed machine.

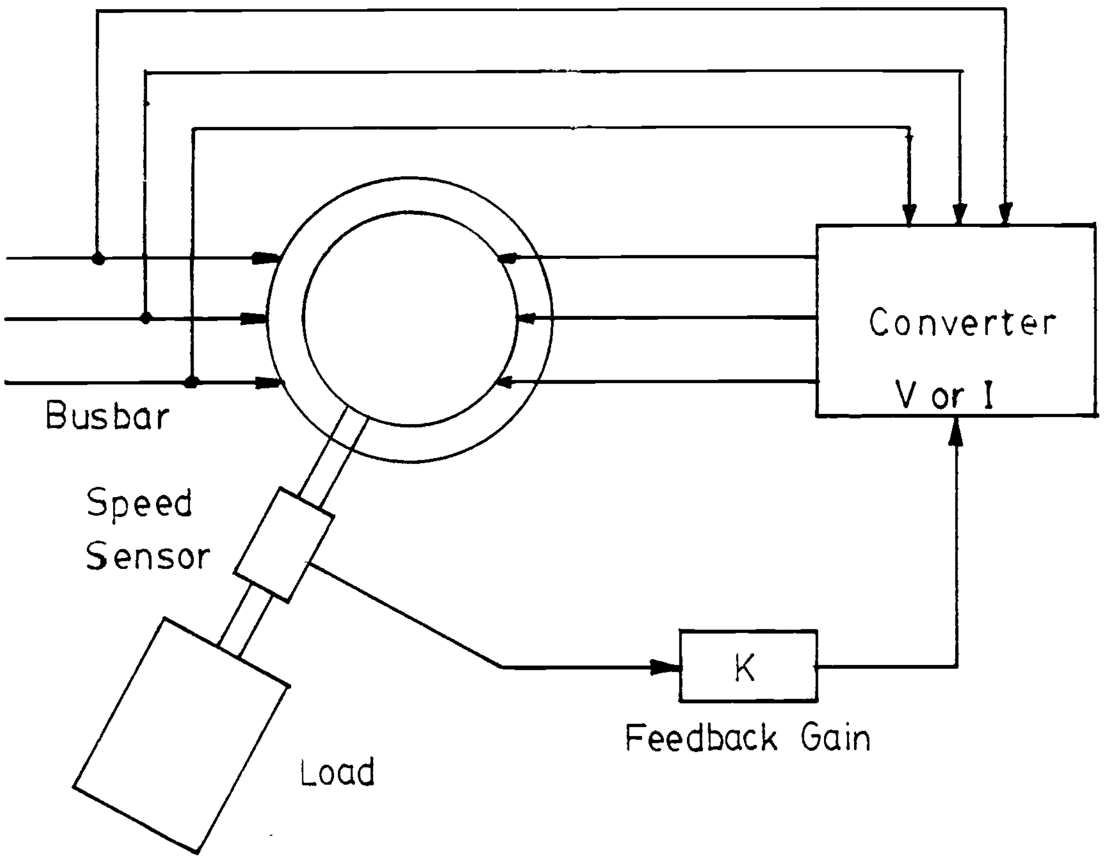


Figure 3.2 Doubly-fed Machine with Converter on Rotor Side

independent of load variations and speed can be controlled without the need for sensing and feedback of this parameter [2]. Hence, the speed feedback loop shown in the Figure 3.2 can be eliminated if the rotor and stator frequencies are known or measured. The supervisory controller shown in Figure 3.1 has to specify the rotor frequency and current which ensures stable operation.

The type of ac to ac converter shown in Figure 3.1 was not available in our laboratory. Instead, a Schwarz ac to dc converter was used (Appendix B). To control the rotor current with this ac to dc converter, the arrangement shown in Figure 3.3 was used. The a.c to d.c. converter, synchronous generator and its prime mover can replace the converter shown in Figure 3.1. The a.c to d.c converter is connected to the field circuit of the synchronous generator. The synchronous generator supplies the rotor excitation current of the doubly-fed machine. The rotor frequency can be changed by varying the speed of the prime mover.

With this configuration, the supervisory controller must control the synchronous generator output current. This mode of operation is called "Constant Current Operation" since the rotor current of the doubly-fed machine is held fixed.

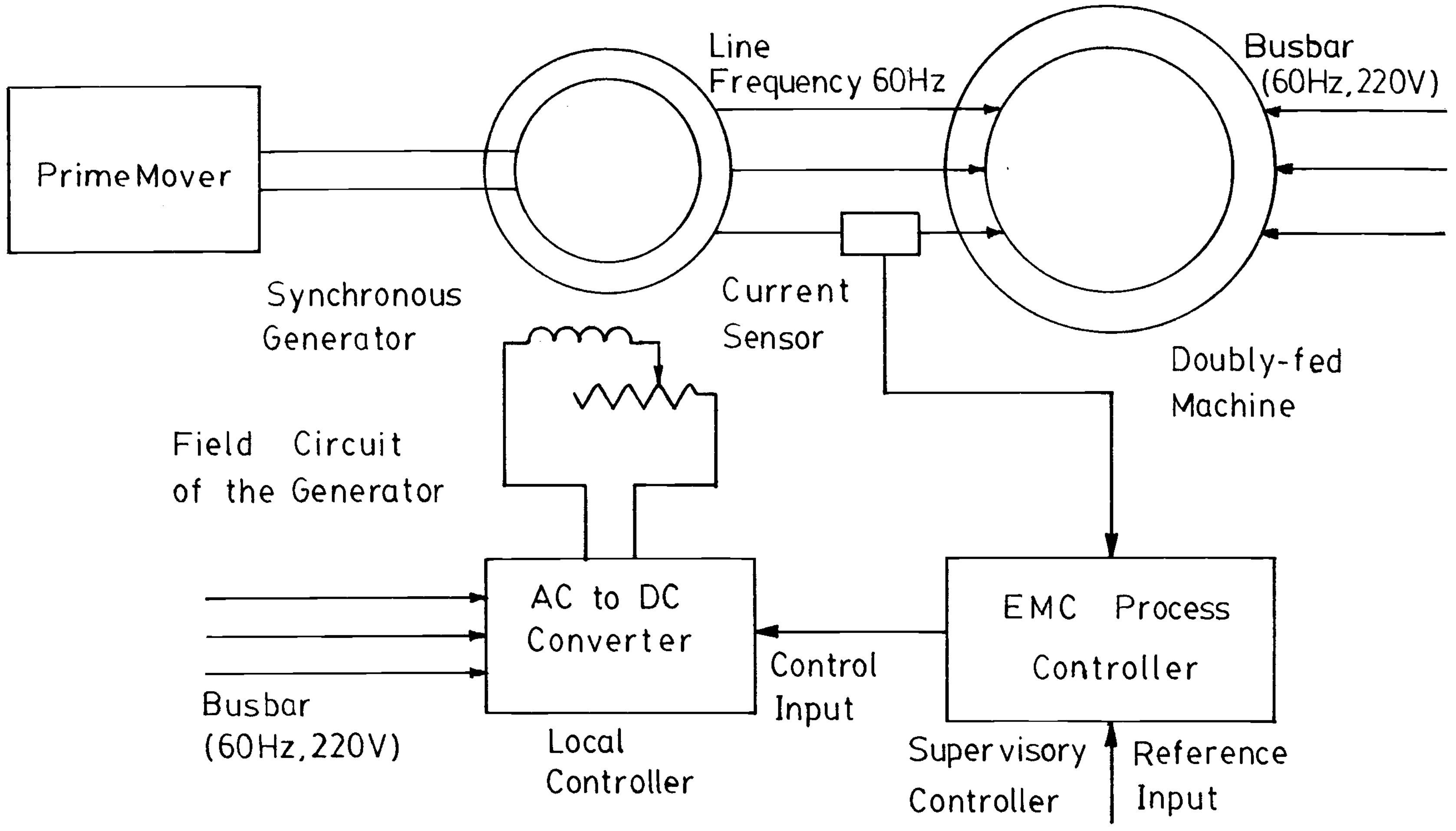


Figure 3.3 Constant Current Operation of the Doubly-fed Machine (for variable speed drives)

3.2 CONTROL CIRCUIT FOR CONSTANT CURRENT OPERATION

The operation of the control circuit shown in Figure 3.3 can be explained as follows (the electrical machine ratings are given in the Appendix A):

The prime-mover is operated at approximately constant speed such that the line frequency is close to 60 Hz. The functions of the supervisory controller in Figure 3.3 are:

1. Specify the reference value of the rotor current of the doubly-fed machine.
2. Control the ac to dc converter such that the rotor current achieves the desired value.

On the assumption that the rotor impedance of the doubly-fed machine is balanced, line current can be measured from any one of the three lines[5]. This value is compared with the reference value specified to the controller. Depending on the sign and magnitude of the 'error' (measured value minus reference value), the controller adjusts the output signal to the converter. This output signal either increases or decreases the field current of the synchronous generator. The change in field current results in a change in the rotor excitation current of the doubly-fed machine.

5. In practice the rotor impedance may not be balanced. A better way of evaluating the rotor excitation current is by considering the rms current of all three phases. A three phase current sensor may be employed.

The new value of the rotor current is measured and compared in the controller with the reference value. This feedback process continues until the rotor current of the doubly-fed machine is within the user defined limits of the reference value. The supervisory control is effective only when the system is in steady state. Hence, the control algorithms includes a 'wait' logic to detect unstable states. Chapter 5 describes algorithm development for the supervisory controller.

4. DATA ACQUISITION

Reliable data acquisition is important in a supervisory controller. This chapter describes the implementation of such a system for the doubly-fed machine. Included are the transducer selection, interfacing with the process controller and use of software to define parameters.

Database modifications and loading of the modified database are discussed. It is the system database that describes the process and control schemes to the process control system. Reference 4, Chapter 4 gives a detailed description of the database. Also included is a description of software that can be used for recording and displaying variables measured by the data acquisition system.

4.1 TRANSDUCER SELECTION AND INTERFACE DESIGN

There are several important considerations in the selection of a transducer. These include:

1. The type of parameter to be measured (temperature, current etc.).
2. The possible range of parameter variation.
3. The frequency of measurement.
4. The type and the magnitude of the required output signal.

The rotor current of the doubly-fed machine shown in Figure 3.3 ranges from 0 to 32.5 amps (rms). The maximum frequency of the fundamental is 120 Hz. A current transducer (Ohio Semitronics Inc., Model CTA113A) was selected to satisfy these requirements. The current transducer output is connected to a free input point of an analog multiplexer [9, Section 2.4.1] on the process controller. Details of the procedure are given in Appendix C.

4.2 DATABASE PREPARATION

Section 4.1 described the hardware involved in interfacing an analog variable to the process controller. Next, this variable must be defined to the software for scanning, monitoring and display purposes. The CAM (Configure And Modify) command file is used to define new analog inputs. Reference 4, Chapter 5 describes the use of CAM command files. An example is given in Appendix D. Further, to display a variable, it must be included in one of the 'groups' (00-99) for the color console terminal. The GROUP command file is used for this purpose. Reference 5, Chapter 14 describes the use of the GROUP command file. An example is given in Appendix E.

The modifications described above are local; i.e. the modifications are stored in the host computer database on disk. To implement these changes, the modified host database must be transferred to the Process Control

Microprocessors (PCM). The SYSLOD command file is used for this purpose. An example of the use of SYSLOD command file is given in Appendix F.

4.3 DATA RECORDING AND DISPLAY

Following the successful execution of the SYSLOD command file, the new analog input variable ARMATURE (also called the EPN or External Point Number) is scanned and its status is displayed on the color console terminal. The LOGEN program and the Historical Trending Package are used for respectively recording and displaying the past values of a variable.

4.3.1 LOGEN PROGRAM

The LOGEN program can be used for printing the values of input variables. The LOGEN program will periodically print the normalized values (in engineering units) of specified variables. The minimum period is one hour and the maximum is 24 hours. The Log Control Language (LCL) is used for the specifications of variables to be printed and their format. The procedure for using the LOGEN program is outlined in the Appendix G. Further details of the LOGEN program are given in Reference 4, Chapter 10. The FORTRAN access method also can be used for similar purposes. Reference 4, Chapter 9 describes the FORTRAN access method.

4.3.2 HISTORICAL TRENDING PACKAGE

One convenient way to collect data and display the 'average values' vs 'time' is in the form of a graph. The Historical Trending Package is used for such purposes. This package records the values of pre-defined variables in a data file (HISREF.DAT[6]). Data older than five days is automatically discarded. The 'trend' key on the color console terminal is used to recall data from the HISREF.DAT file. The procedure for using the Historical Trending Package is similar to the use of the LOGEN program. The description of the desired EPN's are edited into a special file. The Historical Trending Package makes use of this file for the data collection. The procedure is outlined in the Appendix H. Reference 5, Chapter 16 gives more details about this package.

6. The data in HISREF.DAT is stored in binary (unformatted) format. This data can be read from Level 3 FORTRAN database access routines. See Section 5.4 for details.

5. CONTROL CIRCUIT DESIGN AND ALGORITHM DEVELOPMENT

Chapter 3 described the constant current operation of a doubly-fed machine. This chapter describes the hardware implementation of the control circuit. This is followed by a discussion of the requirements for an algorithm to be used in a controller. The PID algorithm selected from the EMCON-D software library is discussed. Finally, a description of an algorithm written in FORTRAN utilizing EMC's FORTRAN database access routines is given.

5.1 CONTROL CIRCUIT HARDWARE

Figure 5.1 shows the hardware configuration which was used to test the ability of the supervisory controller to regulate the rotor current of the synchronous generator. The synchronous generator is connected to a balanced three phase resistive load. In the actual control circuit, the output of the synchronous generator is to be connected to the rotor of the doubly-fed machine. The circuit components are:

1. Current Sensor [Appendix C]
2. AC to DC Converter [Appendix B]
3. Supervisory Controller (EMCON-D Process Controller)
4. Synchronous Generator [ratings in Appendix A]
5. Prime mover

The interface of the AC to DC Converter to the process controller is given in Appendix I.

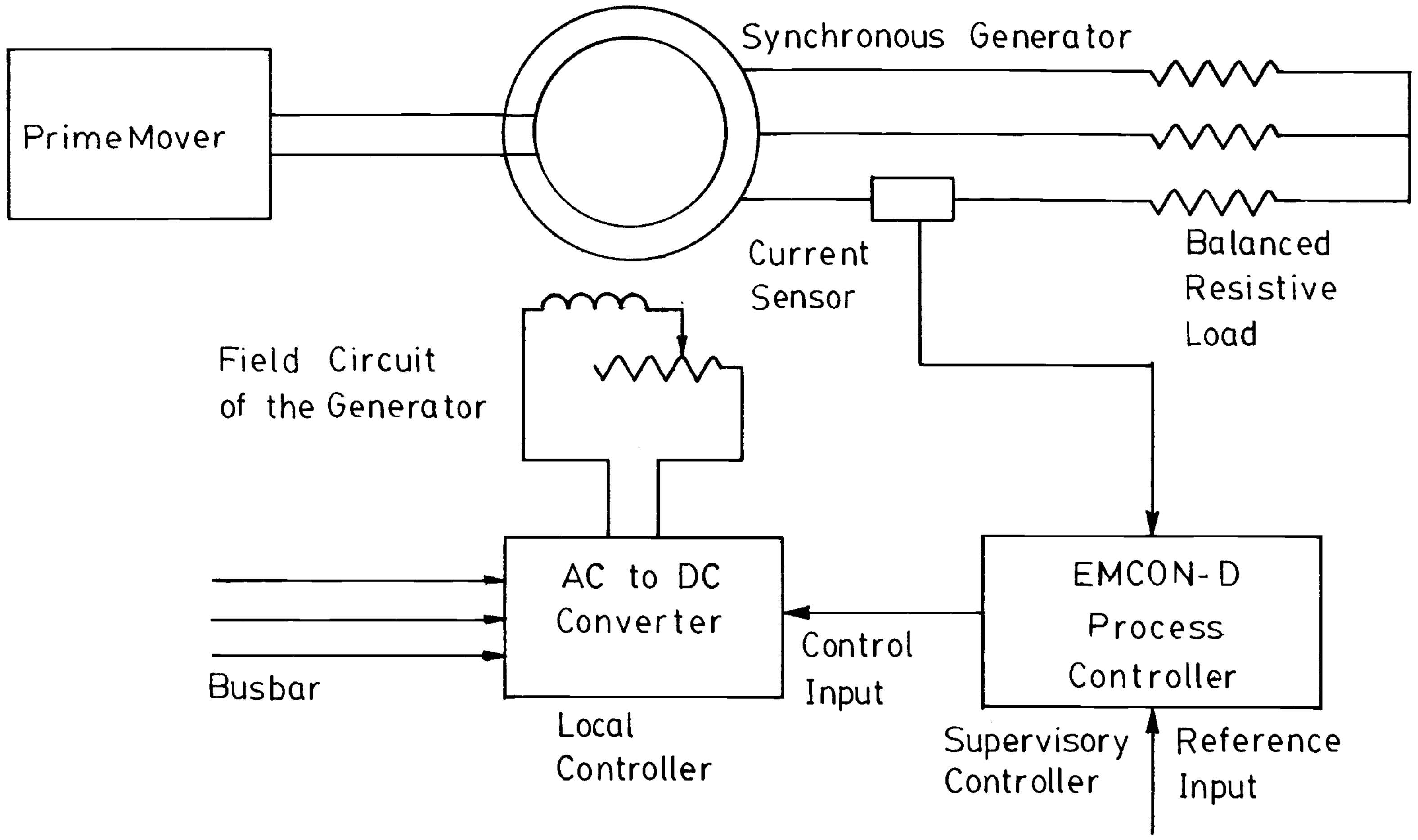


Figure 5.1 Modified Control Circuit for Constant Current Operation

5.2 REQUIREMENTS OF THE CONTROL ALGORITHM

The control algorithm must be capable of regulating the output current of the synchronous generator at a specified value. Such an algorithm can be selected from the EMCON-D software library or can be developed seperately. Any algorithm selected or developed must consider the following basic factors:

1. Measurement of the controlled variable
   The algorithm must be able to measure the most recent value of the controlled variable. In this experiment, it is the output current of the synchronous generator (see Figure 5.1) which must be measured. The PCM scans the controlled variable at a pre-defined rate (see response given to 'PER' of the CAM command file in Appendix D).
2. Specification of the reference value
   One of the important functions of the supervisory controller is to specify the reference values (set-points) of the controlled variables. These values may be determined by start-up, shut-down or steady-state operating requirements.
3. Comparison of the controlled variable and the reference value
   Once the control variable and the set-point values are determined, a comparison can be made to determine the deviation of one from the other. This deviation is

called the 'error'. The control algorithm depends on the 'sign' and the 'magnitude' of the error. Due to errors introduced in a number of stages in the measuring process, it is necessary to specify a tolerance (also called the dead band) for the magnitude of the error; i.e as long as the magnitude of the 'error' falls within the specified tolerance limit, no adjustment to the control variable is required.

4. Changing the control signal
   When the error falls outside the specified tolerance limits, an adjustment is necessary for the control signal to the converter.
5. Magnitude of output changes and number of steps required.
   The algorithm must specify the correction procedure for any deviation of the controlled variable from the set-point. The objective is to bring the controlled variable close to the set-point value in a minimum period of time. The time required to correct the error is a function of the number of steps (number of times the algorithm has to change the control signal before the error falls within tolerance limits) taken by the algorithm. The number of steps required to correct the error is a function of the step size (magnitude of the change in the output control signal).

In general, if the error is too large, the algorithm must begin adjusting the control output in large step sizes. As the error becomes smaller, the step size of the control output should be reduced (Figure 5.2) leading to the idea of proportional control.

6. Time of response

The time of response[7] is the time between two consecutive changes in the control variable. The time 't' shown in Figure 5.2 is the response time. This time is affected by several factors:

a. The delay associated with the circuit before the control output from the digital multiplexer arrives at the remote terminals of the converter. (T1)
b. The internal delay of the converter before changing the output excitation current to the new value. (T2)
c. The internal delay of the synchronous machine before adjusting the rotor output current to the new value. (T3)
d. The delay associated with the current transducer before providing the new value of rotor excitation current to the analog multiplexer of the EMCON-D Process Controller. (T4)

7. This definition for the response time is valid only if the control variable is free from overshoot. Figure 5.2 does not show any oscillations in the control variable. The response time must be redefined to incorporate these variations.

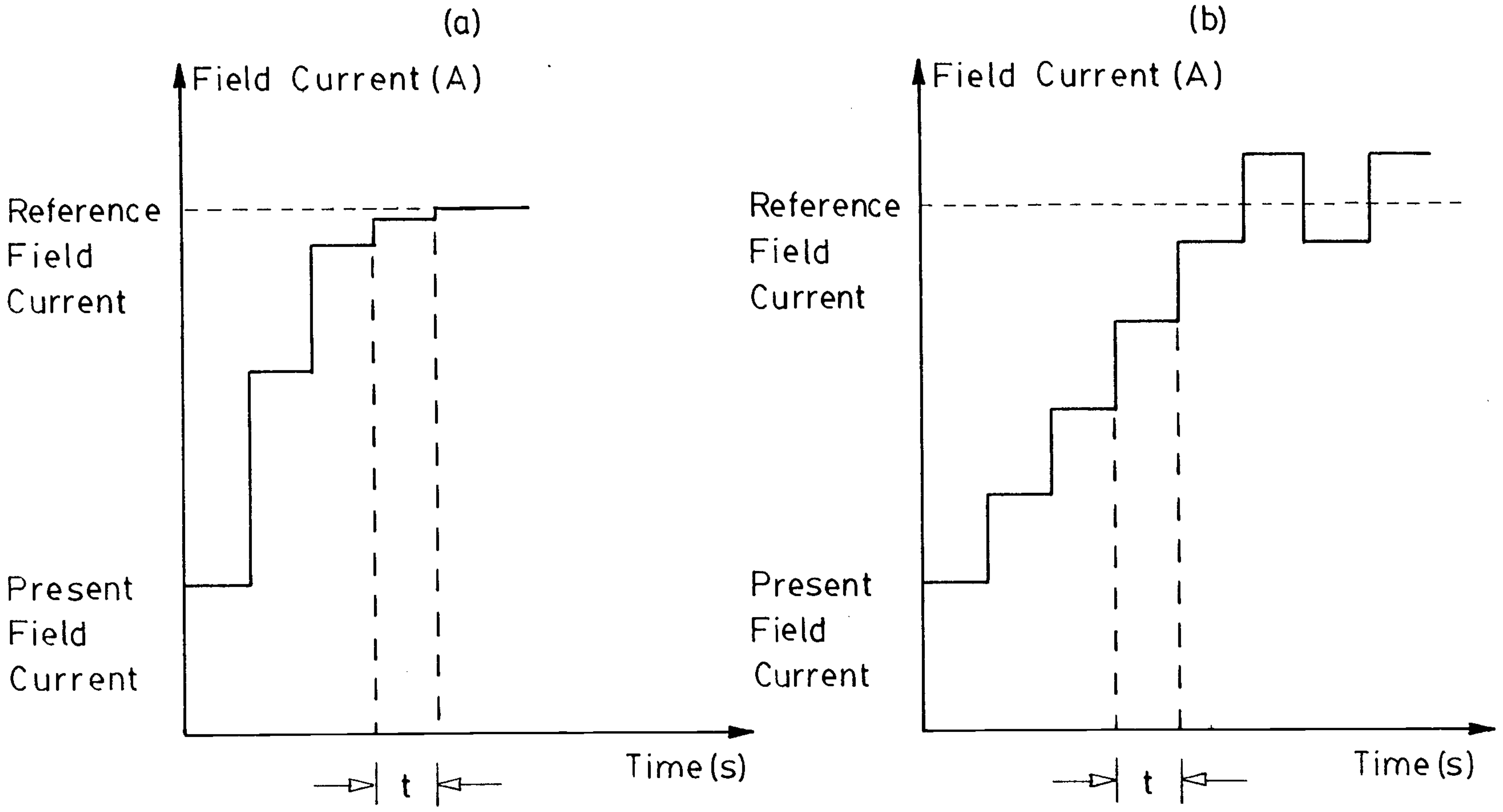


Figure 5.2 Non-uniform Step sizes vs Uniform Step Sizes for the Field Current of the Generator
(a) Non-uniform Steps
(b) Uniform Steps

e. The internal delay of the EMCON-D process controller before adjusting the control output. (T5)

Figure 5.3 illustrates the delay times in each part of the control circuit. The response time 't' is the sum of the delays T1 through T5. The algorithm must allow a sufficient period of time between two consecutive adjustments of the control variable. This time period must be greater than the response time 't'.

7. Transient Behavior:

Requirement 1 through 6 assume steady state operation. There is a possibility that the control variable undergoes overshoot or even oscillates during adjustment periods. No adjustments must be made during these periods. The algorithm must detect such situations and should wait until the system becomes steady. Note that the response time 't' is affected for an unknown period of time because of this behavior. An unstable situation can be detected by the comparison of at least three consecutive samples of the controlled variable. This concept is further discussed in Section 6.3.

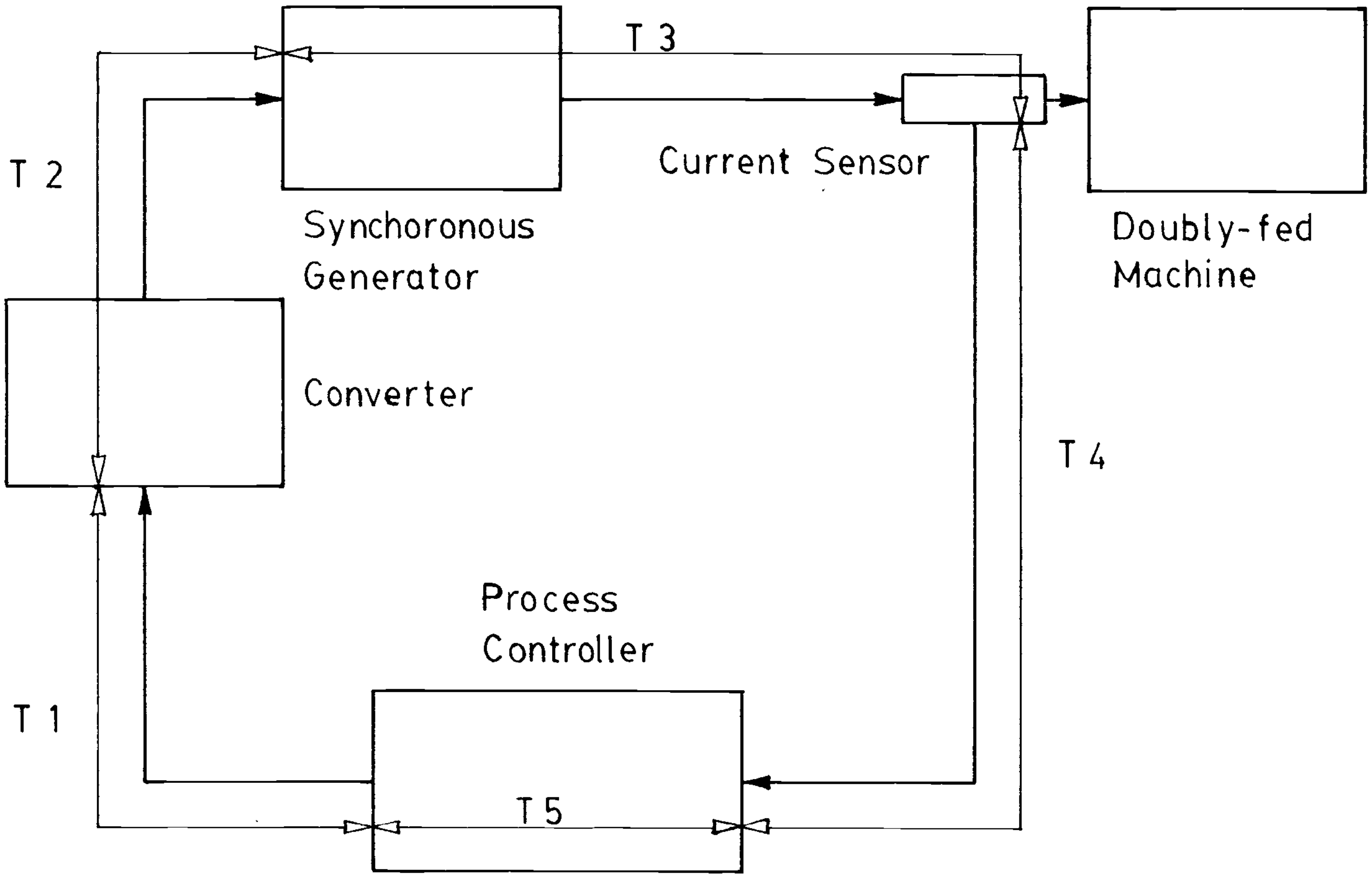


Figure 5.3 Propergation Delays of the Control Circuit

The rest of this chapter is devoted to the selection and development of algorithms to satisfy requirements 1 through 6. Two algorithms are discussed in the next sections; the PID algorithm selected from the EMCON-D software library and a uniform step algorithm developed using the FORTRAN access method.

5.3 PID ALGORITHM

The EMCON-D software library provides [4, Chapter 5] a number of control algorithms. These include:

1. PID - A three term positional algorithm based on change of error, total error and cumulative error.
2. PIN - A two term nonlinear position algorithm based on change in error and total error with a variable gain factor.
3. RAMP- A set-point ramp algorithm.

Use of the above algorithms does not require any programming by the user.

Figure 5.4 shows a typical negative feedback system. For many systems, the output 'y' can be related to the error 'e' according to the following PID equation.

$$y = P\,e + I\int e\,dt + D\,\frac{de}{dt} \qquad (1)$$

where, P = Proportional constant

I = Integral constant

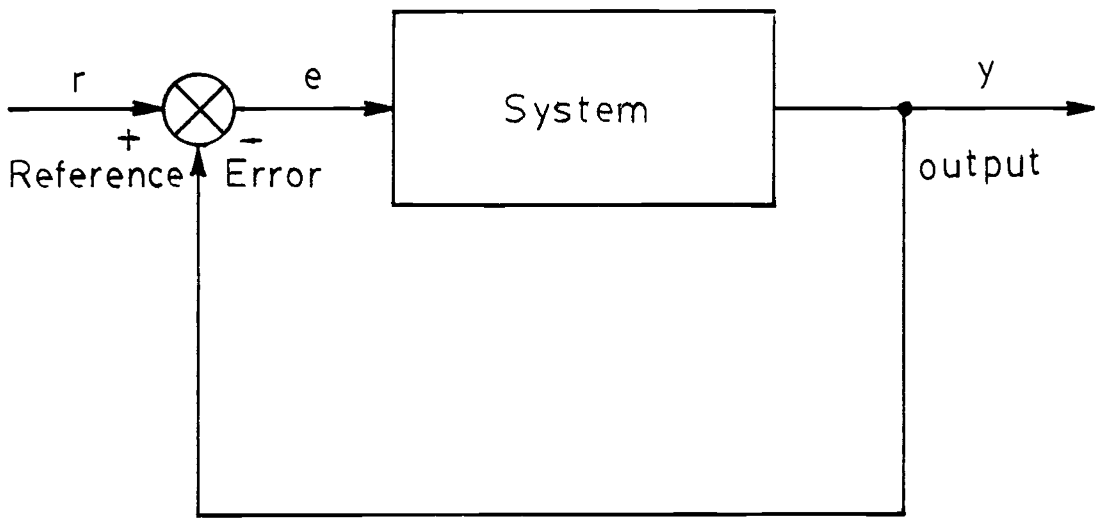


Figure 5.4 Negative Feedback Control System

D = Derivative constant

$e = r - y$

r = Reference (set-point)

Most of the process control applications utilize this PID feedback control function [4, Chapter 5]. Many variations are possible by specifying different values for the tuning constants P, I and D. Hence, it is reasonable that by selecting suitable values for P, I and D, a control algorithm can be developed for the doubly-fed machine controller.

5.4 ALGORITHM DEVELOPMENT USING FORTRAN ACCESS

FORTRAN access is an additional feature of the EMCON-D Process Controller. It allows development of algorithms in FORTRAN IV or MACRO 11 languages. The FORTRAN access library consists of a number of database access routines. These routines either access or modify parameters maintained in the system database. The database access routines [4, Chapter 9] are divided into three sections: Level 1, Level 2 and Level 3.

Level 1 routines directly interface to examine and modify the parameters in the database. The user need not be concerned about the actual database configuration or format conversions. On the other hand, Level 2 calls require a careful study of the database arrangement and conversion routines. Effectively, Level 1 calls use Level 2 calls to perform their functions. Level 3 calls fetch data values collected by the Historical Trending Package (i.e. Level 3 routines read the data values stored in the HISREF.DAT file. See Sec 4.3.2).

Figure 5.5 shows the flow chart of a simplistic algorithm to operate the doubly-fed machine in constant current mode which meets the requirements defined in Section 5.2. The first block in the flow chart initializes variables and constants used by the database access routines. The second block uses several database access routines to fetch the values of the output current of the synchronous generator, reference value and the output (as a percentage) of the control block 'ARMATURE'. These routines return the normalized values of parameters to the program if the operation is successful. Otherwise, the routines return an error value which can be used to locate the error. The error term of the routines is optional. At this point, the algorithm branches into three different paths depending on whether the current is greater than, equal to or less than the reference value. The difference between the current and the reference is called the 'error'.

If there is no error, the present output from the control block is unchanged and the algorithm ends. If the error is positive, the output is increased by 1% and if error is negative, the output is decreased by 1%. The program terminates after these changes. This 1% output change may or may not correct the error. Hence, the

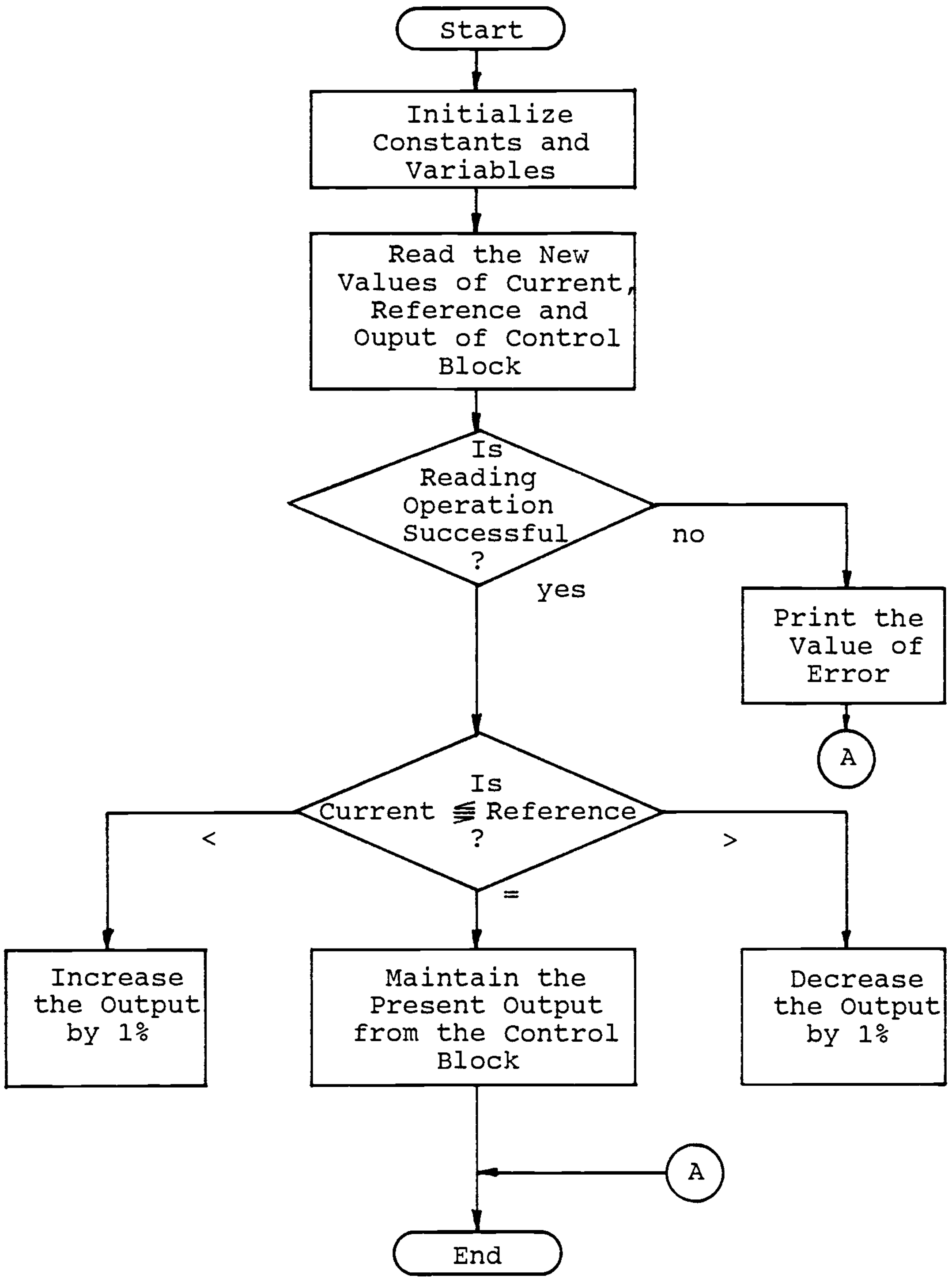

Start
Initialize Constants and Variables
Read the New Values of Current, Reference and Ouput of Control Block
Is Reading Operation Successful ?
no
yes
Print the Value of Error
A
Is Current ⋚ Reference ?
<
>
=
Increase the Output by 1%
Maintain the Present Output from the Control Block
Decrease the Output by 1%
A
End

Figure 5.5 Flow Chart of the Control Algorithm (Program in pp 79)

program must be scheduled to run on a regular basis. The scheduling interval must be greater than the total delay period of the control circuit (see Time of response in Sec 5.2). Since a uniform step size is used in the algorithm, it will take some time to correct if the magnitude of the error is large. Also note that there are no provisions to detect transient behavior of the controlled variable. Section 6.3 describes a modified algorithm which is capable of detecting oscillations of the controlled variable.

A typical program written in FORTRAN IV according to the flow chart of Figure 5.5 is given in Appendix J. The FTNACCPRG command file is used to compile and generate the executable code of this FORTRAN program.

The FTNACCPRG command file is initiated from the system terminal as follows:

```
>@[200,200]FTNACCPRG <CR>
```

Appendix K shows the result of a successful task-building operation using the FTNACCPRG command file.

6. CONSTANT CURRENT OPERATION

The execution of algorithms to operate the doubly-fed machine in constant current mode (Figure 5.1) are described in this chapter. A description of simple adjustments and modifications to the algorithms presented in Chapter 5 are given. The algorithm developed using the FORTRAN access routines is modified to tolerate transient behavior. It is further improved for faster convergence using non-uniform step sizes. Experimental results are given and comparisons are made in the algorithm development procedure.

6.1 PID ALGORITHM

The status of the control block 'ARMATURE' [Appendix D] can be changed from manual to auto by use of the 'auto' key on the color console terminal. In the auto mode, the target state must be entered via the color console. In this mode the PID algorithm was used to control the system.

It was expected that the output of the control block would increase until the output current of the synchronous generator approximated the reference value. However, the output of the control block remained at 0% (as set at the beginning of the experiment).

The control block was set to the manual position. The output was increased (using the 'output' key of the console terminal) until the armature current was almost equal to the reference value. Next, the control block was set to the auto position. The controller should maintain the present output (or a value very close to it) since the reference value and the rotor current are almost the same. However, the output of the control block began to oscillate and corresponding oscillations were observed in the output current of the synchronous generator. These oscillations died out after some time when the output of the control block went to 0%.

The constants associated with the control block were changed as given in Table I from the color console terminal.

TABLE I: <u>Typical Values Assigned for P, I and D</u>

PROPORTIONAL (P)	INTEGRAL (I)	DERIVATIVE (D)
100%	10	0
100%	10	20
80%	40	5
80%	50	10
100%	0	0

Reference 4, Chapter 5, Section 2.13 describes the value allocation and tuning procedure for the PID algorithm for a particular application. The table shows the final values for each constant when the experiment was unsuccessfully terminated. The combination I=D=0 and P=100 makes the PID algorithm a proportional control algorithm.

For every combination given above, the performance of the controller was the same. The control block output either remained at 0% or it approached 0% with oscillations when set to a non-zero value at the beginning.

The PCM scan rate for control block 'ARMATURE' is once a second. The scan rate was reduced to compensate for the loop delay (see Section 5.2, time of response). The scan rate was set to once every five seconds. The algorithm was executed and, once again, similar behavior occured in the output of the control block.

The reason for the failure of the PID algorithm was a software fault. The manufacturer confirmed a software fault in one of the database access routines which would result in the behavior of the system explained above.

6.2 FORTRAN ACCESS

For this mode of operation, the control block 'ARMATURE' must be in the manual position. The algorithm developed in Section 5.4 was scheduled to run once every five seconds.

The task file is installed from the system terminal as follows:

```
> INS [300,54] MSC.TSK <CR>
```

The program is executed with:

```
> RUN MSC /RSI = 5 S <CR>
```

The system now runs the algorithm once every five seconds. RSI is called the rescheduling time interval.

It was expected that the output of the control block would increase in steps of 1% for every execution of the task, provided the reference value is greater than the rotor excitation current. When the output current of the synchronous generator approaches the reference value, the output should oscillate within ± 1% such that the output current of the synchronous generator is around the reference value. Unfortunately, the output of the control block remained at zero.

The algorithm was terminated as follows:

```
> CANCEL MSC
```

The output of the control block was manually increased until the output current of the synchronous generator was close to the reference value. The algorithm was started again. The algorithm should maintain the present output from the control block. Unfortunately, the output of the control block began to oscillate and unlike in the PID algorithm, these oscillations did not die out but continued to grow. At one stage, the oscillations reached 100% output from the control block.

A software fault was detected in the EMCON-D FORTRAN database access routine 'PUTOUT()' (later confirmed by the manufacturer). This fault results in approximately a 20%

offset error in the output of the control block. This was corrected in the algorithm by subtracting 20% from the present output value before calling 'PUTOUT()'. Also, the output of the control block was raised manually to a value above 20% before initiating the algorithm.

The desired results were obtained this time from the algorithm. Table II shows the status of the system before and after the execution of the algorithm.

TABLE II: Status of the System Before and After the Execution of the Control Algorithm

	Reference	output current	Output percentage
Before	0.300	0.250	30
After	0.300	0.290-0.310	38-40

Note that the output current lies between 0.290 A and 0.310 A. These oscillations are a result of the algorithm attempting to adjust the output of the control block so that the output current is the same as the reference value. Since a constant step size (1%) without tolerence limit is selected for the output of the control block, it is not possible to adjust the output current of the synchronous generator to be exactly equal to the reference value. The oscillations can be eliminated by specifying a tolerance limit for the 'error' in the algorithm. The program which has been corrected for the offset error of the 'PUTOUT ()' routine and modified to eliminate oscillations by specifying a tolerance band for 'error' is in Appendix L.

6.3 IMPROVEMENT OF THE FORTRAN ACCESS ALGORITHM

Table III shows the time taken by the algorithm to correct the error for different values of reference and output currents of the synchronous generator. The rescheduling time interval (RSI) is 2 seconds.

TABLE III : Time Taken to Correct the Error Before the Improvement of the Control Algorithm

INITIAL VALUE A	REFERENCE VALUE A	ERROR %	TIME TAKEN TO CORRECT THE ERROR	FINAL VALUE A
0.075	0.25	-17.5	62 sec	0.255
0.247	0.35	-10.3	28 sec	0.354
0.351	0.15	20.1	66 sec	0.146
0.146	0.10	4.8	26 sec	0.099

Non-uniform step sizes (proportional control) can be used for faster convergence. If the error is large, the corresponding change in the step size must be large and if it is small then the step size must be small. On the assumption of a linear relationship between the step size of the control block output and the magnitude of error (as a percentage), the following equations can be developed. Similar results may be expected from the PID algorithm when both I and D constants are set to zero.

$$\text{percentage error} = \frac{(\text{initial value} - \text{reference value})\times 100}{\text{full load current}},$$

$$\text{step size} \propto \text{percentage error},$$

$$\text{step size} = \frac{K(\text{reference value} - \text{present value})\times 100}{\text{full load current}},$$

where the full load current is the upper bound[8] specified in engineering units for the output current of the synchronous generator when the control block 'ARAMATURE' is defined using CAM. The output of the control block is calculated in the algorithm from the following equation:

$$\text{output} = \text{output} \pm \text{step size},$$

For simplicity, K was set to 1. Table IV shows the results obtained after improving the algorithm using proportional step sizes for the output of the control block. The rescheduling time interval (RSI) is 2 seconds (large changes in the control variable may result in oscillations delaying the correction procedure). The response time of the system is greater than 1 second even without oscillations in the controlled variables. Hence, a 2 second schedule time was selected.

8. Note that the default for the high alarm limit is this upper bound. See the response to 'EGU' in the CAM command file in Appendix D. The variables 'MAX' in the program in Appendix K correspond to the high alarm limit.

TABLE IV : Time Taken to Correct the Error After the Improvement of the Control Algorithm

INITIAL VALUE A	REFERENCE VALUE A	ERROR %	TIME TAKEN TO CORRECT THE ERROR	FINAL VALUE A
0.074	0.25	-17.5	12 sec	0.248
0.248	0.35	-10.2	8 sec	0.354
0.354	0.15	20.4	16 sec	0.144
0.144	0.10	4.4	10 sec	0.099

The algorithm must detect oscillations of the output current of the synchronous generator. No adjustments should be made to the output of the control block until these transients settle. The supervisory controller is effective only when the system is in steady state.

Such conditions can only be detected by making a comparison of at least three or more samples of the controlled variable taken at two different time intervals. The second sample is taken t1 seconds after first sample. The third sample is taken t2 seconds after second sample. Note that t1 and t2 must be greater than 1 sec if the scan rate of the measured variable is 1 sec. If the measured variable is sustaining oscillations with fixed frequency, then the three samples will not be the same; at least one will differ from the other two. If all three samples are within limits, it is possible to consider the measured variable to be in steady state.

This scheme fails to detect oscillations in the following case: Let the second sample taken after t1 sec be the same as the first sample. If t1 + t2 = T, where T is the period of oscillation then the third sample will be the same as first two samples. The above scheme will give satisfactory results if the frequency of oscillation is greater than 1 Hz.

Two methods are possible to accomplish this variable sampling interval. Only two samples are considered in the following methods. These can be easily extended to compare three samples. The methods are:

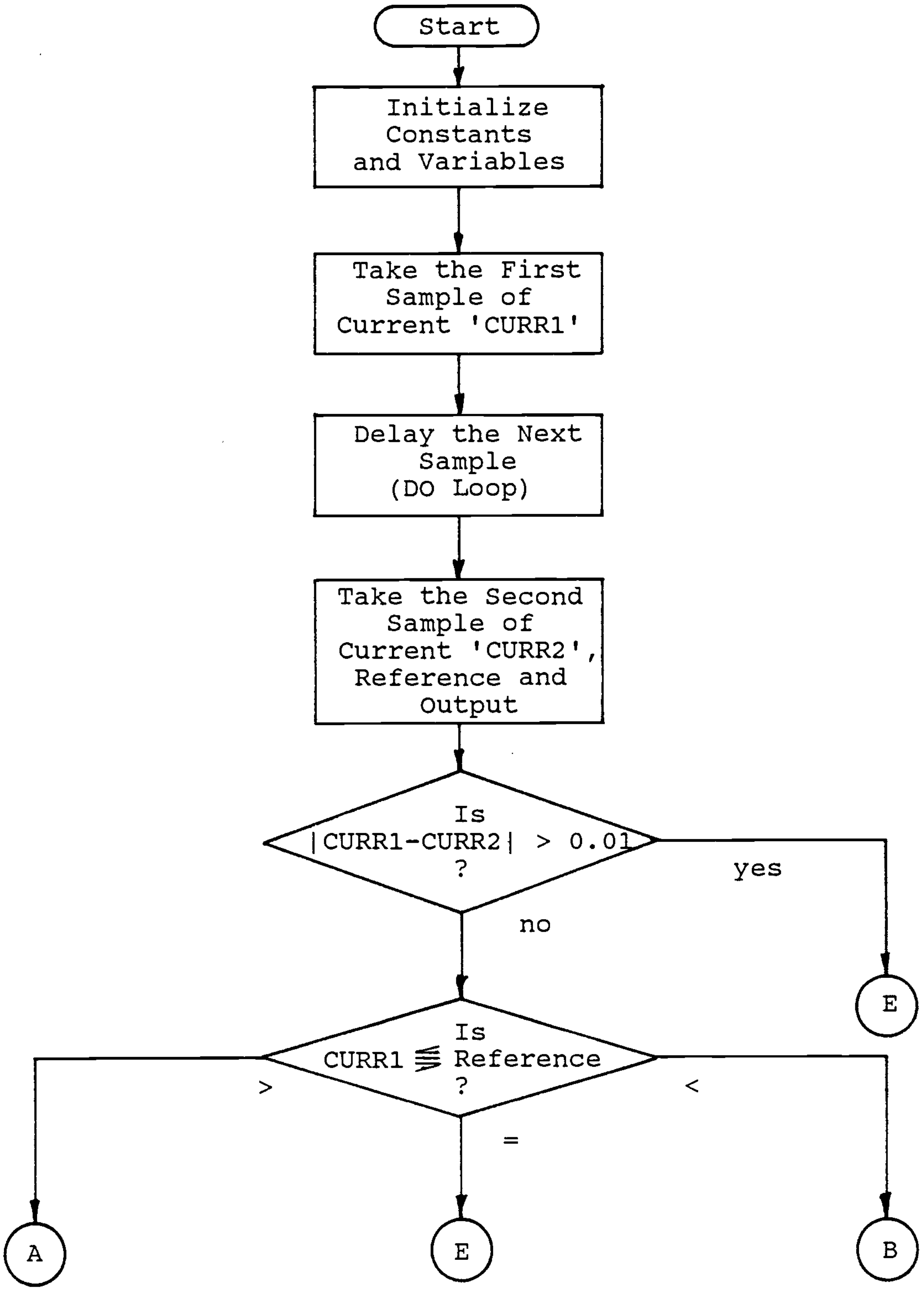
1. Delay loop in the algorithm

   After the first sample is taken the algorithm delays for some time (> 1 second) before taking the second sample. If the first and second samples are similar, then the system is in steady state. Otherwise, the algorithm has to delay further before taking another sample.

2. Use of host disk drives

   The algorithm records the first sample of the output current in a file in the system disk. When the algorithm is called next, the previous value is recalled from the file and compared with the second sample. If the values are similar, the algorithm continues. Otherwise, the second sample is recorded in the file discarding the first sample for comparison with the next sample. This scheme works on the assumption that the host I/O access takes at least 1 second.

The modified flow chart of the algorithm is shown in Figure 6.1. The corresponding FORTRAN source code is in Appendix L.



Start
Initialize Constants and Variables
Take the First Sample of Current 'CURR1'
Delay the Next Sample (DO Loop)
Take the Second Sample of Current 'CURR2', Reference and Output
Is |CURR1-CURR2| > 0.01 ?
yes
no
E
Is CURR1 ⋛ Reference ?
>
<
=
A
E
B

Figure 6.1 Flow Chart of the Modified Control Algorithm (Program in pp 84) Contd..

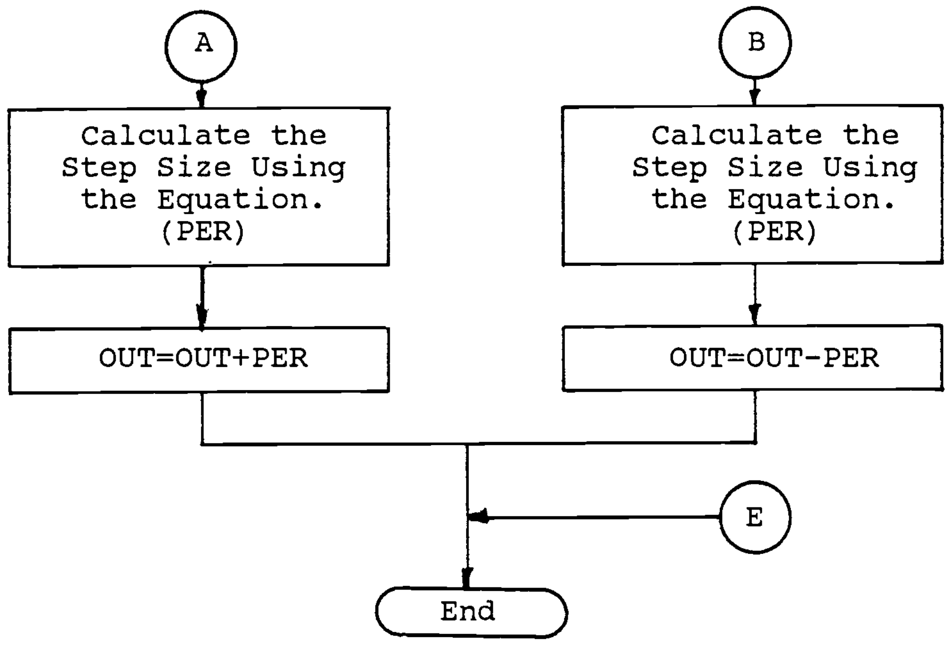

A
B
Calculate the Step Size Using the Equation. (PER)
Calculate the Step Size Using the Equation. (PER)
OUT=OUT+PER
OUT=OUT-PER
E
End

Figure 6.1 Flow Chart of the Modified Control Algorithm (Program in pp 84)Continued

7. SUMMARY AND CONCLUSIONS

This chapter presents conclusions and possible future applications of the EMCON-D Process Controller as a supervisory controller.

7.1 CONCLUSIONS

The EMCON-D Process Controller is suitable, with limitations, for use in the control of a doubly-fed machine. Due to the slow sampling rate, it can only be employed as a higher level controller when controlling a fast system such as a doubly-fed machine.

With suitable sensors, a data acquisition system can be developed. Variables such as current, voltage, torque, speed, input power and power-factor can be measured. The maximum rate at which data can be collected from each variable is once a second. Hence, this data acquisition system is not suitable for the collection of data to study transient behavior of the doubly-fed machine.

Dual slope type A/D converters are used in the analog multiplexer. The number of analog variables that can be allocated to the process controller with maximum scan rate (once a second) is 30 for each PCM station (more analog variables can be allocated at a slower rate). The 30 sample per second rate is chosen so that 60 Hz noise is considerably reduced in the A/D conversion process. The process controller software can be used to display these variables graphically on the color console terminal and also to produce listings on the printer.

Lower level devices such as an ac to dc converter or relays can be successfully interfaced with the process controller. When the system is used as a higher level controller, the FORTRAN access method is highly recommended for control algorithm development. The FORTRAN access method is better for such applications for two reasons:

1. Reference Value:

   The higher level controller decides the target state of the system. After deciding the target state, the controller has to provide one or more reference signals to the local controllers to steer the system to the desired state. The FORTRAN access method provides greater flexibility and a higher degree of freedom in the process of evaluating these reference values.

2. Implementation of 'wait' logic:

   Time is not a critical factor for a higher level controller. Hence, the target state is decided under steady state conditions. The higher level controller has to wait when the system undergoes transients. Two successful methods were investigated to implement the wait logic in the control algorithm.

   Implementation of the wait logic with a delay loop in software is not very efficient since it wastes the CPU time of the host computer. The use of host disk drives is a better way of implementing a wait logic since it does not waste CPU time. This method increases disk activity which may slow other programs and may shorten the life of the equipment.

The algorithms can be further improved when they are developed using the FORTRAN access method. The execution time of the control algorithm heavily depends upon the speed and the capacity of the host computer. The performance can certainly be improved by using a PDP 11/44 instead of a PDP 11/23 as the host computer.

The algorithms developed using the FORTRAN access method successfully controlled the rotor current in step as well as in proportional control although it is a lower level function. The PID algorithm selected from the EMC software library failed due to a software fault.

A software fault was detected in the EMCON-D database access routine 'PUTOUT' and was confirmed by the manufacturer. This software fault resulted in an off-set error of approximately 20% in the control signal provided to the ac to dc converter from the process controller. In the FORTRAN access method, this 20% error was subtracted from the output signal. Similarly, the internal PID algorithm uses these database access routines. Hence, it is very reasonable to assume that a software fault of one or more database access routines led to the failure of the PID algorithm.

PID algorithms in this process controller have the following limitations:

1. Values of P, I and D are limited to:

   P - 0 to 819.1 %

   I - 0 to 255 for one second scan rate

   D - 0 to 255 for one second scan rate
2. Constraints on operating time intervals (1 to 255 seconds maximum).
3. The execution of the control algorithm depends on the speed and the capacity of the PCM. This severely limits its applications for fairly simple control schemes.

7.2 FUTURE APPLICATIONS

The doubly-fed machine can be used for variable speed generation or as a variable speed drive [1,2]. Typical supervisory control functions that may be assigned are:

1. Start-up of the doubly-fed machine.
2. Shut-down of the doubly-fed machine.
3. Supervision of local controllers which are assigned to various tasks such as:
   a. maintain the rotor excitation current.
   b. control the power-factor.
   c. control the speed etc.

BIBLIOGRAPHY

1. Lauw H.K.,"Variable-Speed Generation with the Doubly-fed machine", Prepared for EPRI/DOE Workshop, Denver, May 22, 1983.

2. Lauw H.K.,"Variable Speed Drives with the Doubly Fed Motor", Oregon State University, Dept. of Electrical and Computer Engineering.

3. "Operation Manual of D.C. to A.C. Converter" donated from Delft University of Technology, Netherlands. May 1985.

4. EMC Controls, Incorporated, "EMCON-DR/DS Product Software Documentation Volume 1, V8ADRTO-Version V8A000", P.O.Box 242, Cockeysville, Md 21030.

5. EMC Controls, Incorporated, "EMCON-DR/DS Product Software Documentation Volume 2, V8ADRTO-Version V8A000", P.O.Box 242, Cockeysville, Md 21030.

6. EMC Controls, Incorporated, "EMCON-DR/DS Product Software Documentation Volume 3, V8ADRTO-Version V8A000", P.O.Box 242, Cockeysville, Md 21030.

7. EMC Controls, Incorporated, "RSX-11M Version 3.2 Software Documentation, Digital Equipment Corporation, Maynard Massachusetts, Volume 1-7", P.O.Box 242, Cockeysville, Md 21030.

8. EMC Controls, Incorporated, "EMCON-DR/DS Systems, Distributed Data Acquisition Control System Overview and Configuration Volume 1", P.O.Box 242, Cockeysville, Md 21030.

9. EMC Controls, Incorporated, "EMCON-DR/DS Systems, Distributed Data Acquisition and Control System, Installation, Operation and Drawing Manual, Volume 2", P.O.Box 242, Cockeysville, Md

10. Say M.G., "Performance and Design of Alternating Current Machines", Third Edition 1958, Pitman & Sons, London.

11. Schwarz F.C., "Engineering Information on a Analog Signal to Discrete Time Interval Converter (ASDTIC)", copy right: U.S. Governmant, Publication Office, NASACR-134544, 1975, Washington D.C. 1973.

12. Permantier G.A., "Stabilisation Control Strategies for the Doubly Excited Machine", Thesis submitted to Oregon State University, June 6, 1984.

APPENDICES

Appendix A: Ratings of the Electrical Machines

1. The ratings of the doubly-fed machine shown in Figure 3.1 are:
   The stator is 15 HP, three phase, 60 Hz, 1140 rpm, 230/460 Volts and 42/21 Amps.
   The rotor is three phase, 240 Volts and 32.5 Amps.

2. The ratings of the synchronous generator shown in Figure 3.3 are:
   The stator is 12.5 kVA, three phase, 60 Hz, 10 kW at power factor of 0.8, 110/220 Volts and 65.5/32.8 Amps.
   The field circuit is rated at 125 Volts and 2.5 Amps d.c.

3. The ratings of the synchronous generator shown in Figure 5.1 are:
   The stator is 120 kVA, three phase, 60 Hz at 1800 rpm, 208 Volts and 0.33 Amps.
   The field circuit is rated at 120 Volts and 0.6 Amps with 150 Ohm resistance.

Appendix B: Schwarz AC to DC Converter

Figure G.1 is a schematic of the Schwarz a.c. to d.c. converter used as a controlled current source for the research. This converter was donated to O.S.U. by the Delft University of Technology, Netherlands in May 1985 [10].

Ratings:

Input: 240 Volts, 20 Amps, 3 phase ac.

Output: 200 Volts, 15 amps dc.

Maximum input voltage to the remote terminals: 12 Volts dc

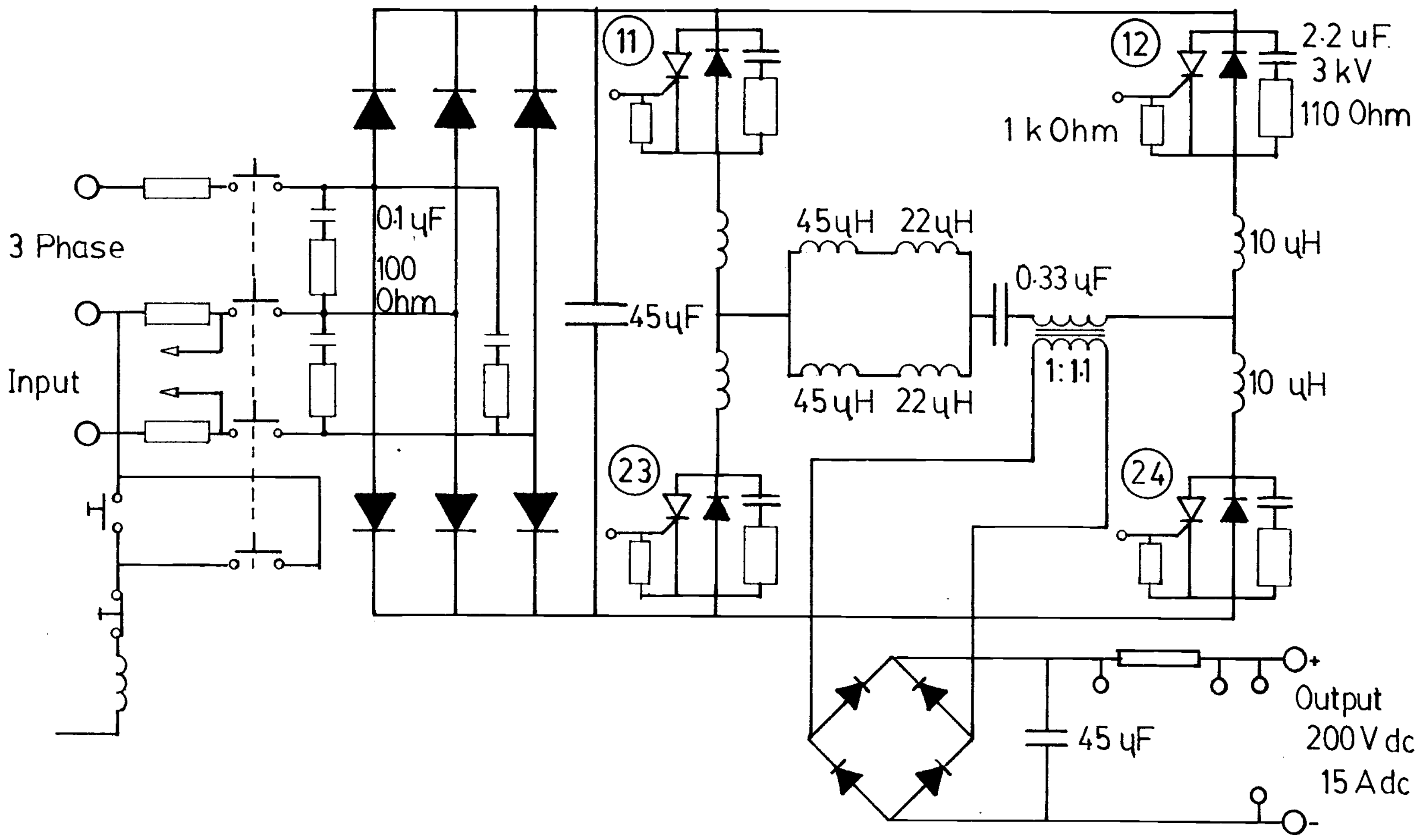


Figure B.1 Main Circuit Diagram of the a.c. to d.c converter (Operation Manual of a.c. to d.c. converter, Netherlands, May 1985)

Appendix C: Current Sensor Interconnection

There are 20 analog input boards in an analog multiplexer in the EMCON-D Process Controller. Each PCB (Printed Circuit Board) carries 5 input signals to the analog multiplexer. The first two inputs are reserved for thermocouples. These are specially compensated to receive positive and negative voltages with magnitudes in the order of millivolts. The last three inputs are for other types of sensors and are polarity sensitive (0 to 10 Volts maximum).

The current transducer (Ohio Semitronics Inc., Model CTA113A; 0 to 100 A rms input, 0 to 10 Volts dc output) output is connected to a free input of an analog multiplexer [9 Section 2.4.1]. Each analog input has three contact points: positive, negative and screen. The positive lead of the transducer output is connected to the positive contact and the negative lead to the negative contact. The screen is connected to the ground of the transducer (if any). In an electrically noisy environment, a twisted pair with the ground is recommended for the connection. This idea is illustrated in Figure C.1.

There is a unique address for each analog input to the multiplexer. The analog input connected to the current transducer output as shown in Figure C.1 has the address of "1,94". The "1" is the Process Control Microprocessor

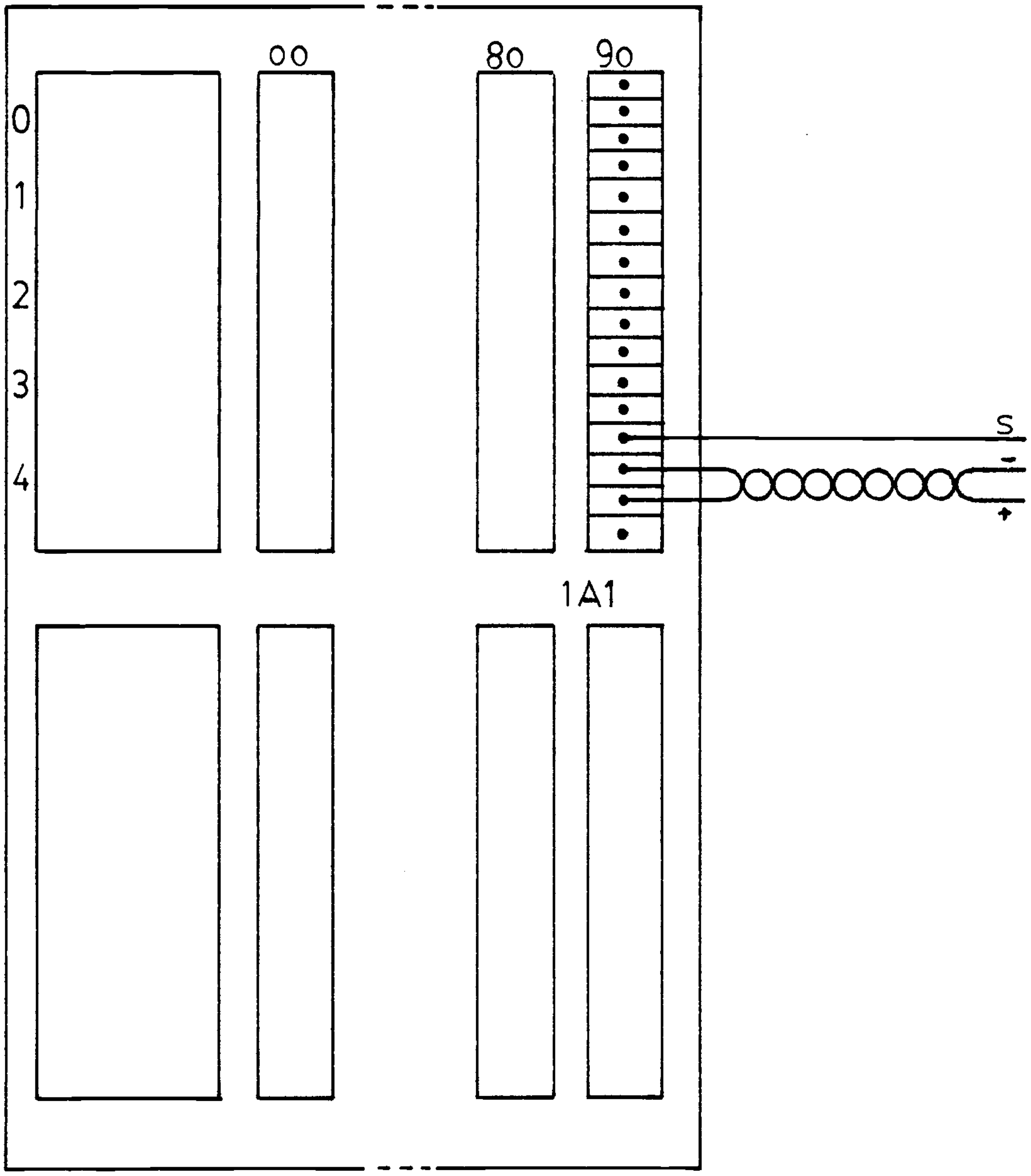


Figure C.1 Connection of the Sensor Output to the Analog Multiplexer

number (PCM 1) while "94" is the Multiplexer Input Number (00-99 inputs for each analog multiplexer) [9, Section 3.2.2.8]. When this variable is defined using the CAM command file, it is necessary to give "1,94" as the response to MUX:PNT (see Appendix D).

Appendix D: CAM Command File

The following listing describes the procedure of defining an analog input to the EMCON-D Process Controller using a CAM command file. For details see Reference 4, Chapter 5.

```
>@[200,200]CAM      <CR> [start from system terminal]
>;+
>; CAM STARTUP COMMAND FILE
>;-
>*  **IS CAM TO RUN FROM A COMMAND  FILE?  [Y/N]:  N  <CR>
LST <CR>  [default for printouts: system terminal]
PCM  1    [working PCM number]
CAM> DEF AIN ARMATURE    [define analog input ARMATURE]
IPN <CR>      [internal point number. default]
MUX: PNT 1,94 <CR> [mux point number. see Appendix C]
GAIN 100 <CR>  [gain factor]
CONV LIN <CR>  [sensor output conversion: linear]
PER 1,1 <CR>   [scan rate: once every second]
EGU 0.000 1.000 AMP <CR> [EGU range of interest]
DISPLAY <CR>  [display: default, same as EGU]
ADI <CR>
ALM A <CR>
HI <CR>    [high alarm limit: default upper bound of EGU]
LO <CR>    [low alarm limit: default lower bound of EGU]
DBAND 0 <CR>
TERM #16 <CR> [terminal word; see FEEDBK]
```

```
DESC MOTOR CURRENT <CR>  [title of the analog input]
CONTROL NO. <CR>  [no control blocks are required]
```

To cascade a control block with this analog input, enter the following description.

```
CONTROL PID <CR>  [combine a PID control block]
PB 100.0 <CR>    [ P = 100.0 ]
RESET 40 <CR>    [ I = 40 ]
RATE 0 <CR>      [ D = 0 ]
SETPT 0.5 <CR>  [ reference value = 0.5 ]
DEVLH <CR>
DEVLL <CR>
GAP <CR>
OPNCLS <CR>
OUTREV <CR>
OUTTYPE POS <CR>  [positional type output is required]
HWTYPE  D54 <CR>   [type of hardware that gives positional
                     type outputs]
FEEDBK #16 <CR>   [feedback for the control block is taken
                    from this terminal word]
OUTPUT 1,6 <CR>  [control block output is taken from  this
                   point. see Appendix I]
>CAM ^Z           [end CAM command file]
  [EXIT]
>* **SAVE HOST DATABASE [Y/N]: Y <CR>
>PIP DD4: [330,54] *.*/PU/NM
>RUN DBSAVE
```

```
     EPNCVT ON 6-AUG-85 15:35:01

     TYPE /#BYTES  HEX OCTAL DECIMAL

     EPN IPN TBL: 4S4A 44112 18506

     6-AUG-85   15:36:23  EPNCUT--NO-1 AT THE END  OF  THE

     TABLE!

     6-AUG-85  15:36:01  DBSAVE--DATABASE SAVE COMPLETE

     6-AUG-85  15:36:01  DBSAVE--DATABASE SAVE COMPLETE


>PIP DD4 [330,54] *.*/PU/NM

>SET /[UIC]=[1,1]

>;+

>; END OF CAM.CMD

>;-

>@[EOF]

>
```

Appendix E: GROUP Command File

After defining the analog input (using a CAM command file), it can be included in one of the groups for display on the color console terminal. The following listing describes the use of GROUP command file to include the analog input ARMATURE in a group. For details see Reference 5, Chapter 14.

```
>@[200,200]GROUP <CR>   [start the GROUP command file]
>;
>; GROUP.CMD
>;
>; ** MODIFIY DEVICE ASSIGNMENTS? [Y/N]: N <CR>
>;
>;
>* *** ARE ASNGRP COMMANDS IN A FILE?? [Y/N]: N <CR>
>;
> SET /UIC= [1,1]
>INS LB:[300,54]ASNGRP
>GRP
GRP>CR <CR>   [create a group]
GROUP  NUMBER (01 TO XX) 07 <CR> [designate this group  as
                                   07]
EPN ARMATURE <CR>    [EPN of the analog input]
EPN // <CR>          [no more inputs]
DISPLAY  MODE  /C  <CR>  [4 over  4  controller  faceplate
                           display mode]
```

```
GROUP TITLE:  DEMONSTRATION <CR>  [group title]

GRP> ^Z      [end GROUP command file]

     [EXIT]

>;

>* *** DO ANOTHER GROUP ASSIGN?? [Y/N]: N <CR>

>;

>* *** SAVE GROUP ASSIGNMENT(S) [Y/N]: Y <CR>

>;

>INS LB:[300,54]DBSAVE

>INS LB:[300,54]EPNCUT

>INS LB:[300,54]DBRSTR

>RUN DBSAVE

     EPNCUT ON      6-AUG-85  15:53:33

     TYPE/$BYTES     HEX  OCTAL  DECIMAL

     EPN-IPN TBL:   484A  44112  18506


     6-AUG-85   15:53:59   EPNCUT--NO-1 AT THE END OF  THE

     TABLE!

     EPNCUT DONE ON 6-AUG-85 AT 15:54:02


     6-AUG-85  15:53:39  DBSAVE--DATABASE SAVE COMPLETE

     6-AUG-85  15:53:35  DBSAVE--DATABASE SAVE COMPLETE

>

>REM GRP

>REM DBSAVE

>REM EPNCUT

>REM DBRSTR
```

```
>; SET /UIC=[1,1]
>;
>; DONE WITH GROUP CMD
>;
>@ <EOF>
>
```

Appendix F: SYSLOD Command File

The modified database is transferred to PCMs, CCM and DCM using the SYSLOD command file. Reference 4, Chapter 2 describes the microprocessor loading procedure in detail.

```
>@[200,200]SYSLOD <CR>
>;+
>; SYSLOD.CMD
>;
>; MICRO LOAD AND STARTUP COMMAND FILE
>;-
>* **MODIFY DEVICE ASSIGNMENTS? [Y/N]: N <CR>
>* ***LOAD ALL MICROS? [Y/N]: Y <CR>
>;+
>;LOAD CCM0
>;-
>MPL STOP $,0
     6-AUG-85 16:11:24 MPL--MP STOPPED
>MPL $0,0
     6-AUG-85 16:11:26 CCM0 NOT SELECTED
     6-AUG-85 16:11:30  CCM1 OFF LINE
>MPL LB0: [110,54] CCMSYS,0
     6-AUG-85 16:12:01  CCM0 OFFLINE
          ALL CCM'S OFFLINE/NOT SELECTED
     6-AUG-85 16:12:08 MPL--MP LOADED
>MPL $1100,0
>MPL RUN $,0
```

```
     6-AUG-85 15:12:16 MPL--MP STARTED

>;+

>; LOAD PCM1

>;-

>MPL STOP $,0,1

     6-AUG-85  16:12:21  MPL--MP STOPPED

>MPL $0,0,1

>MPL $0,0,1

>MPL LB0: [110,54] PCMSYS,0,1

     6-AUG-85  16:13:09 CCM0 SIO TIMEOUT

     6-AUG-85  16:13:10 CCM0 ACTIVE

     6-AUG-85  16:13:10  PCM01A LOAD REQUEST

     6-AUG-85  16:13:16  MPL--MP LOADED

>MPL $1100,0,1

     6-AUG-85  16:13:20  PCM01A STOPPED

     6-AUG-85  16:13:21  PCM01A LOADED

>MPL STOP $,0,1

     6-AUG-85  16:13:24  MPL--MP STOPPED

>MPL DD4:[330,54] PCM01DB.DAT,0,1

     6-AUG-85  16:13:41  MPL--PCM DATABASE LOADED

>MPL DD4: [330,54] PCM01DB ILK,0,1

     6-AUG-85  16:13:50  MPL--INTERLOCK DATABASE  LOADED

>MPL DD5: [330,54] PCM01DB.IPM,0,1

     6-AUG-85  16:13:53 MPL--NON-EXISTENT FILE

>MPL RUN $,0,1

     6-AUG-85  16:13:56  MPL--MP STARTED

     6-AUG-85  16:13:58  PCM01A ONLINE
```

```
>SET /UIC= [1,1]

>@<EOF>

>
```

Appendix G: LOGEN Program

The procedure for using the LOGEN program can be outlined as follows:

Using the host editor, create a file on disk drive number 2, directory [202,10], as follows:

```
>EDI DD2:[202,10]LOG.GEN <CR>
```

Next, specify the desired EPNs to print and their format using LCL (Log Control Language). A sample program is given in the Reference 4, Chapter 10, page 6.

The next step is to preprocess this file in order to verify that the LCL commands in the file are executable. This step is done by the following command:

```
>@[200,200]LOGCHK <CR>
```

The LOGCHK Command File generates listfile DD2:[202,34] LOG.LST and object file DD2:[209,24]LOG.TMP. The object file is to be used by the on-line processor, LOGEN.TSK. Any errors in LOG.GEN will be indicated in LOG.LST. If LOG.GEN is error-free, it is now ready to be executed by the LOGEN program.

Next, install the LOGEN program as follows:

```
>INS[300,54]LOGEN.TSK
```

The next step is to run the LOGEN program periodically. This is done by rescheduling the LOGEN program by the host computer in the specified time interval (1 to 24 hours).

```
>RUN LOGEN /RSI=1H <CR>
```

This will execute the LOGEN program in one hour time intervals. For further details, read Reference 4, Chapter 10.

Appendix H: Historical Trending Package

The procedure for using the Historical Trending Package is as follows:

```
>EDI DL0:[200,200]EPNFIL.DAT
```

The EPNs are included, one in each line, in the EPNFIL.DAT file.

```
>SET /UIC=[200,200]
>ABO SPNCOL
```

This will deactivate the Historical Trending Package if it is active.

```
>PIP DL0:[200,200]HISREF.DAT;*/DE/NM
```

This deletes the previous datafile. If it is necessary to save previously collected data, skip this and the next step:

```
>RUN [300,54]CRFILE
```

Using data from the EPNFIL.DAT file, CRFILE generates cross reference files and a skeleton data file.

```
>RUN [300,54] SPONCOL M
```

SPONCOL collects data about the EPNs defined in the EPNFIL.DAT file. At the end of the collection cycle, it calls HISTCOL. The HISTCOL program opens the HISREF.DAT data file, then stores and averages all valid data.

```
>SET /UIC=[330,54]
```

```
>RUN [300,54]COMSAV
```

The database modifications are saved. The system now collects the data and records it. The 'TREND' key of the color console terminal can be used to recall and display the data collected.

```
>ABO SPNCOL
```

This will abort the Historical Trending Package when necessary. For more details, read Reference 5, Chapter 16.

Appendix I: AC to DC Converter Interconnection

The output control signal which is connected to the remote terminals of the converter is taken from the digital multiplexer. There are several types of digital input/output cards. Since an analog signal is required for the converter, a D/A converter output card is selected. This type of card is available in the digital multiplexer 1D1. Effectively, the D/A converter card output is a controlled current source. The magnitude of the output current can be varied between 0 and 20 mA. There are 16 channel cards in the digital multiplexer 1D1. One of the cards is selected to provide the control signal. The current signal is converted to a voltage signal by connecting a resistor across the card output. The magnitude of the resistor is determined by the maximum voltage that can be applied across the remote terminals of the converter. This voltage is limited to 12 volts (maximum). The proper resistor size can be selected by:

$$V = I\,R\,,$$

which leads to

$$R = 600 \text{ Ohms},$$

for $V = 12$ V and $I = 20 \times 10^{-3}$ A.

The external connections for the digital input/output card are shown in Figure I.1. The address of the channel card is '1,6'. '1' is the PCM number and '6' is the position of the channel card selected in the digital

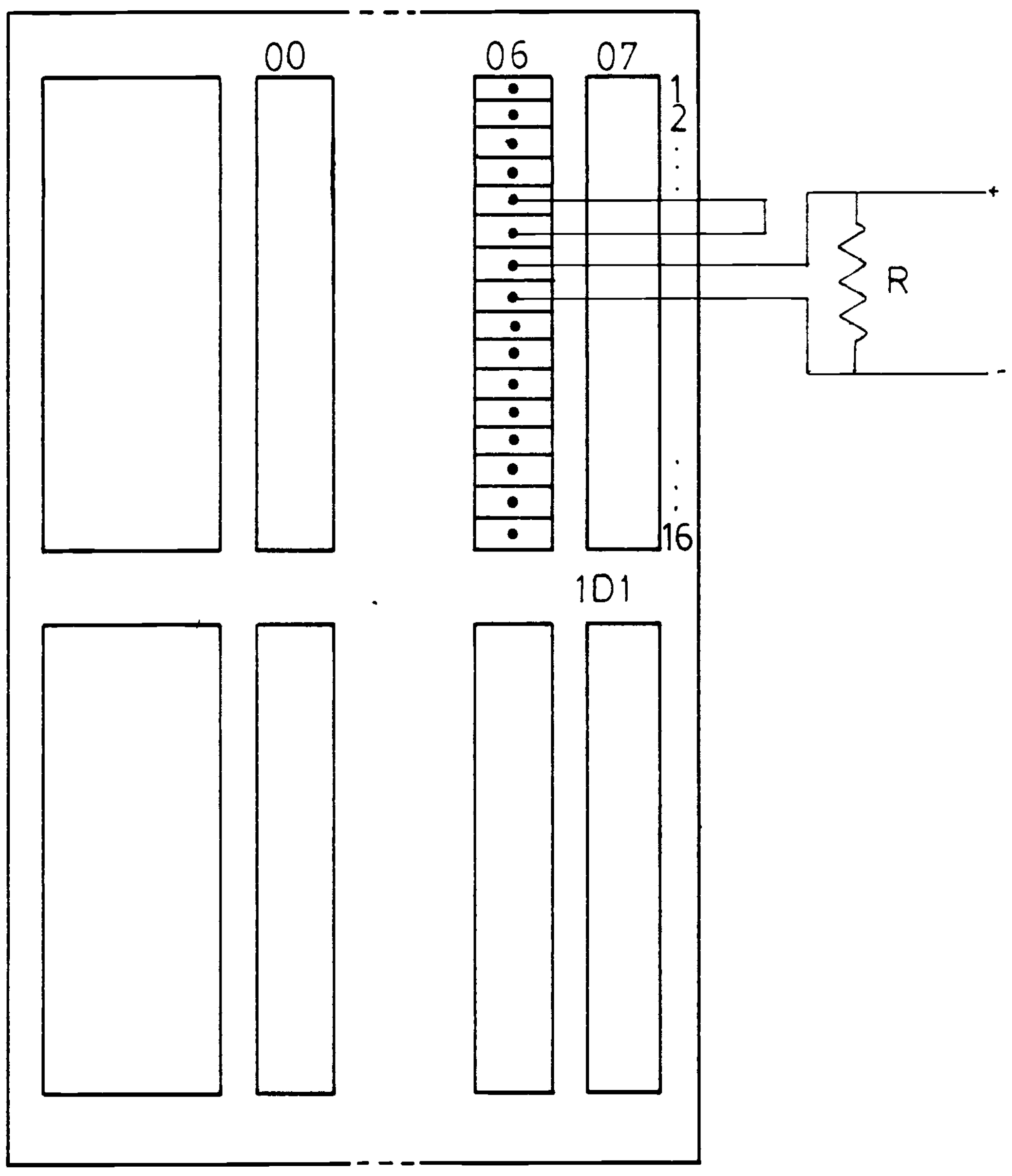


Figure I.1 Analog Output from the Digital Multiplexer

multiplexer. The CAM command file is used to cascade a control block with the analog input 'ARMATURE'. The command file requests the desired position of the control output. "1,6" is given as the response to this request (see the response given to "OUTPUT" of the CAM command file in Appendix D).

Appendix J: <u>FORTRAN Program 1</u>

A typical program written in FORTRAN IV according to the flow chart of Figure 5.5 would be as follows:

```
C THIS PROGRAM IS USED TO CONTROL THE CURRENT OF A
C SYNCHRONOUS GENERATOR. THE DATABASE ACCESSS ROUTINE
C 'GETVAL' IS USED TO FETCH THE CURRENT, REFERENCE VALUE
C AND THE OUTPUT OF THE CONTROL BLOCK.  DIFFERENT FUNCTION
C CODE VALUES ARE USED TO SPECIFY THE VALUES READ.
C FTC1 FOR CURRENT, FTC2 FOR REFERENCE VALUE AND FTC3 FOR
C CONTROL OUTPUT.  THE NAME OF THE CONTROL BLOCK IS GIVEN
C AS AN ARRAY.  ROUTINE 'FTNEPN' USES THE NAME TO GET THE
C GIPN. GIPN IS USED IN ROUTINE 'GETVAL'.
     INTEGER   EPN(5),ERR,GIPN,FTC1,FTC2,FTC3
     REAL  VALUE,CURR,OUT,SETPT
     DATA  FTC1/2/, FTC/14/, FTC/13/
     DATA  EPN/'AR','MA','TU','RE','  '/
C READ THE PRESENT VALUES OF THE CURRENT, REFERENCE
C VALUE AND THE OUTPUT (%).  IF THE 'ERR' TERM IS NEGATIVE
C PRINT ROUTINE IN ERROR.
     CALL  FTNEPN(EPN,GIPN,ERR)
     IF (ERR.LT.0) GOTO 300
     CALL  GETVAL(GIPN,FTC1,VALUE,ERR)
     IF (ERR.LT.0) GOTO 300
     CURR=VALUE
     CALL  GETVAL(GIPN,FTC2,VALUE,ERR)
     IF (ERR.LT.0) GOTO 300
```

```
      SETPT=VALUE
      CALL  GETVAL(GIPN,FTC3,VALUE,ERR)
      IF (ERR.LT.0) GOTO 300
      OUT=VALUE
C THIS PART OF THE PROGRAM DECIDES WHETHER THE CURRENT IS
C GREATER THAN, LESS THAN OR EQUAL TO THE REFERENCE VALUE.
      IF (CURR-SETPT) 100,400,200
C WHEN THE REFERENCE IS GREATER THAN THE CURRENT,
C INCREASE THE OUTPUT BY 1% AND EXIT. THE 'PUTOUT' ROUTINE
C IS USED TO ACCESS THE CONTROL OUTPUT.
  100 VALUE=OUT + 1.
      CALL  PUTOUT(EPN,VALUE,ERR)
      IF (ERR.LT.0) GOTO 300
      GOTO 400
C WHEN THE REFERENCE IS LESS THAN THE CURRENT, DECREASE
C THE OUTPUT BY 1% AND EXIT.  THE 'PUTOUT' ROUTINE IS USED
C TO ACCESS THE CONTROL  OUTPUT.
  200 VALUE=OUT - 1.
      CALL  PUTOUT(EPN,VALUE,ERR)
      IF (ERR.LT.0) GOTO 300
      GOTO 400
C ERROR ENTRY FOR THE ROUTINES
  300 WRITE(6,10)ERR
   10 FORMAT(/,2X,'PROGRAM IN ERROR',2X,I5)
C THIS THE END OF THE PROGRAM.
  400 END
```

Appendix K: FTNACCPRG Command File

Some special directories are required in the system disk to use the FTNACCPRG command file. The source code must be in a file in the directory [340,10]. Further, directories [340,24] and [340,34] must be present before compiling or assembling the source code. The FTNACCPRG command file creates the object file for the source code in the directory [340,24] and the corresponding list file and the map file in the directory [340,34]. If any of those directories are not in the system disk, the user must create these directories. Finally, the executable code is transferred to the directory [300,54]. More details are given in Reference 2, Chapter 9. A typical session follows:

```
>ED [340,10]MSC.FTN <CR>
  (Enter the FORTRAN source code)
  -
  -
 [EXIT]
>@[200,200]FTNACCPRG <CR>
>;+
>; FTNACCPRG CMD
>; BUILD FORTRAN ACCESS PROGRAM
>;
>* **MODIFY DEVICE ASSIGNMENTS?  [Y/N]: N <CR>
>* **MACRO AND/OR FORTRAN ASSEMBLIES DESIRED? [Y/N]: Y <CR>
>* **TASK BUILD ANY PROGRAMS?  [Y/N]: Y <CR>
>* ***ODT DEBUGGING TOOL DESIRED ON TASK BUILDS?  [Y/N]: N
                                                          <CR>
>;
>* ****ENTER FILE NAME (NO EXTENSION) [SR:1-9]:MSC <CR>
>;
>;*  ****DOES YOUR PROGRAM USE LEVEL 3 (CALL GETSPN)?[Y/N]
                                                    :N <CR>
>;
>SET /UIC=[340,24]
>PIP @SY:[340,24]FTNACCDEL.CMD
>FOR @SY:[340,24]FTNACCFOR.CMD
 .MAIN.   (Comment 1: The compilation errors will be
          indicated at this stage)
>SET /UIC=[340,24]
>PIP SY:[340,24]FTNACCDEL.CMD;*/DE/NM,FTNACCFOR.CMD;*
>SET /UIC=[340,24]
>SET /UIC=[1,1]
>PIP @SY:[340,24]FNTACCDEL.CMD
          (Comment 2: If task building is not requested,
          the command file will come to an end at this
          stage)
>TKB @SY:[340,24]FTNACCTKB.CMD
>SET /UIC=[340,24]
>PIP SY:[340,24] FTNACCDEL.CMD;*/DE/NM,FTNACCTKB.CMD;*
>SET /UIC=[1,1]
>;+
>; ** DONE WITH BUILDING FTNACC PROGRAM

>;+

>@ <EOF>

>
```

Appendix L: FORTRAN Program 2

```
C THIS PROGRAM IS USED TO CONTROL THE OUTPUT CURRENT OF A
C SYNCHRONOUS GENERATOR.  THIS PROGRAM IS IMPROVED FOR
C FASTER CONVERGANCE.  A DELAY LOOP IS INCLUDED TO
C DETECT OSCILLATIONS AND OVERSHOOT (IF ANY) IN THE
C OUTPUT CURRENT. TWO SAMPLES OF OUTPUT CURRENT ARE TAKEN,
C ONE BEFORE THE DELAY 'CURR1' AND THE OTHER AFTER THE
C DELAY 'CURR2'.  IF THE DIFFERENCE BETWEEN THESE TWO ARE
C GREATER THAN 0.01, NO ADJUSTMENTS ARE MADE TO THE
C OUTPUT OF THE CONTROL BLOCK.  IF THE DIFFERENCE FALLS
C BETWEEN 0.01, THE PROGRAM CONTINUES.  DATABASE ACCESS
C ROUTINE 'GETVAL' IS USED TO FETCH THE OUTPUT CURRENTS
C (CURR1 AND CURR2), REFERENCE VALUE (SETPT), HIGH ALARM
C LIMIT (MAX) AND THE OUTPUT OF THE CONTROL BLOCK (OUT).
C FTC1 THROUGH FTC4 ARE USED TO DEFINE EACH TERM.  'CHECK'
C IS THE VARIABLE USED IN THE DO LOOP TO ADD A DELAY TO
C THE ALGORITHM.  'PER' IS THE STEP SIZE CALCULATED BY
C THE ALGORITHM.  THE NAME OF THE CONTROL BLOCK IS GIVEN
C AS AN ARRAY.  ROUTINE 'FTNEPN' USE THE NAME OF THE ARRAY
C AND RETURN VARIABLE 'GIPN'.  'GIPN' IS USED BY ROUTINE
C 'GETVAL'.  A TOLERENCE LIMIT IS ADDED TO THE PROGRAM.
C WHEN THE DEVIATION OF CURRENT FROM THE REFERENCE FALLS
C BETWEEN THIS LIMIT NO ADJUSTMENTS ARE NECESSARY TO THE
C OUTPUT OF THE CONTROL BLOCK.
```

```
      INTEGER  EPN(5),ERR,GIPN,FTC1,FTC2,FTC3,FTC4,CHECK
      REAL  VALUE,CURR1,CURR2,OUT,SETPT,MAX,PER
      DATA  FTC1/2/, FTC2/14/, FTC3/13/, FTC4/6/
      DATA  CHECK/0/
      DATA  EPN/'AR','MA','TU','RE','  '/
C READ THE FIRST SAMPLE OF THE OUTPUT CURRENT CURR1
C BEFORE THE DELAY.  IF THE 'ERR' IS NEGATIVE PRINT ERROR
C MESSAGE.
      CALL  FTNEPN(EPN,GIPN,ERR)
      IF(ERR.LT.0) GOTO 300
      CALL  GETVAL(GIPN,FTC1,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
      CURR1=VALUE
C DELAY LOOP
      DO1 I=1,5000
      CHECK=CHECK+1
    1 CONTINUE
C READ THE SECOND SAMPLE OF THE OUTPUT CURRENT 'CURR2'
      CALL  GETVAL(GIPN,FTC1,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
      CURR2=VALUE
C READ THE REST OF THE PARAMETERS
      CALL  GETVAL(GIPN,FTC2,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
      SETPT=VALUE
      CALL  GETVAL(GIPN,FTC3,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
      OUT=VALUE
      CALL  GETVAL(GIPN,FTC4,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
C IF CURR1 AND CURR2 DOES NOT FALL BETWEEN 0.01, EXIT
C FROM THE PROGRAM.
      IF(ABS(CURR1-CURR2).GT.0.01) GOTO 400
C IF CURR1 AND CURR2 FALL BETWEEN 0.01, CONTINUE TO FIND
C THE DEVIATION OF CURRENT FROM THE REFERENCE.  IF THE
C DEVIATION IS LESS THAN 0.009, EXIT FROM THE PROGRAM.
C THIS IS THE TOLERENCE MARGIN SPECIFIED.
      IF(ABS(CURR1-SETPT).LT.0.009) GOTO 400
C OTHERWISE CALCULATE THE STEP SIZE
      PER=((CURR1-SETPT)*100.)/MAX
C DEPENDING ON THE SIGN OF 'PER' PROGRAM DIVIDES INTO
C THREE PATHS
      IF(PER) 100,400,200
C THE REFERENCE IS LARGER THAN THE CURRENT.  INCREASE THE
C OUTPUT OF THE CONTROL BLOCK.  NOTE THAT 20 IS SUBTRACTED
C FROM THE 'OUT' IN ORDER TO CORRECT THE OFF-SET ERROR
C OF THE ROUTINE 'PUTOUT.'
 100  VALUE=OUT-20.+PER
      CALL  PUTOUT(EPN,VALUE,ERR)
      IF(ERR.LT.0) GOTO 300
      GOTO 400
```

```
C THE REFERENCE IS LESS THAN THE CURRENT.  DECREASE THE
C OUTPUT OF THE CONTROL BLOCK.  20 IS SUBTRACTED FROM THE
C 'OUT' TO CORRECT THE OFF-SET ERROR OF THE ROUTINE
C 'PUTOUT.
 200 VALUE=OUT-20.-PER
     CALL  PUTOUT(EPN,VALUE,ERR)
     IF(ERR.LT.0) GOTO 300
     GOTO 400
C ROUTINE ERROR DIAGNOSTICS
 300 WRITE(5,10)ERR
  10 FORMAT(/,2X,'PROGRAM IN ERROR',2X,I5)
C EXIT FROM THE PROGRAM
 400 END
```