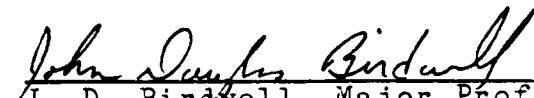
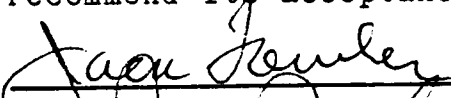
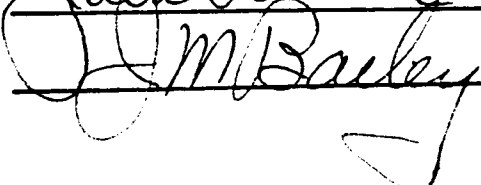


To the Graduate Council:

I am submitting herewith a thesis written by Yu-Hua Wang entitled "A Dynamic Model for an Induction Motor and its Verification." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the Master of Science Degree, with a major in Electrical Engineering.


J. D. Birdwell, Major Professor

We have read this thesis and
recommend its acceptance:

Accepted for the Council:


Vice Chancellor
Graduate Studies and Research

A DYNAMIC MODEL FOR AN INDUCTION
MOTOR AND ITS VERIFICATION

A Thesis
Presented for the
Master of Science
Degree
The University of Tennessee, Knoxville

Yu-Hua Wang

June 1932

3060960

ACKNOWLEDGMENTS

The author would like to thank Dr. Birdwell for his guidance and his encouragement. Without his help and support, this research would not have been possible.

Appreciation is also extended to the other members of my committee, Dr. Lawler and Dr. Bailey, for their useful suggestions and assistance.

A special thanks goes to the Oak Ridge National Laboratory; this research was partially supported by its subcontract 7685 project authorization x40.

ABSTRACT

This experimental work was conducted to verify a dynamic model of an induction motor in the laboratory environment. A computer was used to generate control signals to change the load torque to the induction motor, and to collect the transient data from the motor in response to load torque changes.

The dynamic model of the induction motor chosen for the experiment was a well-known fifth order nonlinear differential model. A derivation of it was presented. The extended Kalman filter was used to estimate the model parameters.

The measured data were compared with those predicted by the fifth order model. It was found that the data predicted by the model agreed well with the experimental data.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION.	1
2. DERIVATION OF THE MODEL FOR AN INDUCTION MOTOR.	4
Assumptions for Deriving the Model.	4
Derivation of the Model	5
The Model in State Variable Form.	21
Summary	23
3. EXPERIMENTAL SETUP.	24
The Motor-generator Unit and Experimental Instruments	24
Electronic Switch	29
Schematic Circuit Diagram of Experiment	30
4. INDUCTION MOTOR PARAMETER DETERMINATION	34
Parameter Experimental Measurement	34
The Extended Kalman Filter.	36
Estimation of the Inertia and Damping Factor of the Motor	42
5. MODEL VERIFICATION.	49
Model Verification.	49
6. CONCLUSIONS AND RECOMMENDATIONS	61
Conclusions	61
Recommendations	62
LIST OF REFERENCES.	63
APPENDIXES.	66
APPENDIX A. COMPUTER HARDWARE DOCUMENTATION	67
APPENDIX B. SOFTWARE DESCRIPTION.	78
APPENDIX C. SOFTWARE LISTINGS	94
VITA.	139

LIST OF TABLES

TABLE	PAGE
1. Definitions and Units for Parameters Used in Figure 1.	7
2. Definitions and Units for Parameters Used in the Derivation.	13
3. Basic Nameplate Data of Electrical Machines	27
4. Basic Nameplate Data of Transducers	28
5. Parts List for High Power Transistor Switch	32
6. Summary of the Extended Kalman Filter Equations	41
7. Estimated Values for J and D of the Motor	47
8. The Parameters of the Experimental Induction Motor	48
9. The Back-plane Assignment	72
10. The Device Register and Interrupt Vector Addresses	73
11. Summary of Commands for the Program MODE11.	88
12. Summary of Control Words for the Program COMUT.	93

LIST OF FIGURES

FIGURE	PAGE
1. Concentrated Coil Representation of a Three-phase Induction Motor	6
2. Concentrated Coil Representation of a Two-phase Induction Motor	14
3. Concentrated Coil Representation of a Two-phase Induction Motor in a Stationary Reference Frame .	16
4. Concentrated Coil Representation of a Two-phase Induction Motor in a Synchronous Reference Frame .	19
5. Composite Motor-generator Unit	25
6. High Power Transistor Switch Circuit	31
7. The Schematic Circuit Diagram of the Experiment. . .	33
8. Timing Diagram of the Kalman Filter.	37
9. Power and Slip Variation for Load Torque Change from 30% to 56%	50
10. Power and Slip Variation for Load Torque Change from 53% to 75%	51
11. Power and Slip Variation for Load Torque Change from 74% to 91%	52
12. Power and Slip Variation for Load Torque Change from 94% to 76%	53
13. Power and Slip Variation for Load Torque Change from 114% to 96%.	54
14. Power and Slip Variation for Load Torque Change from 115% to 136%	55
15. Power and Slip Variation of the Motor Started from Rest	56
16. Computer System Architecture	68

FIGURE	PAGE
17. The Back-planes and Their Slot Assignment.	71
18. Power Restart, Single Instruction and Line Frequency Clock Circuit.	75
19. Diagrams of the Control Panel and Output Panel . . .	76
20. The Typical Load Torque Generated by Command BN. . .	82

CHAPTER 1

INTRODUCTION

In the past decade much effort has been expended in the development of dynamic models of induction motors for transient stability studies and for designs of control systems which use motors as drivers. Many simplified models for special applications can be found in the literature [1], [5], [10], [11], [12] and [13]. Almost all of these models are derived from a fifth order nonlinear differential model, which considers electrical transients in both stator and rotor windings, as well as the mechanical transients.

In references [5], [11], [12] and [13] a model which neglects the stator transient is derived from the fifth order model, and the accuracy of the model is discussed. In reference [1] a linearized version of the fifth order model is presented. In reference [10] a simplified second order model is discussed. All these simplified models give sufficiently accurate results for their special applications.

The main purpose of this work is to choose a dynamic model for an induction motor which should have reasonable accuracy and wide applicability, to identify the model parameters, and to verify the accuracy of the model. From

the models available in the literature, the fifth order model was chosen since it has a general form and good accuracy for many applications. This model will be used as a foundation for further work on models of induction motors in the future, and can be used as a reference model on the computer for load transient modeling applications after verification.

In this report, the following work is documented:

The transient response of an induction motor was collected, and the data were used to verify the fifth order model and estimate its parameters. The results verified the accuracy of the fifth order model. Specialized interface hardware was designed and fabricated to allow direct computer control of the induction motor's load torque and acquisition of motor response data. A program, which simulated the induction motor on the computer using the fifth order model, was written in the FORTRAN language and run on the DEC-10 computer to verify the model performance.

This report is organized in six chapters. Chapter 2 gives a simple derivation of the fifth order model of an induction motor. Chapter 3 presents the power devices and tailor-made interface hardware used in the experiment. Chapter 4 discusses how to measure and estimate the parameters of the experimental motor for simulating the

fifth order model. An extended Kalman filter is used. Chapter 5 includes the verification of the fifth order model by comparing the experimental data with the predicted data from the model. Chapter 6 summarizes the research and presents recommendations for future research. The detailed computer hardware is presented in Appendix A. Appendices B and C give the description and listings for all the software needed for collecting the transient response of the motor and simulating the fifth order model on the computer.

CHAPTER 2

DERIVATION OF THE MODEL FOR AN INDUCTION MOTOR

In this chapter the assumptions for deriving the model are discussed; these assumptions are justifiable for most applications. The derivation of the fifth order model is presented. As shown later this model considers the electrical transient of both rotor and stator coils, as well as the mechanical transient of the motor.

ASSUMPTIONS FOR DERIVING THE MODEL

In order to simplify the derivation of the model, it is customary to assume the induction motor satisfies the following conditions, which are well accepted in deriving models for induction motors such as in references [1] and [2].

These assumptions are:

1. The windings and excitations in both rotor and stator are balanced.
2. The air gap is smooth. The magnetic coupling between the rotor and stator coils varies cosinusoidally with the electrical angle between their axes.

3. Hysteresis and eddy currents in both rotor and stator are neglected.
4. The distributed winding for each phase of the rotor and stator can be represented by a concentrated coil.
5. The machine is operated on the linear portion of the saturation curve.

The first four assumptions are justifiable under all conditions for which the model may be applied. The fifth assumption is questionable for radical variations in terminal voltage or for overload. Since none of the assumptions are very restrictive, the derived model should have wide applicability.

DERIVATION OF THE MODEL

Although the mathematical derivation of the model based on the previous assumptions can be found in reference [1], a simple derivation of the model is presented here in order to clarify the notation and make this work self-contained. The concentrated coil representation of the three-phase induction motor is shown in Figure 1. Table 1 lists all definitions and units for all parameters appearing in Figure 1.

Applying Faraday's induction law to each coil of the

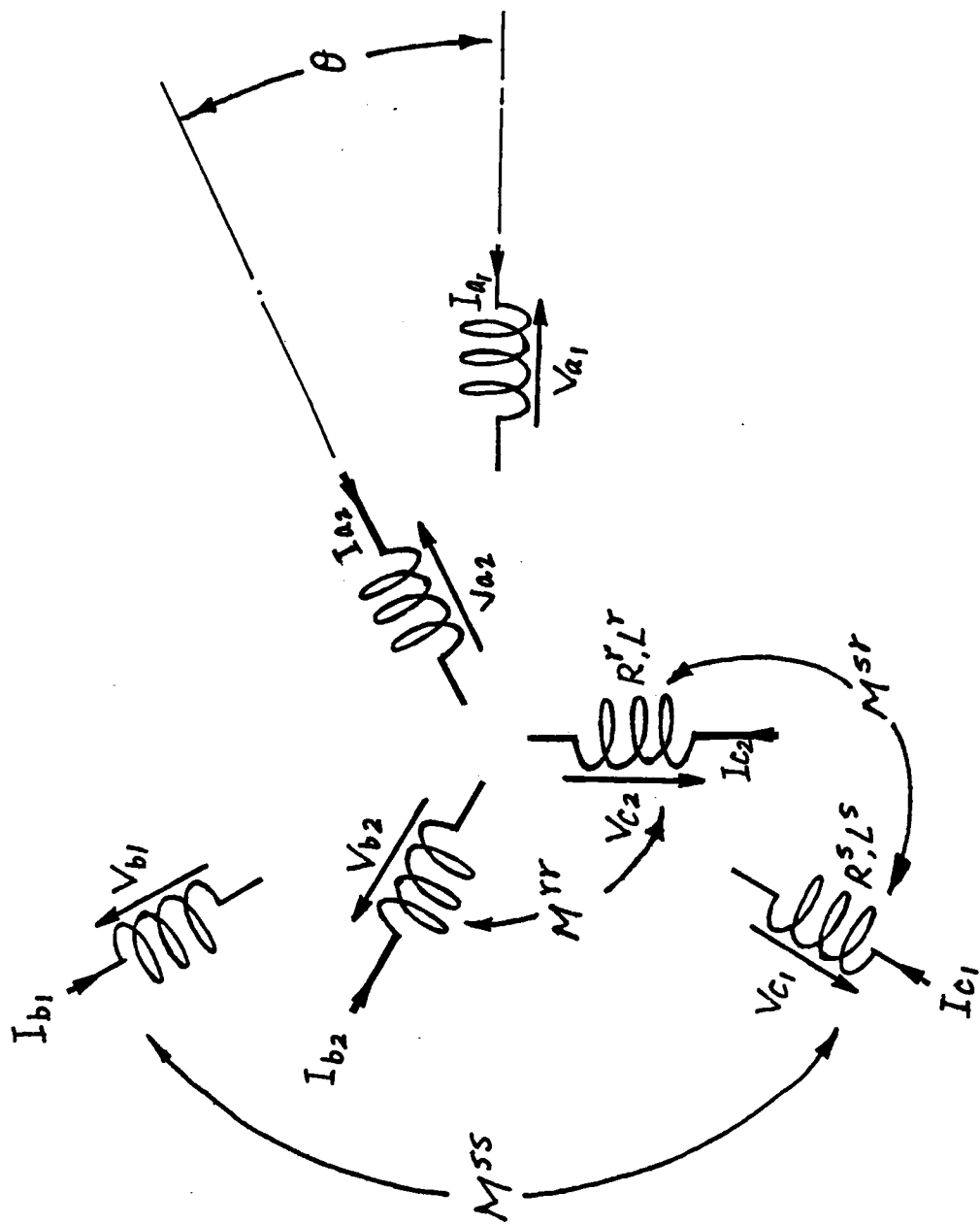


Figure 1. Concentrated Coil Representation of a Three-phase Induction Motor.

Table 1

Definitions and Units for Parameters Used in Figure 1

Parameter	Definition	Unit
R^s	Per phase resistance of stator coil of a three-phase motor	Ω
R^r	Per phase resistance of rotor coil of a three-phase motor	Ω
L^s	Per phase self-inductance of stator coil of a three-phase motor	H
L^r	Per phase self-inductance of rotor coil of a three-phase motor	H
M^{sr}	Per phase maximum value of mutual-inductance between rotor and stator coils of a three-phase motor	H
M^{ss}	Per phase mutual-inductance between stator coils of a three-phase motor	H
M^{rr}	Per phase mutual-inductance between rotor coils of a three-phase motor	H
θ	Per phase electrical angle between a stator coil and its corresponding rotor coil	rad/s
$\underline{V}_i, \underline{I}_i$	With appropriate subscripts represent the phase voltage and current in a three-phase motor	V, A

stator and rotor, six voltage equations may be derived. For example the equation for a stator coil of the C phase is :

$$V_{C_1} = R^S I_{C_1} + L^S \frac{dI_{C_1}}{dt} + M^{ss} \frac{dI_{A_1}}{dt} + M^{ss} \frac{dI_{B_1}}{dt} + M^{sr} \cos 0 \frac{dI_{C_2}}{dt} + M^{sr} \cos(\theta + 120^\circ) \frac{dI_{A_2}}{dt} + M^{sr} \cos(\theta + 240^\circ) \frac{dI_{B_2}}{dt} \quad (1)$$

Assuming the motor is balanced and powered by three phases,

let

$$I_{C_1} = A \cos \omega t \quad (2)$$

then

$$I_{B_1} = A \cos(\omega t + 240^\circ) \quad (3)$$

$$I_{A_1} = A \cos(\omega t + 120^\circ) \quad (4)$$

$$\frac{dI_{A_1}}{dt} = -A\omega \sin(\omega t + 120^\circ) \quad (5)$$

$$\frac{dI_{B_1}}{dt} = -A\omega \sin(\omega t + 240^\circ) \quad (6)$$

therefore

$$M^{ss} \frac{dI_{A_1}}{dt} = -M^{ss} \frac{dI_{B_1}}{dt} \quad (7)$$

That is, the net voltage in the C phase coil of the stator induced by the A and B phase coils of the stator is zero. The same effect occurs for the rotor coils as long as the phase coils are distributed evenly.

Applying equation 7, equation 1 becomes:

$$V_{c1} = R^s I_{c1} + L^s \frac{dI_{c1}}{dt} + M^{sr} \cos \theta \frac{dI_{c1}}{dt} + M^{sr} \cos(\theta + 240^\circ) \frac{dI_{b2}}{dt} + M^{sr} \cos(\theta + 120^\circ) \frac{dI_{a1}}{dt} \quad (8)$$

Using matrix notation the equations may be written as follows:

$$\underline{V}_1 = [R_1] \underline{I}_1 + D(\underline{I}_1) \quad (9)$$

where

$$D(\cdot) = \frac{d}{dt}(\cdot) \quad (10)$$

$$[R_1] = \begin{bmatrix} R^s & & & \\ & R^s & & \underline{0} \\ & & R^s & \\ & & & R^r \\ \underline{0} & & & R^r & R^r \\ & & & & R^r \end{bmatrix} \quad (11)$$

$$\underline{V}_1 = \begin{bmatrix} V_{a1} \\ V_{b1} \\ V_{c1} \\ V_{a2} \\ V_{b2} \\ V_{c2} \end{bmatrix} \quad (12)$$

$$\underline{I_1} = \begin{bmatrix} I_{a1} \\ I_{b1} \\ I_{c1} \\ I_{a2} \\ I_{b2} \\ I_{c2} \end{bmatrix} \quad (13)$$

and

$$\underline{\lambda_1} = \begin{bmatrix} L^S & M^{SY} & M^{SY} & M_1 & M_2 & M_2 \\ M^{SY} & L^S & M^{SY} & M_2 & M_1 & M_2 \\ M^{SY} & M^{SY} & L^S & M_2 & M_2 & M_1 \\ M_1 & M_2 & M_2 & L^Y & M^{SY} & M^{SY} \\ M_2 & M_1 & M_2 & M^{SY} & L^Y & M^{SY} \\ M_2 & M_2 & M_1 & M^{SY} & M^{SY} & L^Y \end{bmatrix} \begin{bmatrix} I_{a1} \\ I_{b1} \\ I_{c1} \\ I_{a2} \\ I_{b2} \\ I_{c2} \end{bmatrix} \quad (14)$$

where

$$M_1 = M^{SY} \cos \theta \quad (15)$$

$$M_2 = M^{SY} \cos(\theta - 120^\circ) \quad (16)$$

Equation 9 is the initial form of the desired model. Since Faraday's induction law is universal and holds true for any voltage and current waveforms, the model based on equation 9 can be used for dynamic analysis as well as for static analysis.

Equation 9 requires measurement of three phase

voltages and currents for both rotor and stator coils. In order to simplify the calculations, the orthogonal transformation $[C_1]$ is used to transform the three-phase machine to a two-phase machine.

$$[C_1] = \begin{bmatrix} 1 & 0 & \sqrt{1/2} & 0 \\ -1/2 & \sqrt{3}/2 & \sqrt{1/2} & 0 \\ -1/2 & -\sqrt{3}/2 & \sqrt{1/2} & 0 \\ 0 & 1 & 0 & \sqrt{1/2} \\ 0 & -1/2 & \sqrt{3}/2 & \sqrt{1/2} \\ 0 & -1/2 & -\sqrt{3}/2 & \sqrt{1/2} \end{bmatrix} \quad (17)$$

The application of the transformation gives:

$$\underline{I}_1 = [C_1] \underline{I}_2 \quad (18)$$

$$\underline{V}_1 = [C_1] \underline{V}_2 \quad (19)$$

where

$$\underline{I}_2 = \begin{bmatrix} I_{\alpha_1} \\ I_{\beta_1} \\ 0 \\ I_{\alpha_2} \\ I_{\beta_2} \\ 0 \end{bmatrix} \quad (20)$$

$$\underline{V}_2 = \begin{bmatrix} V_{\alpha 1} \\ V_{\beta 1} \\ 0 \\ V_{\alpha 2} \\ V_{\beta 2} \\ 0 \end{bmatrix} \quad (21)$$

Table 2 lists the definitions and parameters used here and in the sequel. Since the machine windings and excitation are balanced, as mentioned above, the in-phase induced voltages will be zero and may be dropped from the equation. Figure 2 shows the concentrated coil representation of this equivalent two-phase induction machine.

The resulting equations are:

$$\begin{bmatrix} V_{\alpha 1} \\ V_{\beta 1} \\ V_{\alpha 2} \\ V_{\beta 2} \end{bmatrix} = \begin{bmatrix} R_1 & & & 0 \\ & R_1 & & \\ & & R_2 & \\ 0 & & & R_2 \end{bmatrix} \begin{bmatrix} I_{\alpha 1} \\ I_{\beta 1} \\ I_{\alpha 2} \\ I_{\beta 2} \end{bmatrix} + \begin{bmatrix} L_1 & 0 & M \cos \gamma \theta & -M \sin \gamma \theta \\ 0 & L_1 & M \sin \gamma \theta & M \cos \gamma \theta \\ M \cos \gamma \theta & M \sin \gamma \theta & L_2 & 0 \\ -M \sin \gamma \theta & M \cos \gamma \theta & 0 & L_2 \end{bmatrix} \underline{D} \begin{bmatrix} I_{\alpha 1} \\ I_{\beta 1} \\ I_{\alpha 2} \\ I_{\beta 2} \end{bmatrix} \\ + \gamma \omega_m \begin{bmatrix} & 0 & & -M \sin \gamma \theta & -M \cos \gamma \theta \\ & & & M \cos \gamma \theta & -M \sin \gamma \theta \\ -M \sin \gamma \theta & M \cos \gamma \theta & & \\ -M \cos \gamma \theta & -M \sin \gamma \theta & & 0 \end{bmatrix} \begin{bmatrix} I_{\alpha 1} \\ I_{\beta 1} \\ I_{\alpha 2} \\ I_{\beta 2} \end{bmatrix} \quad (22)$$

Table 2

Definitions and Units for Parameters Used in the Derivation

Parameter	Definition	Unit
R_1	Per phase resistance of stator in a two-phase equivalent motor, $R_1 = R^s$	Ω
R_2	Per phase resistance of rotor in a two-phase equivalent motor, $R_2 = R^r$	Ω
L_1	Per phase self-inductance of a two-phase equivalent motor, $L_1 = 3/2L^s$	H
L_2	Per phase self-inductance of a two-phase equivalent motor, $L_2 = 3/2L^r$	H
M	Per phase maximum value of mutual-inductance between rotor and stator coils of a two-phase equivalent motor, $M = 3/2M^{sr}$	H
ω	Electrical angular frequency	rad/s
ω_m	Mechanical angular frequency	rad/s
γ	Number of pole pairs	—
J	Rotation inertia of rotor	kg-m
D	Damping factor	n-m-s
T	Load torque to the motor	n-m
$\underline{V}_2, \underline{I}_2$	With appropriate subscripts represent the phase voltage and current for a two-phase equivalent motor	V, A
$\underline{V}_3, \underline{I}_3$	With appropriate subscripts represent the phase voltage and current for a two-phase equivalent motor in stationary reference frame	V, A
$\underline{V}, \underline{I}$	With appropriate subscripts represent the phase voltage and current for a two-phase equivalent motor in synchronous reference frame	V, A

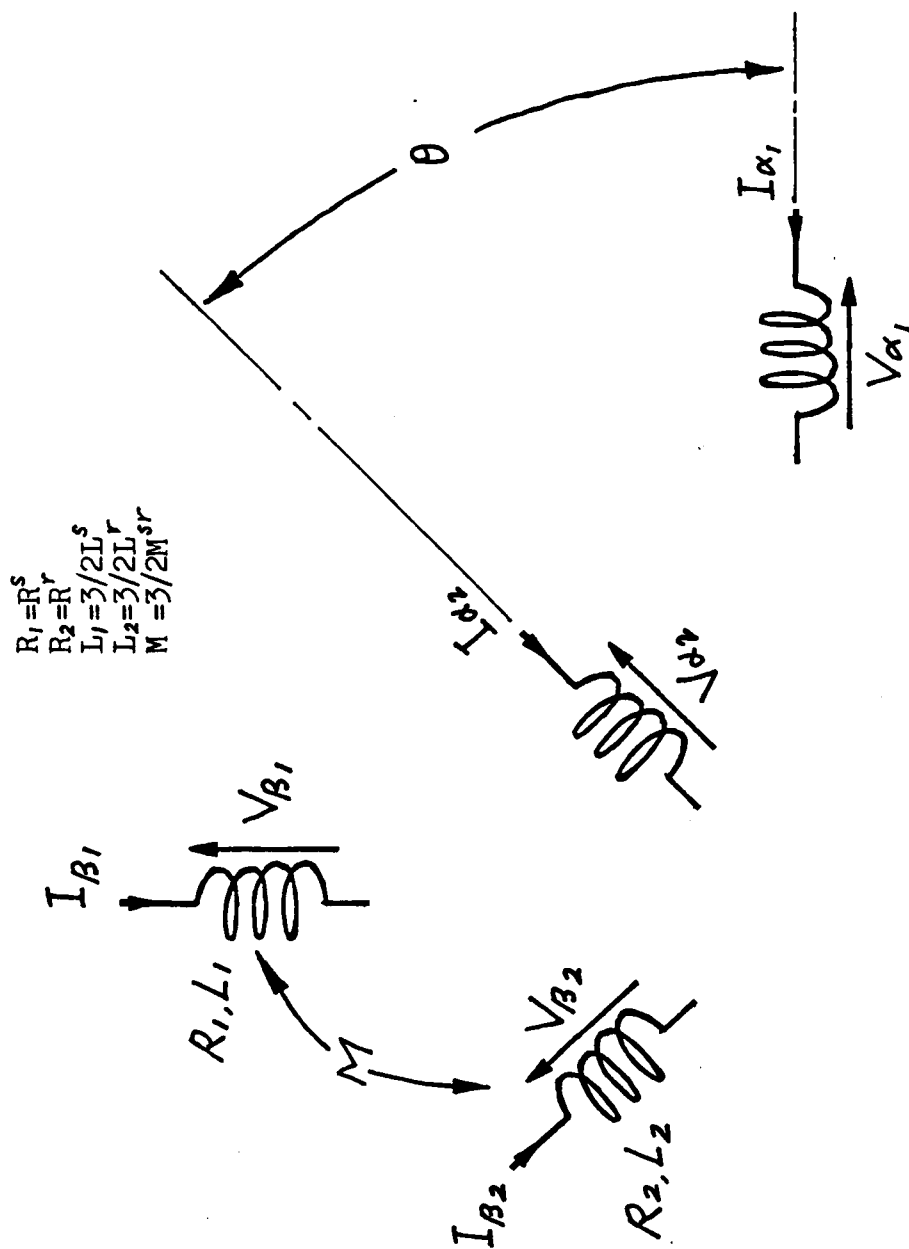


Figure 2. Concentrated Coil Representation of a Two-phase Induction Motor.

Another orthogonal transformation $[C_2]$ is used to reflect the rotor coils to a stationary frame of reference.

$$[C_2] = \begin{bmatrix} 1 & & & \underline{0} \\ & 1 & & \\ \underline{0} & & \cos \gamma \theta & -\sin \gamma \theta \\ & & \sin \gamma \theta & \cos \gamma \theta \end{bmatrix} \quad (23)$$

The application of the transformation gives the following equations:

$$\begin{bmatrix} V_{3\alpha} \\ V_{3\beta} \\ V_{3d} \\ V_{3q} \end{bmatrix} = \begin{bmatrix} R_1 & & \underline{0} \\ & R_1 & & \\ \underline{0} & & R_2 & \\ & & & R_2 \end{bmatrix} \begin{bmatrix} I_{3\alpha} \\ I_{3\beta} \\ I_{3d} \\ I_{3q} \end{bmatrix} + \begin{bmatrix} L_1 & 0 & M & 0 \\ 0 & L_1 & 0 & M \\ M & 0 & L_2 & 0 \\ 0 & M & 0 & L_2 \end{bmatrix} D \begin{bmatrix} I_{3\alpha} \\ I_{3\beta} \\ I_{3d} \\ I_{3q} \end{bmatrix} \\ + \gamma \omega_m \begin{bmatrix} \underline{0} \\ 0 & M & 0 & L_2 \\ -M & 0 & L_2 & 0 \end{bmatrix} \begin{bmatrix} I_{3\alpha} \\ I_{3\beta} \\ I_{3d} \\ I_{3q} \end{bmatrix} \quad (24)$$

Figure 3 shows the representation of this equivalent induction machine. Note: The voltage source in the rotor coil is caused by the rotational inductance.

Another orthogonal transformation $[C_3]$ is used to reflect both the rotor and stator coils to a frame of reference that is rotating at synchronous speed with respect to the stator.

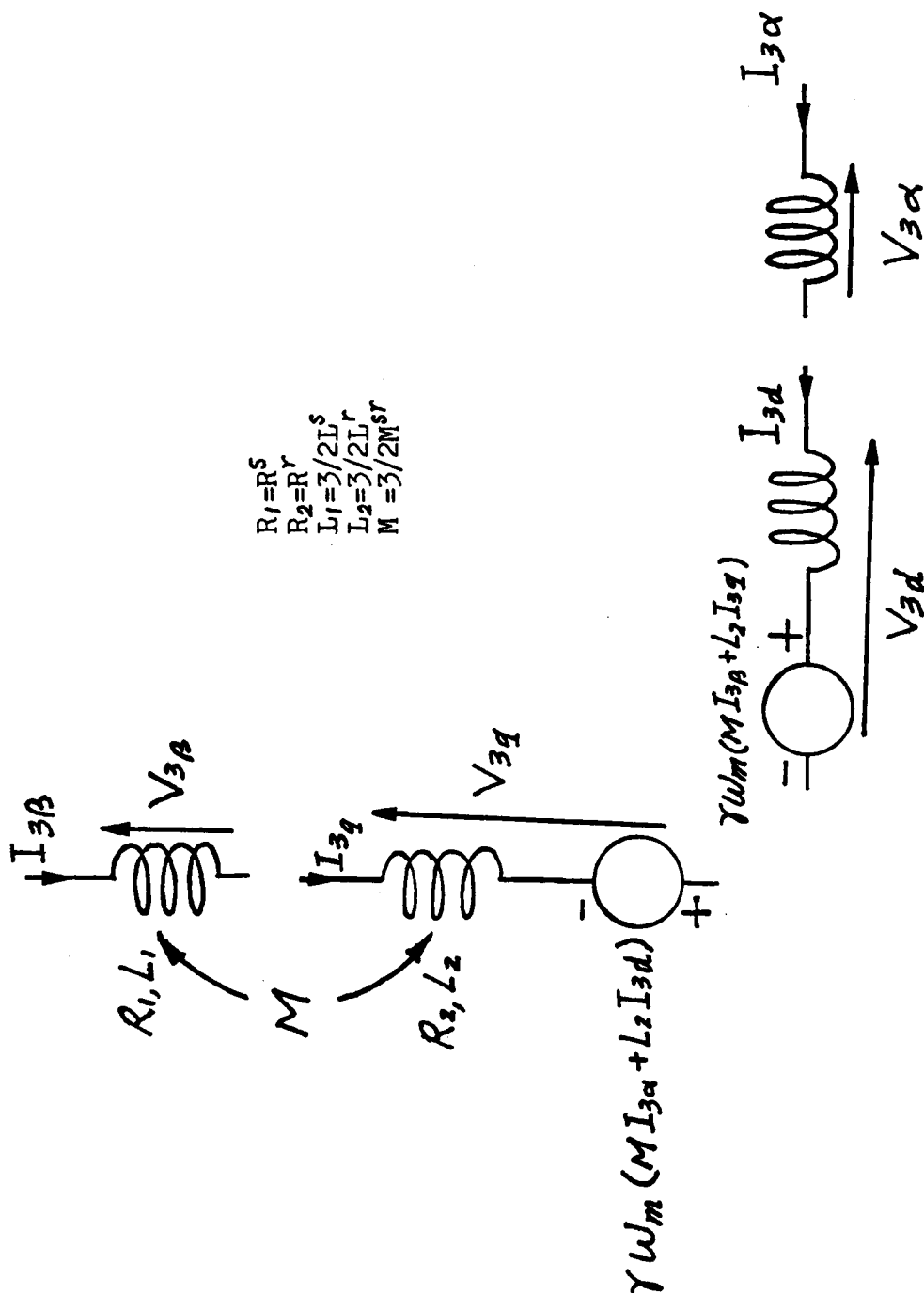


Figure 3. Concentrated Coil Representation of a Two-phase Induction Motor in a Stationary Reference Frame.

$$[C_3] = \begin{bmatrix} \cos \omega t & -\sin \omega t & \underline{0} \\ \sin \omega t & \cos \omega t & \\ \underline{0} & \cos \omega t & -\sin \omega t \\ & \sin \omega t & \cos \omega t \end{bmatrix} \quad (25)$$

Applying the transformation gives:

$$\underline{I}_3 = [C_3] \underline{I} \quad (26)$$

$$\underline{V}_3 = [C_3] \underline{V} \quad (27)$$

where

$$\underline{I}_3 = \begin{bmatrix} I_{3\alpha} \\ I_{3\beta} \\ I_{3d} \\ I_{3q} \end{bmatrix} \quad (28)$$

$$\underline{V}_3 = \begin{bmatrix} V_{3\alpha} \\ V_{3\beta} \\ V_{3d} \\ V_{3q} \end{bmatrix} \quad (29)$$

$$\underline{I} = \begin{bmatrix} I_\alpha \\ I_\beta \\ I_d \\ I_q \end{bmatrix} \quad (30)$$

$$\underline{V} = \begin{bmatrix} V_\alpha \\ V_\beta \\ V_d \\ V_q \end{bmatrix} \quad (31)$$

The resulting equations are:

$$\begin{bmatrix} V_\alpha \\ V_\beta \\ V_d \\ V_q \end{bmatrix} = \begin{bmatrix} R_1 & 0 & 0 & 0 \\ 0 & R_1 & 0 & 0 \\ 0 & 0 & R_2 & 0 \\ 0 & 0 & 0 & R_2 \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \\ I_d \\ I_q \end{bmatrix} + \begin{bmatrix} L_1 & 0 & M & 0 \\ 0 & L_1 & 0 & M \\ M & 0 & L_2 & 0 \\ 0 & M & 0 & L_2 \end{bmatrix} \frac{d}{dt} \begin{bmatrix} I_\alpha \\ I_\beta \\ I_d \\ I_q \end{bmatrix} + \begin{bmatrix} 0 & -\omega L_1 & 0 & -\omega M \\ \omega L_1 & 0 & \omega M & 0 \\ 0 & -(\omega - \gamma \omega_m)M & 0 & -(\omega - \gamma \omega_m)L_2 \\ (\omega - \gamma \omega_m)M & 0 & (\omega - \gamma \omega_m)L_2 & 0 \end{bmatrix} \begin{bmatrix} I_\alpha \\ I_\beta \\ I_d \\ I_q \end{bmatrix} \quad (32)$$

Figure 4 shows the concentrated coil representation of this equivalent induction machine, which is transformed to a rotating reference that is synchronous with respect to the stator.

The power delivered to the motor can be expressed as

$$\underline{P} = \underline{I}^T \underline{V} \quad (33)$$

where $\underline{I}^T$ is the transpose of $\underline{I}$.

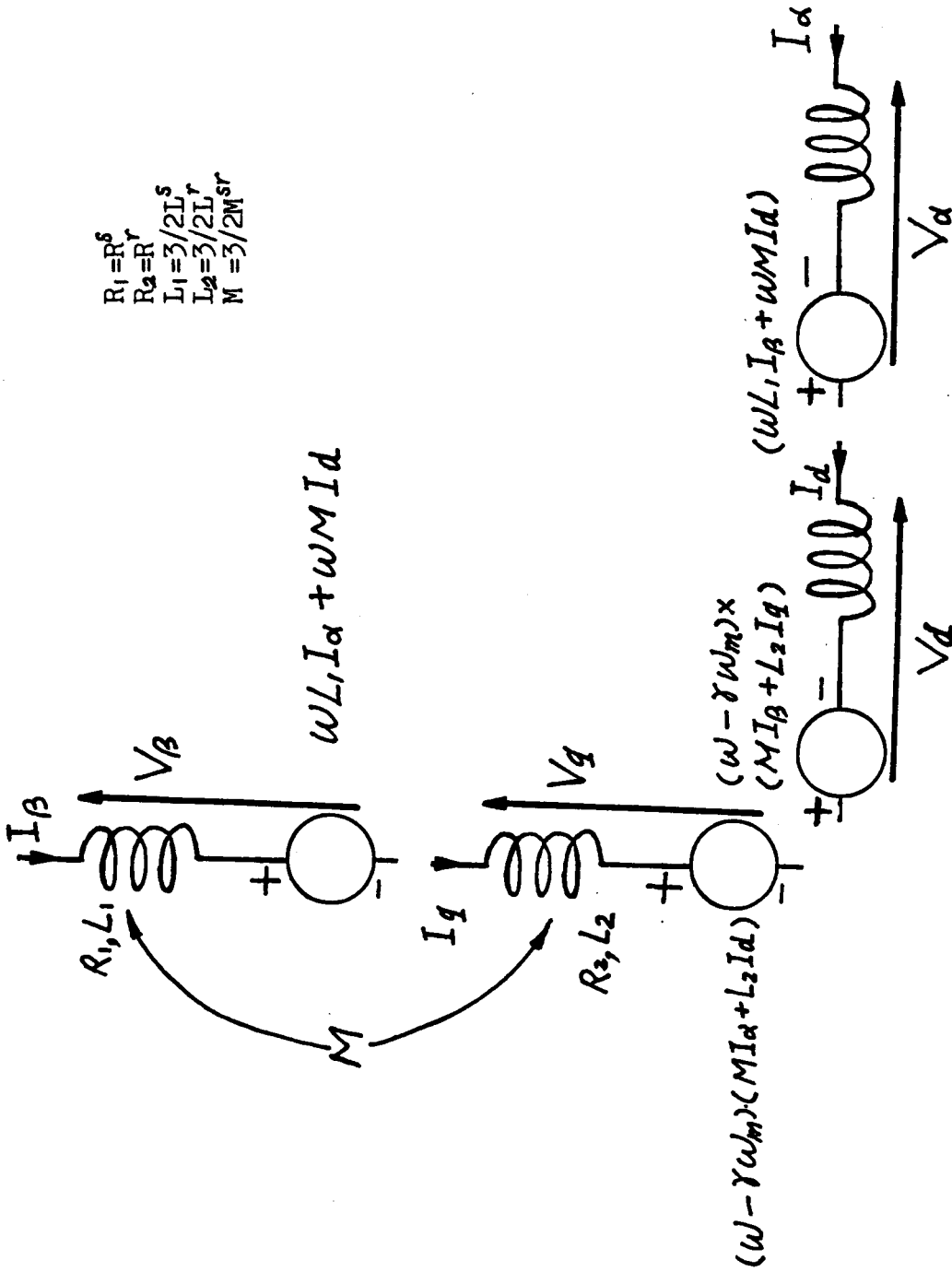


Figure 4. Concentrated Coil Representation of a Two-phase Induction Motor in a Synchronous Reference Frame.

The differential equation describing the mechanical speed of the motor is

$$J \dot{\omega}_m + D \omega_m = T_e - T_L \quad (34)$$

$$T_e = \gamma M (I_\beta I_d - I_\alpha I_f) \quad (35)$$

where T_e is the torque developed by the motor, and T_L is the load torque to the motor.

Equations 32, 33, 34, and 35 describe the dynamic behavior of the motor and are the general form of a nonlinear model for the induction motor.

For the specific case of a short-circuited rotor, the voltage vector becomes

$$\underline{V}_3 = \begin{bmatrix} V_{3\alpha} \\ V_{3\beta} \\ V_{3d} \\ V_{3q} \end{bmatrix} = \begin{bmatrix} V \cos \omega t \\ V \sin \omega t \\ 0 \\ 0 \end{bmatrix} \quad (36)$$

therefore

$$\underline{V} = [C_3]^{-1} \underline{V}_3 = [V \ 0 \ 0 \ 0]^T \quad (37)$$

THE MODEL IN STATE VARIABLE FORM

Choosing the state variables for the system as

$$\underline{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \\ X_5 \end{bmatrix} = \begin{bmatrix} I_\alpha \\ I_d \\ I_\beta \\ I_f \\ \omega_m \end{bmatrix} \quad (38)$$

and after substituting equations 37 and 38 into equations 32, 33, 34, and 35, the following system of first order differential equations results for the short-circuited rotor case.

$$\dot{X}_1 = -R_1 L_2 X_1 / \Delta + R_2 M X_2 / \Delta + \omega X_3 + \gamma M^2 X_3 X_5 / \Delta + \gamma L_2 M X_4 X_5 / \Delta + L_2 V / \Delta \quad (39)$$

$$\dot{X}_2 = R_1 M X_1 / \Delta - R_2 L_1 X_2 / \Delta - \gamma L_1 M X_3 X_5 / \Delta + \omega X_4 - \gamma L_1 L_2 X_4 X_5 / \Delta - M V / \Delta \quad (40)$$

$$\dot{X}_3 = -\omega X_1 - \gamma M^2 X_1 X_5 / \Delta - \gamma L_2 M X_2 X_5 / \Delta - R_1 L_2 X_3 / \Delta + R_2 M X_4 / \Delta \quad (41)$$

$$\dot{X}_4 = \gamma L_1 M X_1 X_5 / \Delta - \omega X_2 + \gamma L_1 L_2 X_2 X_5 / \Delta + R_1 M X_3 / \Delta - R_2 L_1 X_4 / \Delta \quad (42)$$

$$\dot{X}_5 = [\gamma M (X_2 X_3 - X_1 X_4) - D X_5 - T_L] / J \quad (43)$$

$$\text{where } \Delta = L_1 L_2 - M^2 \quad (44)$$

Equations 38 through 44 are the final goal: A dynamic model for a short-circuited rotor induction motor in state variable form. Using this notation the active and reactive

powers delivered to the motor are

$$\begin{aligned}
 P_a = & \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix}^T \begin{bmatrix} R_1 & \underline{0} \\ & R_1 \\ \underline{0} & R_2 \\ & R_2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix} \\
 + & \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix}^T \begin{bmatrix} 0 & -\omega L_1 & 0 & -\omega M \\ \omega L_1 & 0 & \omega M & 0 \\ 0 & -(\omega - \gamma \omega_m)M & 0 & -(\omega - \gamma \omega_m)M \\ (\omega - \gamma \omega_m)M & 0 & (\omega - \gamma \omega_m)M & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix} \quad (45)
 \end{aligned}$$

$$P_r = \omega \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix}^T \begin{bmatrix} L_1 & 0 & M & 0 \\ 0 & L_1 & 0 & M \\ M & 0 & L_2 & 0 \\ 0 & M & 0 & L_2 \end{bmatrix} \begin{bmatrix} X_1 \\ X_3 \\ X_2 \\ X_4 \end{bmatrix} \quad (46)$$

where P_a is the active power delivered to the motor, and P_r is the reactive power delivered to the motor.

The speed of the induction motor is:

$$SP = X_5 \quad (47)$$

SUMMARY

In this chapter the fifth order model for an induction motor was presented in equations 32 through 35. This is a general form of the model. Any variables can be chosen as inputs or outputs of the system for particular applications. The form of the model for the short-circuited rotor case was presented in equations 39 through 43 and equations 45 through 46. In order to be consistent with the laboratory experiment, which is intended to verify the model, the load torque of the motor was chosen as the input; the active and reactive power delivered to the motor and the motor speed were chosen as the outputs of the system. In the sequel the dynamic equations describing the model of the motor will be used to simulate the motor on the computer.

CHAPTER 3

EXPERIMENTAL SETUP

In this chapter all the devices and equipment which were used in the experiment are presented; they are the motor-generator unit, and the its specialized interface hardware and sensors. The circuit diagram for the experiment is included.

THE MOTOR-GENERATOR UNIT AND EXPERIMENTAL INSTRUMENTS

The hardware in the power system laboratory is a motor-generator unit consisting of an induction motor and a DC generator, with a torque/speed transducer on a common shaft, as shown in Figure 5.

The induction motor is a 5 KW wound rotor induction machine manufactured by General Electric. It can be powered at either 220 or 110 volts and operated as either a 1-, 3-, or 6-phase device.

The DC machine is manufactured by Westinghouse and has a rating of 7.5 horsepower as a motor and 5 KW as generator. It can be used as a series, shunt, or compound machine.

The torque transducer mounted on the shaft between the induction motor and the DC machine is manufactured by Lebow. It provides a bipolar 5 volt DC output for 250

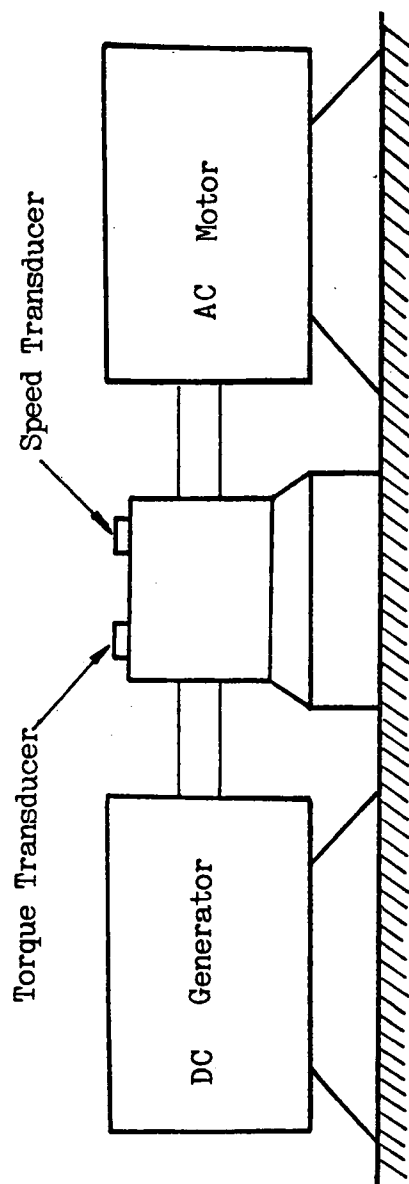


Figure 5. Composite Motor-generator Unit.

IN-LB of torque sensed on the shaft. Its bandwidth is from DC to 10 Hz. It needs an external AC power supply of 115 volts.

The speed transducer mounted on the shaft between the induction and DC machines is also manufactured by Lebow. It provides a unipolar 5 volt output when the shaft is turning at 2000 rpm. It needs a 115 volt AC power supply.

In order to measure the input active and reactive power, the phase voltage, current and active power need to be measured. These three values are monitored by transducers manufactured by Scientific Columbus. All of them have an input to case breakdown voltage of 1800 volts, a 400 ms rise time to 99% of final value and a 1 ma DC full scale output. The power transducer is a two element transducer whose output is accurate regardless of the phase imbalance. It needs an external 110 volt AC power supply and has a full scale output of 1 ma DC for 1000 watts input. The voltage transducer has a full scale output of 1 ma DC for a 120 volt AC input. The current transducer has a full scale output of 1 ma DC for a 5 amp input.

The nameplate data of the two machines and the five transducers used in the experiment are shown in Tables 3 and 4.

Table 3

Basic Nameplate Data of Electrical Machines

Device	Manufacturer	Power Rating	Voltage (v)	Current (a)	Speed (rpm)
DC Shunt Generator	Westinghouse	5 KW	125	40	1200
Induction Motor	Westinghouse	5 KW	220	13	1200 (sync.)

Table 4

Basic Nameplate Data of Transducers

Transducer	Manufacturer	External power	Input	Output	Response Time(99%)	Linearity
Active Power	Scientific Columbus	85-135v 2 watt	0-150v 0-6.25a	0-+1ma (0-1kw)	<400ms	+0.25%
Voltage	Scientific Cloumbus	No	0-150v	0-1ma (0-150v)	400ms	+0.5%
Current	Scientific Cloumbus	No	0-5 a	0-1ma (0-5 a)	400ms	+0.5%
Speed	Lebow	115v	0-2000 (rpm)	0-5 v (0-2000rpm)	no	+0.25%
Torque	Lebow	115v	+250 (IN-LB)	+ 5 v (+250IN-LB)	0-10 Hz	+1%

ELECTRONIC SWITCH

Since the experiment is intended to verify the dynamic model derived in Chapter 2, a method for generating a step function in the load applied to the induction motor needed to be developed. In the motor-generator unit described above, the DC generator is used as a load for the induction motor. In order to get a short rise time of the load torque, the DC generator is run separately excited, and its armature load is a set of resistors which are linked to the armature through electronic switches.

Since there is no power electronic switch available in the power system laboratory, these switches were built with power transistors. Considering the rated value for the induction motor of 5 KW, the DC generator is set at an output voltage of 100 volts, and a maximum current of 50 amps (short duration). Since the armature of the DC machine has some self-inductance, it induces an overshoot voltage when the armature resistor is switched off.

Considering these factors, with a margin for safety, the power transistor must have a minimum breakdown voltage of 200 volts. Unfortunately, in the department electronics shop there is no such transistor available which meets this specification. Transistors 2N6338 and 2N6463 in the shop have the highest breakdown voltages of 120 and 100 volts,

respectively. Therefore, they need to be used in series to raise the breakdown voltage. Although the maximum current of these transistors is 25 amps, the limit of the steady-state current flowing through them is set to 10 amps in order to avoid the occurrence of secondary breakdown during the transition from ON to OFF. The circuit diagram is shown in Figure 6. The parts are listed in Table 5. For a total of 50 amps, 5 electronic switches were built.

SCHEMATIC CIRCUIT DIAGRAM OF EXPERIMENT

The schematic diagram for the primary and second circuits is shown in Figure 7. The cable linking the transducers to the computer is shielded cable and is grounded to the computer metal frame. The wiring for the power transducers must follow the circuit diagrams printed on their cases exactly.

The voltage, current, power, speed and torque transducers are linked to channels 0 through 4 of the A-D converter, respectively. The digital outputs on the A-D converter board are linked to the electronic switches through optical isolators. Since there are only four digital outputs available on the A-D converter board, two of the five electronic switches are paralleled to one digital output. The computer hardware is described in Appendix A.

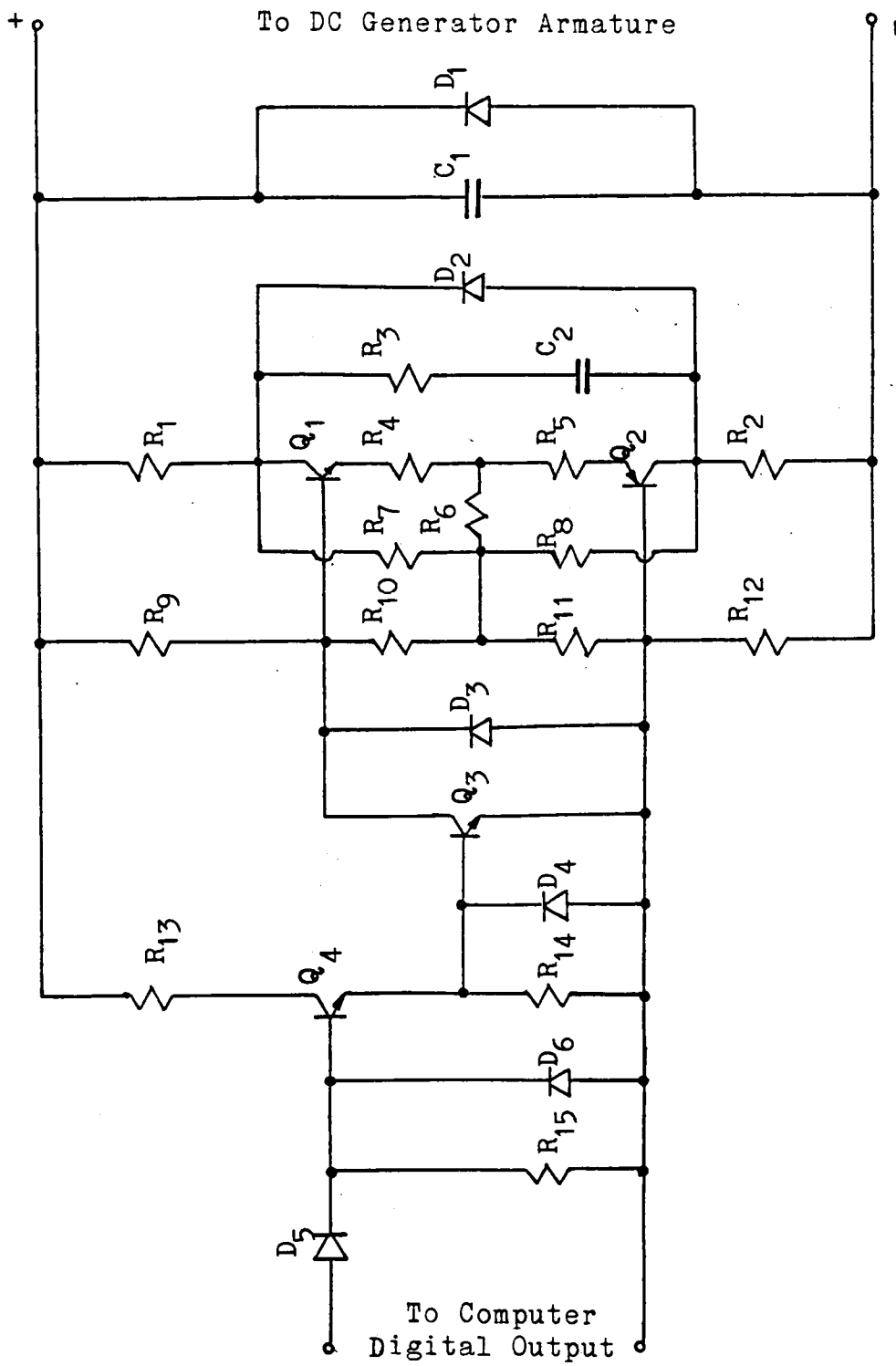


Figure 6. High Power Transistor Switch Circuit.

Table 5

Parts List for High Power Transistor Switch

Parts	Mode or Specification	
C ₁ *	1000 uf,	200 v
C ₂	1 uf,	200 v
D ₁ , D ₂	1N4004,	400 v, 1 a
D ₃ , D ₄ , D ₅ , D ₆	1N4446,	75 v, 200 ma
Q ₁	2N6338,	120 v, 25 a
Q ₂	2N6436,	100 v, 25 a
Q ₃	FT317,	100 v, 5 a
Q ₄	2N6516,	300 v, 0.5 a
R ₁	5.9 Ω ,	600 watt
R ₂	4.5 Ω ,	500 watt
R ₃	2.2 Ω ,	0.5 watt
R ₄ , R ₅	0.05 Ω ,	5 watt
R ₆	60 Ω ,	0.5 watt
R ₇	1 K Ω ,	0.5 watt
R ₈	800 Ω ,	0.5 watt
R ₉	40 Ω ,	50 watt
R ₁₀ , R ₁₁	10 Ω ,	0.5 watt
R ₁₂	32 Ω ,	50 watt
R ₁₃	600 Ω ,	10 watt
R ₁₄	3 K Ω ,	0.5 watt
R ₁₅	10 K Ω ,	0.25 watt

* Since the 1000 uf capacitor is not available in the laboratory, a 3700 uf capacitor was used in the experiment.

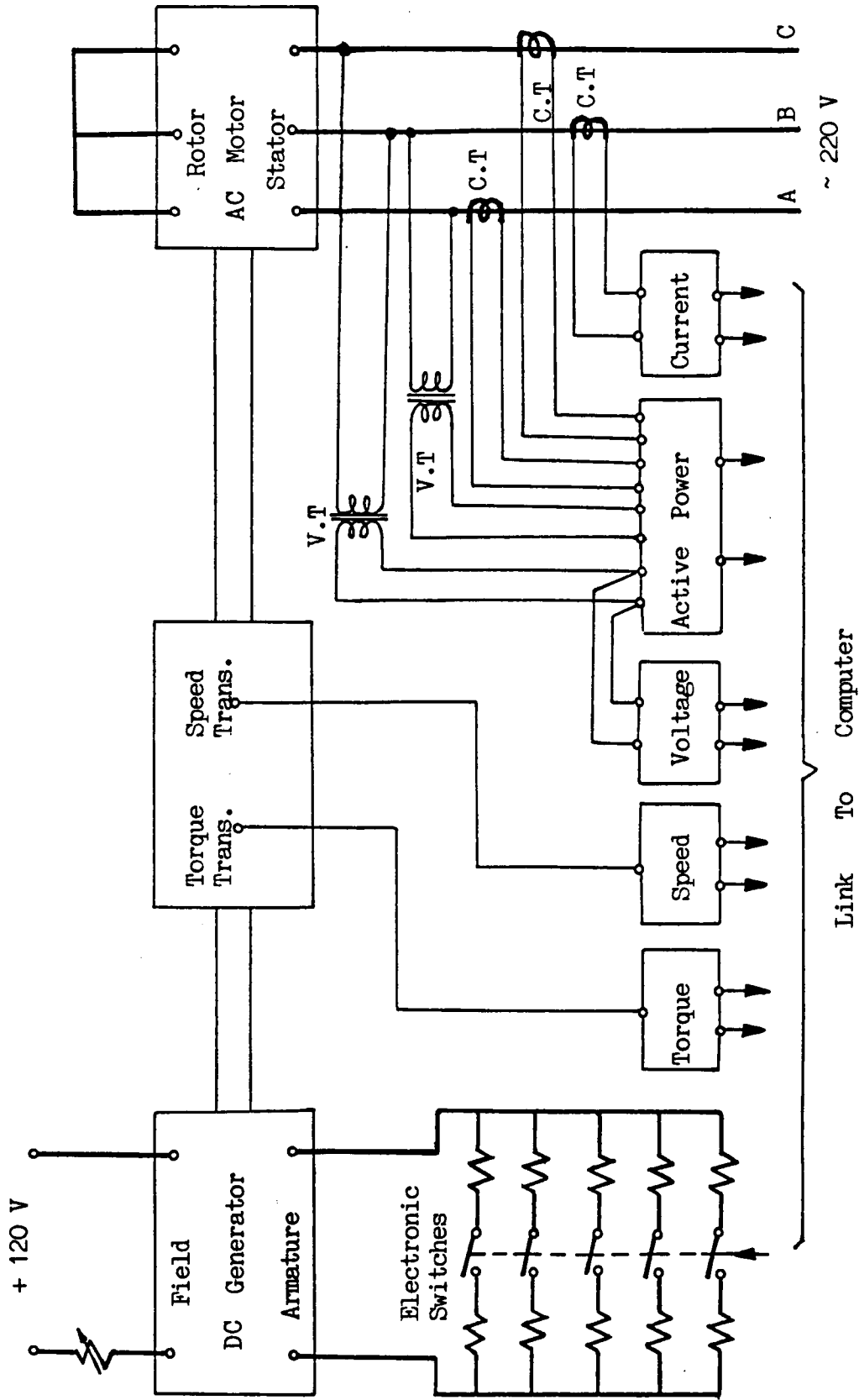


Figure 7. The Schematic Circuit Diagram of the Experiment.

CHAPTER 4

INDUCTION MOTOR PARAMETER DETERMINATION

In this chapter the parameter calculations for the induction motor in the power laboratory are presented. In order to simulate the fifth order model, the self-inductance and resistance of both rotor and stator coils were measured, as well as the mutual inductance between rotor and stator coils. A brief introduction of the extended Kalman Filter is presented, and the filter is applied to the experimental data to estimate the damping factor and rotational inertia.

PARAMETER EXPERIMENTAL MEASUREMENT

In order to simulate the dynamic model on the computer, all the parameters of the experimental induction motor must be known. Since there is no manufacturer-supplied data available, the parameters are found by experiment.

Many methods are available to measure the parameters for an induction motor. The most frequently used one is the block rotor test, which is described in reference [7].

To obtain the model parameters, consider each phase coil of both the stator and rotor as a resistance in series with an inductance. After measuring the voltage across the

phase coil, the current through it and the active or reactive power delivered to it, the resistance and inductance values are easy to find. Together with the block rotor test, the parameters R_1 , R_2 , L_1 , and L_2 can be determined.

The mutual inductance is the constant M such that

$$V_r = M \frac{dI_s}{dt} \quad (48)$$

where V_r is the maximum voltage of an open-circuited rotor phase coil, and I_s is the current through a stator phase coil.

Let

$$I_s = I \cos \omega t \quad (49)$$

then

$$V_r = -M \omega I \sin \omega t = V \sin \omega t \quad (50)$$

Since the readings of the voltage and current meters are the average values of the sinusoidal signals, then after measuring the voltage and current V and I by meters, M can be calculated

$$M = \frac{V}{I \omega} \quad (51)$$

where ω is the power line frequency.

Since the coupling factor between rotor and stator coils is dependent on the relative position of these two coils, by turning the rotor slowly the rotor phase voltage

will change and the maximum voltage V can be found.

THE EXTENDED KALMAN FILTER

Generally speaking any measurement will be inevitably corrupted by noise. In some cases the noise is small, and it can be simply neglected. In other cases the noise may be so large that the measurement is badly corrupted. In the latter case the optimal estimate that minimizes the estimation error can be applied to the corrupted data to find the true values. In Reference [3] a variety of optimal estimation methods are well described. One of them is the Kalman filter, which estimates the state vector at the current time based upon all the past measurements. A brief introduction to the Kalman filter follows.

First consider the Kalman filter for a linear system, which has a continuous system model and a discrete measurement model. The Kalman filter has a recursive form, and the timing diagram with the various quantities involved in the filter equations is presented in Figure 8. The variables t_{K-1} and t_K represent the times when the $K-1$ th and K th measurements occur, respectively; the $(-)$ and the $(+)$ denote times immediately before and immediately after a discrete measurement, respectively.

Let the system state variables be denoted by $\underline{X}$, which is an n element vector for n state variables. The system

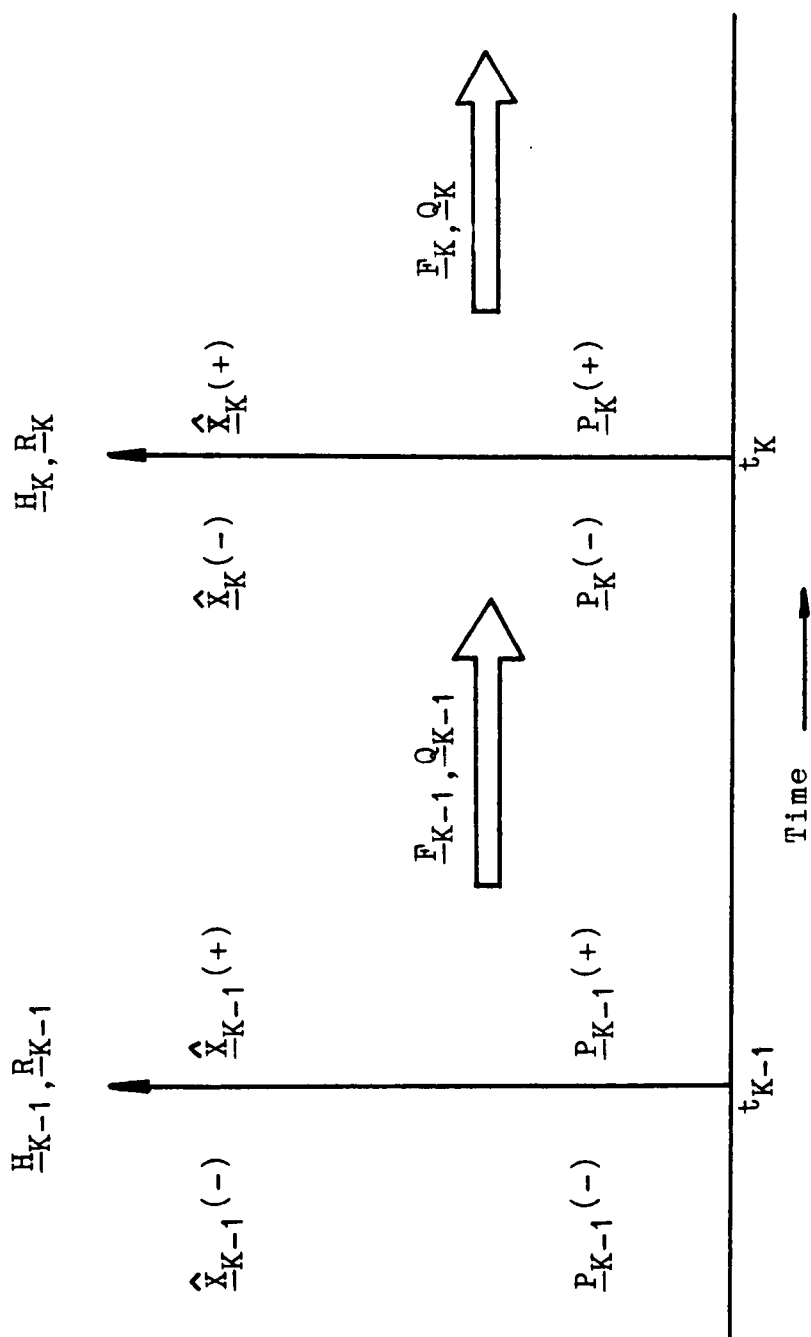


Figure 8. Timing Diagram of the Kalman Filter.

model has the form:

$$\dot{\underline{X}}(t) = \underline{F}(t) \underline{X}(t) + \underline{W}(t) \quad (52)$$

where $\underline{W}(t)$ is a zero mean white noise with covariance $\underline{Q}(t)$.

Let the system measurements be denoted by $\underline{Z}_k$, which is an m element vector for m kinds of measurements. The discrete measurement model at time t_k has the form:

$$\underline{Z}_k = \underline{H}_k(t_k) \underline{X}(t_k) + \underline{V}_k \quad (53)$$

where $\underline{V}_k$ is a zero mean white noise with covariance $\underline{R}_k$.

Let $\underline{X}_{k(-)}$ be the estimate for the state vector $\underline{X}(t_k)$ immediately before measurement $\underline{Z}_k$ is taken at time t_k . The updated estimate for $\underline{X}(t_k)$ immediately after the measurement $\underline{Z}_k$ is calculated by

$$\hat{\underline{X}}_{k(+)} = \hat{\underline{X}}_{k(-)} + \underline{K}_k (\underline{Z}_k - \underline{H}_k \hat{\underline{X}}_{k(-)}) \quad (54)$$

where $\underline{K}_k$ is the Kalman filter gain matrix. Equation 54 is intuitive, and it implies that the state estimate after the measurement at time t_k is equal to the value before the measurement plus a modifier which equals the difference between direct measurements and their predicted values multiplied by a gain.

Let the error covariance between the true values and

the estimated values be denoted by $\underline{P}_K$, which is an n by n matrix. $\underline{P}_{K(-)}$ and $\underline{P}_{K(+)}$ represent the error covariances immediately before and after time t_K . Given $\underline{P}_{K(-)}$, the update equation for $\underline{P}_{K(+)}$ is :

$$\underline{P}_{K(+)} = [\underline{I} - \underline{K}_K \underline{H}_K] \underline{P}_{K(-)} \quad (55)$$

The optimal Kalman filter gain matrix $\underline{K}_K$, which is derived in Reference [3], has the form:

$$\underline{K}_K = \underline{P}_{K(-)} \underline{H}_K^T [\underline{H}_K \underline{P}_{K(-)} \underline{H}_K^T + \underline{R}_K]^{-1} \quad (56)$$

Between two measurements at time t_{K-1} and t_K , the state variable $\underline{X}$ and error covariance $\underline{P}$ should propagate from time t_{K-1} to t_K and have the forms:

$$\dot{\underline{X}}_{(t)} = \underline{f}(\underline{X}_{(t)}, t) \quad (57)$$

$$\dot{\underline{P}}_{(t)} = \underline{F}(\underline{X}_{(t)}, t) \underline{P}_{(t)} + \underline{P}_{(t)} \underline{F}^T(\underline{X}_{(t)}, t) + \underline{Q}_{(t)} \quad (58)$$

The Kalman filter equations described above are for a linear system, but in this work the system model for an induction motor presented in Chapter 2 is nonlinear. Therefore, consider a nonlinear system and assume its system model has the form:

$$\dot{\underline{X}}_{(t)} = \underline{f}(\underline{X}_{(t)}, t) + \underline{W}_{(t)} \quad (59)$$

Let the nonlinear discrete measurement model have the form:

$$\underline{z}_k = \underline{h}_k(\underline{x}(t_k)) + \underline{v}_k \quad (60)$$

The functions $\underline{f}$ and $\underline{h}$ are nonlinear in $\underline{x}$. The method often used for linearization is to expand $\underline{f}$ and $\underline{h}$ as the Taylor series about the current estimates and take the first-order parts of $\underline{F}$ and $\underline{H}$ as their approximations.

$$\underline{F}(\underline{\hat{x}}(t), t) = \left. \frac{\partial \underline{f}(\underline{x}(t), t)}{\partial \underline{x}(t)} \right|_{\underline{x}(t) = \underline{\hat{x}}(t)} \quad (61)$$

$$\underline{H}_k(\underline{\hat{x}}(t), t) = \left. \frac{\partial \underline{h}_k(\underline{x}(t), t)}{\partial \underline{x}(t_k)} \right|_{\underline{x}(t_k) = \underline{\hat{x}}_k(t)} \quad (62)$$

After linearization, all the other equations for the filter have similar forms to those in the linear case. The derivation for them can be found in Reference [3], and a summary of them is presented in Table 6. Since the filter for a nonlinear system is the natural extension of that for a linear system, it is referred to as the extended Kalman filter. Since it is based on linearizations of the model about the current state estimate, the extended Kalman filter is suboptimal, and $\underline{P}$ is only an approximation to the error covariance matrix; however, this algorithm often works quite well.

Table 6

Summary of the Extended Kalman Filter Equations

System Model Measurement model	$\dot{\underline{X}}(t) = \underline{f}(\underline{X}(t), t) + \underline{W}(t), \quad \underline{W}(t) \sim N(0, \underline{Q}(t))$ $\underline{Z}_K = \underline{h}_K(\underline{X}(t_K)) + \underline{V}_K, \quad K = 1, 2, \dots; \quad \underline{V}_K \sim N(0, \underline{R}_K)$
Initial Conditions Other Assumptions	$\underline{X}(0) \sim N(\hat{\underline{x}}_0, \underline{P}_0)$ $E[\underline{W}(t) \underline{V}_K^T] = 0$
State Estimate Propagation Approximate Error Covariance Propagation	$\dot{\hat{\underline{X}}}(t) = \underline{f}(\hat{\underline{X}}(t), t)$ $\dot{\underline{P}}(t) = \underline{F}(\hat{\underline{X}}(t), t) \underline{P}(t) + \underline{P}(t) \underline{F}^T(\hat{\underline{X}}(t), t) + \underline{Q}(t)$
State Estimate Update Approximate Error Covariance Update Gain Matrix	$\hat{\underline{X}}_{K(+)} = \hat{\underline{X}}_{K(-)} + \underline{K}_K [\underline{Z}_K - \underline{h}_K \hat{\underline{X}}_{K(-)}]$ $\underline{P}_{K(+)} = [\underline{I} - \underline{K}_K \underline{H}_K] \underline{P}_{K(-)}$ $\underline{K}_K = \underline{P}_{K(-)} \underline{H}_K^T [\underline{H}_K \underline{P}_{K(-)} \underline{H}_K^T + \underline{R}_K]^{-1}$
Linearization Equations	$\underline{F}(\hat{\underline{X}}(t), t) = \left. \frac{\partial \underline{f}(\underline{X}(t), t)}{\partial \underline{X}(t)} \right _{\underline{X}(t) = \hat{\underline{X}}(t)}$ $\underline{H}_K(\hat{\underline{X}}_{K(-)}) = \left. \frac{\partial \underline{h}_K(\underline{X}(t_K))}{\partial \underline{X}(t_K)} \right _{\underline{X}(t_K) = \hat{\underline{X}}_{K(-)}}$

ESTIMATION OF THE INERTIA AND DAMPING FACTOR OF THE MOTOR

There is no meter which can directly measure the inertia and damping factor of a motor. The extended Kalman filter, which is described in the previous section, was chosen to estimate these two parameters. In order to implement the Kalman filter, initial estimates of the inertia and damping factor are needed.

As the motor stops after turning off the power supply, there are no voltages and no currents in the motor. Therefore equation 43, which describes the mechanical behavior of the motor, has the form

$$\dot{X}_5 = (-DX_5 - T_L)/J \quad (63)$$

If the load torque is large enough so that the damping torque can be neglected, then equation 63 becomes

$$\dot{X}_5 = -T_L/J \quad (64)$$

The discrete form of this equation is

$$J = \frac{-T_L}{\frac{X_5(t+\Delta t) - X_5(t)}{\Delta t}} \quad (65)$$

After measuring the load torque, motor speed and sampling time, the rotational inertia can be calculated. The same method can be used to estimate the damping factor, except the load torque is set to zero. In that case

equation 63 becomes

$$\dot{X}_5 = -D X_5 / J \quad (66)$$

After measuring the motor speed and sampling time under zero load torque condition, the damping factor can be calculated. Two data sets for these two cases were collected and the approximate results were used as initial values for the Kalman filter.

The experimental data used to drive the filter were collected, which represented the motor response when starting from rest. Four variables were measured; they were AC line voltage, motor speed, load torque and active power delivered to the motor.

The system model for the Kalman filter is nearly the same as the fifth order model derived in Chapter 2, except the rotational inertia J and damping factor D are considered to be constant states driven by white noise. The model has the form:

$$\begin{bmatrix} \dot{X}_1 \\ \dot{X}_2 \\ \dot{X}_3 \\ \dot{X}_4 \\ \dot{X}_5 \\ \dot{J} \\ \dot{D} \end{bmatrix} = \begin{bmatrix} -R_1 L_2 X_1 / \Delta + R_2 M X_2 / \Delta + \omega X_3 + \gamma M^2 X_3 X_5 / \Delta + \gamma L_2 M X_4 X_5 / \Delta + L_2 V / \Delta \\ R_1 M X_1 / \Delta - R_2 L_1 X_2 / \Delta - \gamma L_1 M X_3 X_5 / \Delta + \omega X_4 - \gamma L_1 L_2 X_4 X_5 / \Delta - M V / \Delta \\ -\omega X_1 - \gamma M^2 X_1 X_5 / \Delta - \gamma L_2 M X_2 X_5 / \Delta - R_1 L_2 X_3 / \Delta + R_2 M X_4 / \Delta \\ \gamma L_1 M X_1 X_5 / \Delta - \omega X_2 + \gamma L_1 L_2 X_2 X_5 / \Delta + R_1 M X_3 / \Delta - R_2 L_1 X_4 / \Delta \\ [\gamma M (X_2 X_3 - X_1 X_4) - D X_5 - T_L] / J \\ 0 \\ 0 \end{bmatrix} + \underline{W} \quad (67)$$

where $\underline{W}$ is white noise with zero mean value and error covariance $\underline{Q}$.

The measurement model for the Kalman filter is:

$$\begin{bmatrix} SP(k) \\ Pa(k) \end{bmatrix} = \begin{bmatrix} X_5(k) \\ R_1 (X_1^2(k) + X_3^2(k)) + R_2 (X_2^2(k) + X_4^2(k)) \\ + YMX_5(k) (X_2(k)X_3(k) - X_1(k)X_4(k)) \end{bmatrix} \pm V_K \quad (68)$$

where V_K is the white noise with zero mean value and error covariance $\underline{R}$, $SP(k)$ is the K th measurement of the speed, and $Pa(k)$ is the K th measurement of the active power.

Since there are 7 state variables, the error covariance matrix is 7 by 7. This matrix is symmetric; therefore, only 28 entries are independent. During computation of the state estimate and error covariance, there are a total of 35 differential equations which must be solved numerically. The state estimate propagation equation has the same form as equation 67 except $\underline{W}$ is not present. The error covariance propagation equation is

$$\dot{\underline{P}} = \underline{F}\underline{P} + \underline{P}\underline{F}^T + \underline{Q} \quad (69)$$

where

$$\underline{F} = \begin{bmatrix} -R_1 L_2 / \Delta & R_2 M / \Delta & (\omega + \gamma M^2 X_5 / \Delta) & \gamma L_2 M X_3 / \Delta & \gamma M (M X_3 + L_2 X_4) / \Delta & & \\ R_1 M / \Delta & -R_2 L_1 / \Delta & -\gamma L_1 M X_5 / \Delta & (\omega - \gamma L_1 L_2 X_5 / \Delta) & \gamma L_1 (-M X_3 - L_2 X_4) / \Delta & & \\ -\omega - \gamma M^2 X_5 / \Delta & -\gamma L_2 M X_5 / \Delta & -R_1 L_2 / \Delta & M R_2 / \Delta & -\gamma M (M X_1 + L_2 X_2) / \Delta & & \\ \gamma L_1 M X_5 / \Delta & -\omega + \gamma L_1 L_2 X_5 / \Delta & R_1 M / \Delta & -R_2 L_1 / \Delta & \gamma L_1 (M X_1 + L_2 X_2) / \Delta & & \\ -\gamma M X_4 / \gamma & \gamma M X_3 / \gamma & \gamma M X_2 / \gamma & -\gamma M X_1 / \gamma & -D / \gamma & (-\dot{X}_5 / \gamma) & (-X_5 / \gamma) \\ & & & & & 0 & \end{bmatrix} \quad (70)$$

The Kalman gain matrix is

$$\underline{K}_K = \underline{P}_K \underline{H}_K^T \left[\underline{H}_K \underline{P}_K \underline{H}_K^T + \underline{R}_K \right]^{-1} \quad (71)$$

where

$$\underline{H}_K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ (2R_1 X_1 & (2R_2 X_2 & (2R_1 X_3 & (2R_2 X_4 - [\gamma M \cdot \\ -\gamma M X_4 X_5) & +\gamma M X_3 X_5) & +\gamma M X_2 X_5) & \gamma M X_1 X_5) & (X_2 X_3 - X_1 X_4)] & 0 & 0 \end{bmatrix} \quad (72)$$

The state estimate update equation is

$$\underline{X}_K(+) = \underline{X}_K(-) + \underline{K}_K \underline{\Delta H}_K \quad (73)$$

where

$$\underline{\Delta H}_K = \begin{bmatrix} SP(K) - X_5(K) \\ P_a(K) - R_1(X_1^2(K) + X_3^2(K)) - R_2(X_2^2(K) + X_4^2(K)) \\ -\gamma M X_5(K)(X_2(K)X_3(K) - X_1(K)X_4(K)) \end{bmatrix} \quad (74)$$

The error covariance update equation is

$$\underline{P}_{K(+)} = \left[\underline{I} - \underline{K}_K \underline{H}_K \right] \underline{P}_{K(-)} \quad (75)$$

The program for the Kalman filter was written in FORTRAN and executed on the DEC-10 computer. The results are listed in Table 7. The sampling time for this data is 0.0333 second. All the model parameters of the experimental motor are listed in Table 8.

Table 7

Estimated Values for J and D of the Motor

N	1	2	3	4	5	6
J	0.200	0.200	0.200	0.200	0.200	0.200
D	0.0500	0.0500	0.0500	0.0500	0.0500	0.0500
N	7	8	9	10	11	12
J	0.200	0.200	0.201	0.201	0.201	0.201
D	0.0500	0.0500	0.0500	0.0501	0.0501	0.0501
N	13	14	15	16	17	18
J	0.201	0.201	0.201	0.201	0.201	0.201
D	0.0501	0.0501	0.0501	0.0500	0.0500	0.0499
N	19	20	21	22		
J	0.201	0.201	0.201	0.201		
D	0.0500	0.0500	0.0500	0.0500		

Table 8

The Parameters of the Experimental Induction Motor

Parameter	Name	Symbol	Value
Per Phase Stator Resistance		R^s	0.500 Ω
Per Phase Rotor Resistance		R^r	0.650 Ω
Per Phase Stator Inductance		L^s	0.0346 H
Per Phase Rotor Inductance		L^r	0.0346 H
Per Phase Maximum Mutual Inductance between Rotor and Stator Coils		M^{sr}	0.0333 H
Rotation Inertia		J	0.201 kg-m
Damping Factor		D	0.0500 n-m-s

CHAPTER 5

MODEL VERIFICATION

In this chapter a comparison between the experimental data and the predicted data from the model is presented. The model for an induction motor, which is derived in Chapter 2, is proved to have good accuracy. All the software for the experiment and the documentation are presented in Appendix B and C.

MODEL VERIFICATION

The data collected by experiment and predicated by the model cover a wide range of load conditions, from 25% rated value to 130% rated value. The results are plotted in Figure 9 through Figure 14.

One set of data which describes the motor starting from rest was collected. The motor was started with the rotor short-circuited, the terminal voltage of the stator at line voltage, and the DC generator open-circuited. The result was plotted in Figure 15.

The rated values for the experimental induction motor are:

rated active power delivered to it is 5 KW,
rated reactive power delivered to it is 3250 var,
rated load torque is 32 nt-m.

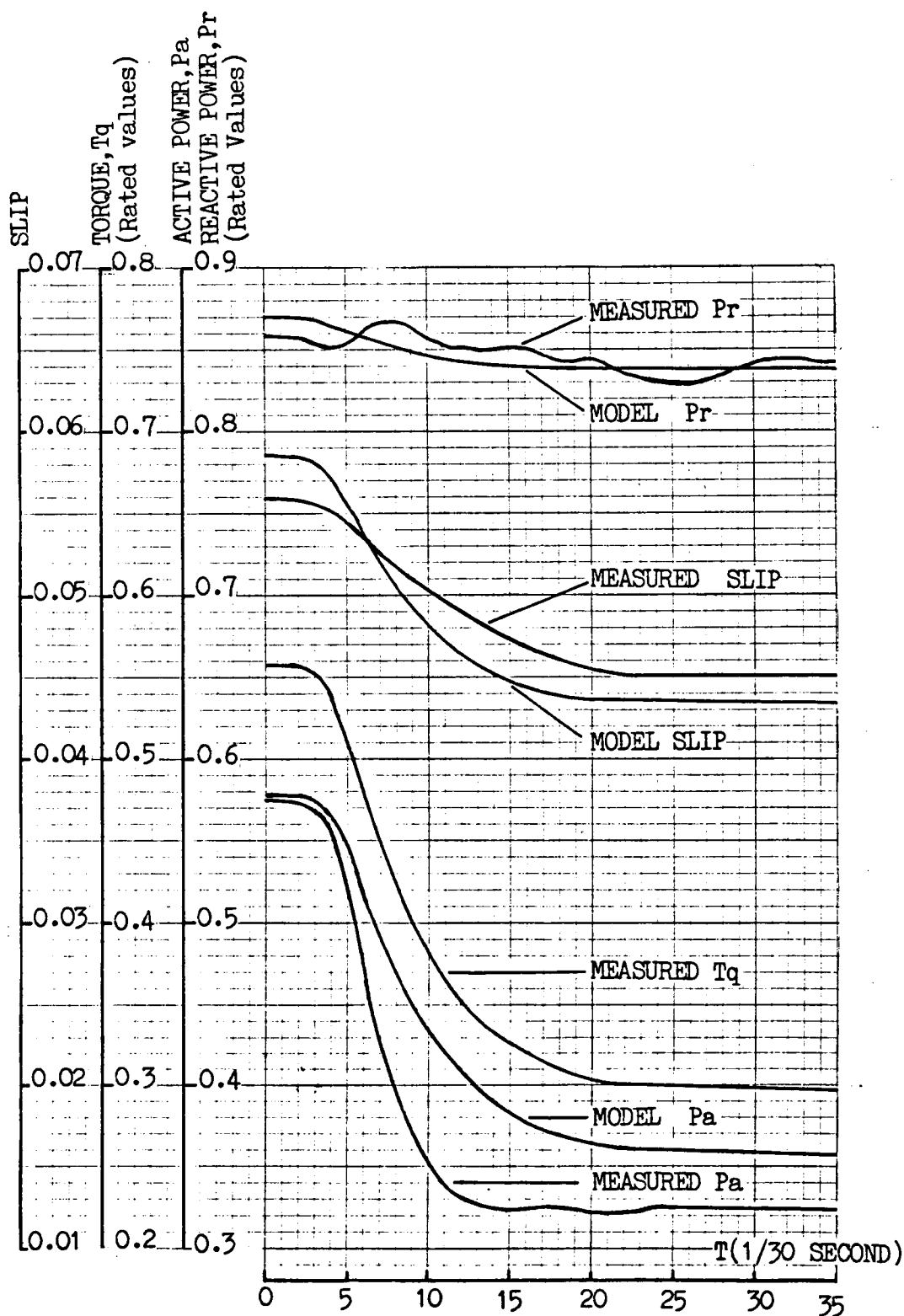


Figure 9. Power and Slip Variation for Load Torque Change from 30% to 56%.

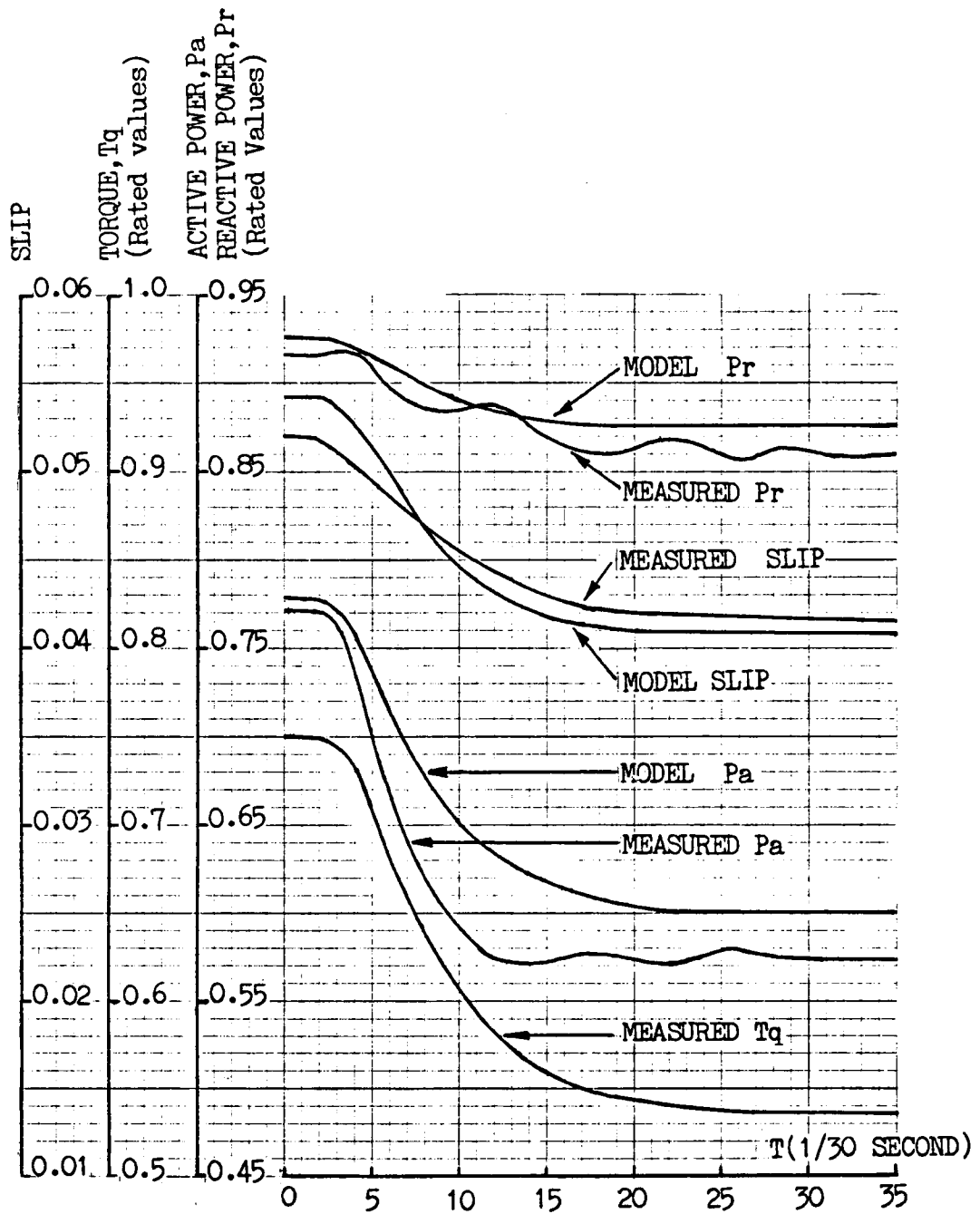


Figure 10. Power and Slip Variation for Load Torque Change from 53% to 75%.

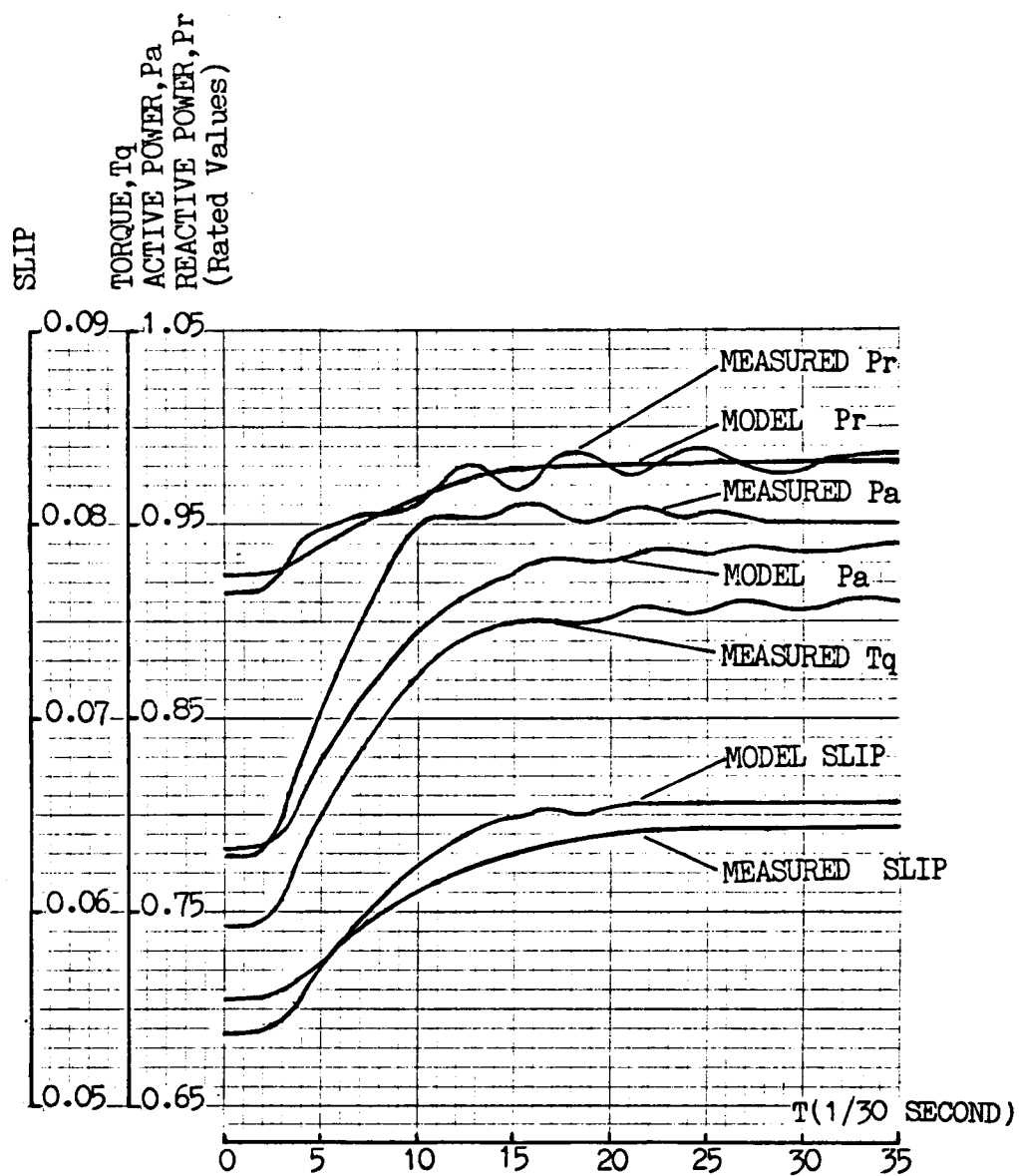


Figure 11. Power and Slip Variation for Load Torque Change from 74% to 91%.

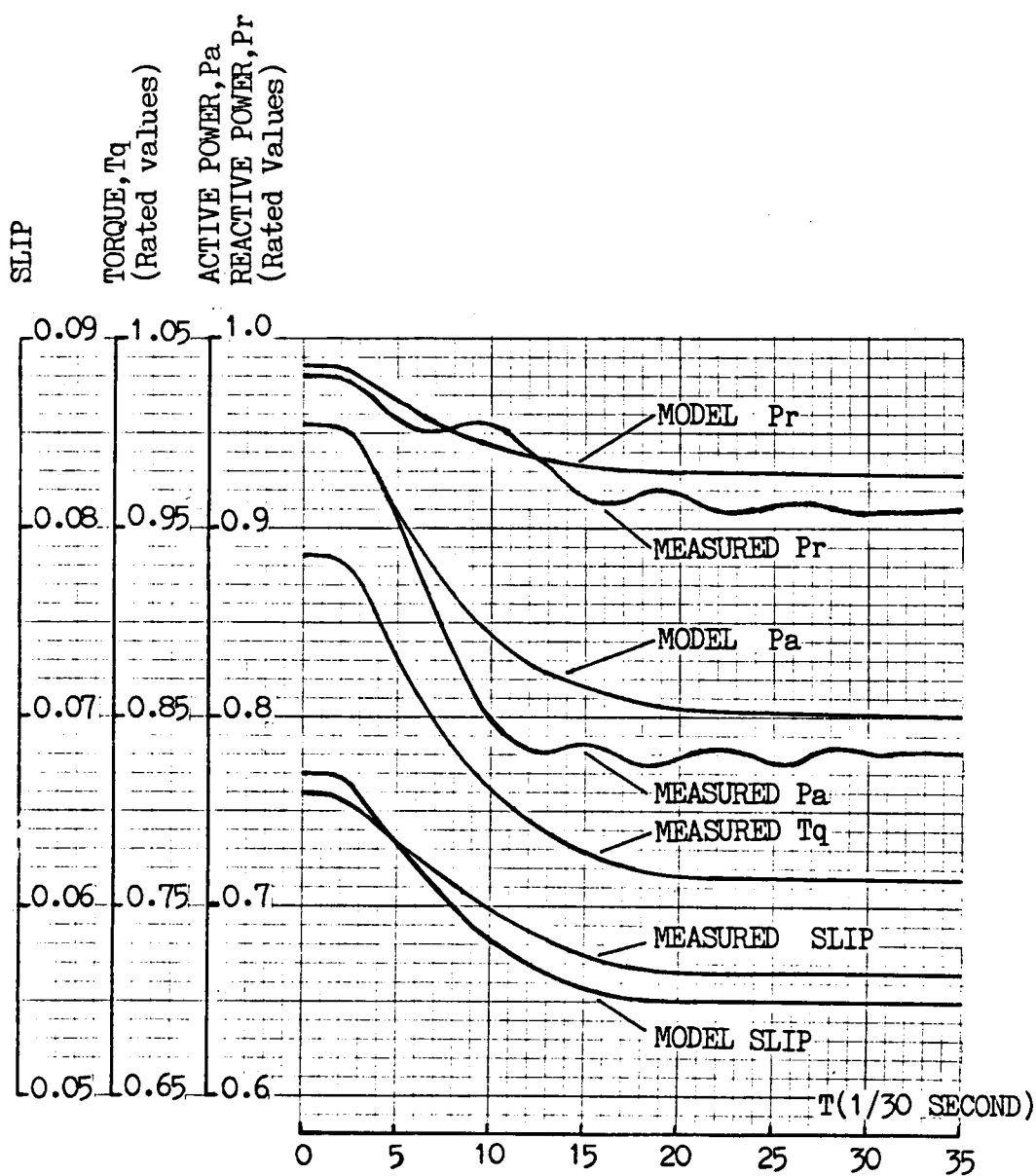


Figure 12. Power and Slip Variation for Load Torque Change from 94% to 76%.

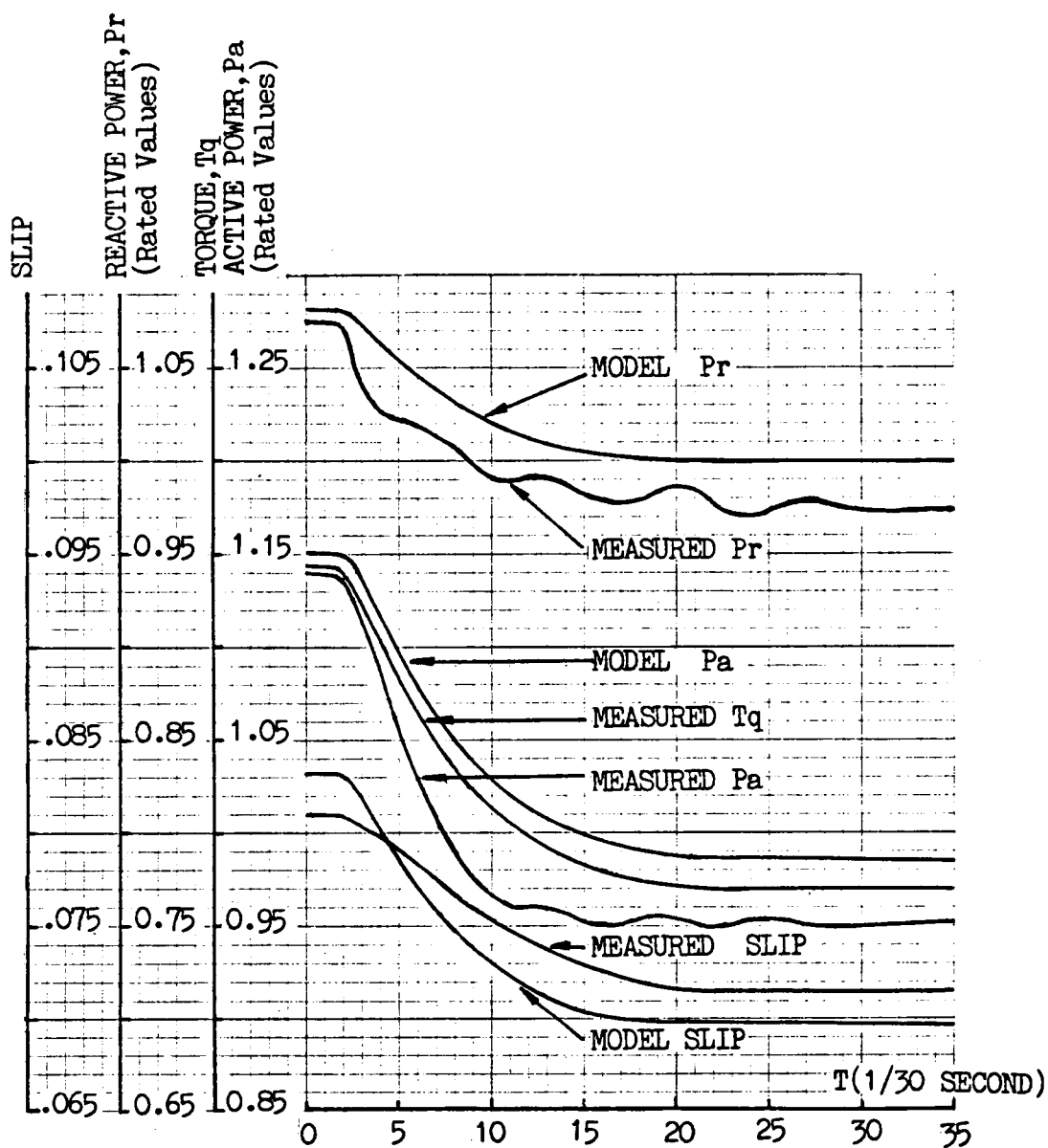


Figure 13. Power and Slip Variation for Load Torque Change from 114% to 96%.

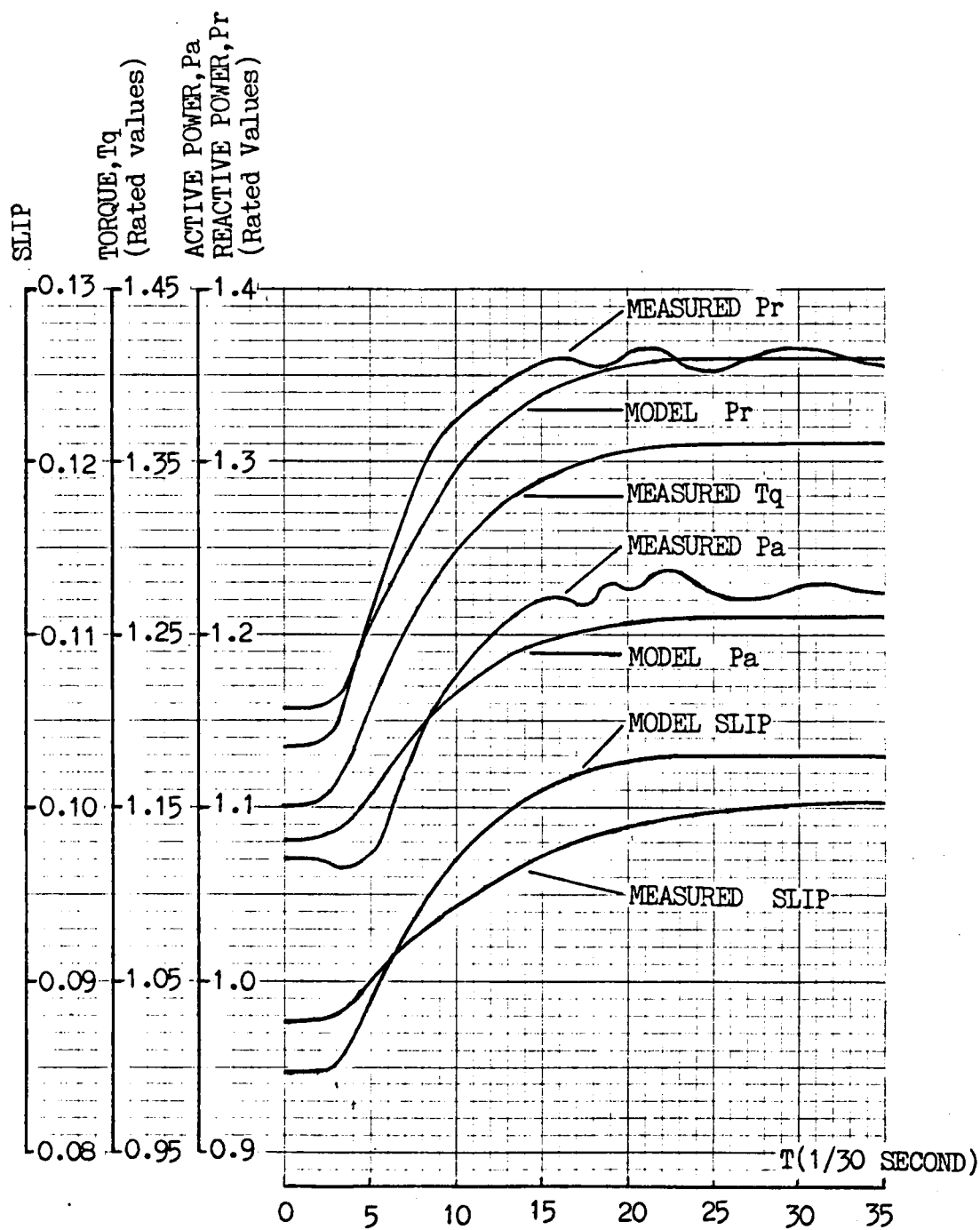


Figure 14. Power and Slip Variation for Load Torque Change from 115% to 136%.

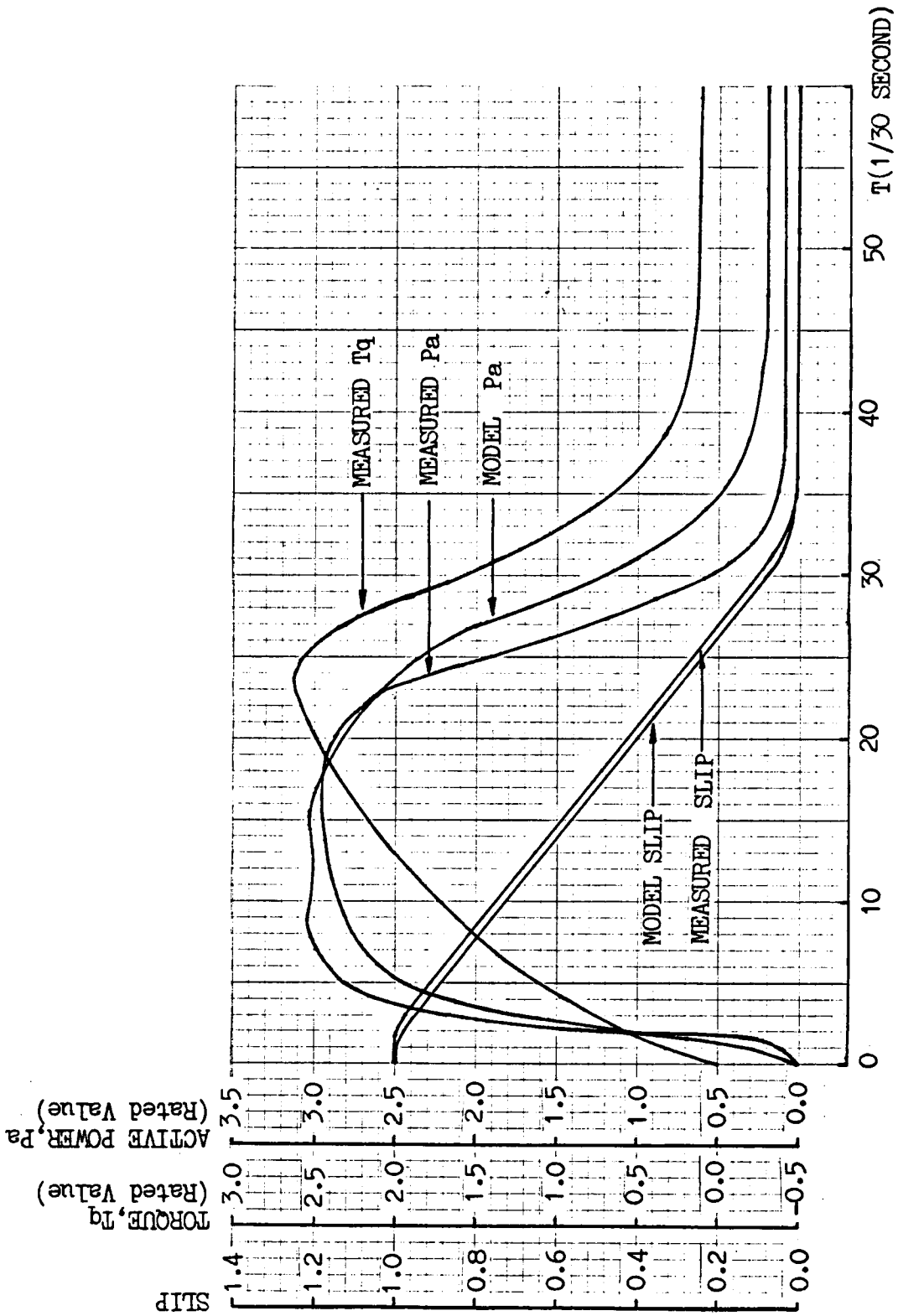


Figure 15. Power and Slip Variation of the Motor Started from Rest.

Since there is no reactive power transducer available in the power laboratory, there is no directly measured experimental data about the reactive power delivered to the motor, but since the line voltage and phase current were measured, then the reactive power for steady state can be calculated by equation 76 as follows

$$P_r = \sqrt{(V \times I)^2 \times 3 - P_a^2} \quad (76)$$

where P_r is the reactive power delivered to the motor, P_a is the active power delivered to the motor, V is the line voltage, and I is the phase current.

The reactive power calculated by equation 76 for transients is not accurate. The faster the transient is the worse the result becomes. The reason for this will explain later.

Even though the transient reactive power is invalid, it is still plotted, as calculated by equation 76, in Figure 9 through Figure 14. These curves are accurate in the steady state regions and can be used to verify the steady state values of the model. Although they are not accurate during transients, they still give some sense about the behavior of the reactive power. Considering these factors, after examining Figure 9 through 14, we can say that the model yields accurate information about active power, slip and reactive power.

Now we explain the reason why equation 76 is not accurate for transient studies. The time responses of the voltage, current and active power transducers are not the same. Suppose the current transducer is slower than voltage and active power transducers by ΔT seconds. For a correct calculation of reactive power, equation 76 should be

$$P_r(t) = \sqrt{(V(t) \times I(t+\Delta T))^2 \times 3 - P_a^2(t)}$$

Since there is no easy way to know what ΔT is, and the computer can not account for the equipment delays, then if these delays are significant, errors are inevitably introduced by equation 76 into the calculation of the reactive power transient response. For steady state, the current, voltage and active power change slowly enough that the measurement delays can be neglected.

Figure 9 gives the variations of active and reactive power delivered to the motor and its slip as the load torque changes from 30% to 56% of its rated value. This is a very light load representation. The differences between measured and predicted values are less than 5% of the rated values.

Figure 10 gives the variations of active and reactive power delivered to the motor and its slip as the load torque changes from 53% to 75% of its rated value. This is

a light load representation. The differences between measured and predicated values are less than 5% of the rated values.

Figure 11 gives the variations of active and reactive power delivered to the motor and its slip as load torque changes from 74% to 91% of its rated value. This is a normal load representation of the motor. The maximum errors are less than 5% of rated values.

Figure 12 shows the variations of active and reactive power delivered to the motor and its slip as the load torque changes from 94% to 76% of its rated value. This is a normal load representation of the motor. The errors are around 5% of its rated values.

Figure 13 gives the variations of active and reactive power delivered to the motor and its slip as the load torque changes from 114% to 96% of its rated value. This is a heavy load representation. The errors are less than 5% of its rated values.

Figure 14 gives a very heavy overload representation of the induction motor. It shows the variations of active and reactive power delivered to the motor and its slip as the load torque changes from 115% to 136% of rated value. The errors are less than 5% of its rated values.

Figure 15 gives the data about the motor starting from rest. The active power and slip predicted by the fifth

order model are quite close to the data collected by experiment. The errors between them are bigger than previous ones. The reason is that during starting the damping factor and rotational inertia play an important role. The damping factor is assumed be a constant for all speeds; this assumption is reasonable for small speed changes. For large changes in speed, the damping factor needs to be considered as a function of the speed. Also, since the measured values have a wide range, such as the active power changing from zero to three times rated value, the absolute errors of the transducer will increase. But for most engineering applications concerning the starting of the motor in a power system, the model can meet the required accuracy.

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

Examining Figure 9 through 15 and considering the measurement errors caused by the meter dynamics and the deviations of the motor parameters from actual values, the model developed for an induction motor has good accuracy, and can be used for most engineering applications. Its errors will be less than 5% of the rated values.

Since the model assumes the motor works on the linear part of the magnetic saturation curve, a heavy overload will cause additional error due to saturation. The result of an overload is equivalent to reducing the self-inductance and mutual-inductance. Since the saturation point is chosen by the manufacturer and will have different values for different manufacturers, it is difficult to define the exact values. From the data taken on the experimental motor, the model gives satisfactory results upto 150% of the rated values.

RECOMMENDATIONS

Since the model is sensitive to some parameters, such as self-inductance and mutual-inductance, the accuracy of the motor model is largely dependent on accurate knowledge of the parameters of the motor. Unfortunately, it is not easy to precisely determine the parameter values for the motor. There are many methods available to measure parameter values by experiment, and most of them can yield the required accuracy. If more accuracy is needed, an estimation method can be used as presented in Chapter 4 of this work. An interesting subject for power system application is to extend the estimation algorithm for one motor to a group motors; then the model can more precisely describe the true aggregate load for motors in the power system.

LIST OF REFERENCES

LIST OF REFERENCES

1. J.Cathey. Transient Load Model of an Induction Motor. University Microfilms International. 1972.
2. D.C.White and H.H.Woodson. Electromechanical Energy Conversion. New York: John Wiley and Sons. 1959.
3. Arthur Gelb. Applied Optimal Estimation. The M.I.T.Press. Third Printing. 1977.
4. Ralph Blach. A Computerized Data Acquisition and Analysis System for a Power Laboratory. M.S. Thesis. The University of Tennessee, Knoxville. December 1979.
5. K.P.R.Sastry and R.E.Burridge. Investigation of a Reduced Order Model for Induction Machine Dynamic Studies. IEEE. Trans. Power App. Syst. Pas-95. No.3, May/June 1976.
6. Digital Microcomputer Handbook. Digital Equipment Corporation. Maynard, Massachusetta. 1976.
7. Vincent Del Toro. Electromechanical Devices for Energy Conversion and Control System. Pentice-Hall Inc. 1968.
8. A-D Converter Manual. Datel Systems Inc. Maynard, Massachusetta. 1977.
9. RT-11 Advanced Programmer's Guide. Digital Equipment Corporation. Maynard, Massachusettes. 1978.
10. Richards Gill. and Tan Qwent. Simplified Models for Induction Machine Transients under Balanced and Unbalanced Condition. IEEE. Trans. on

Industry Applications. IA-17 Jan./Feb.1981.

11. N.Gunaratnam and D.W.Novotny. The Effects of Neglecting stator Transients in Induction Machine Modeling. IEEE. Trans. Power App. Syst. Pas-99. Nov./Dec. 1980.
12. P.C.Krause, F.Nozari, T.L.Skrarenina, and D.W.Olive. The Theory of Neglecting Stator Transients. IEEE. Trans. Power App. Syst. Pas-98, Jan./Feb. 1979.
13. T.L.Skvarenina and P.C.Krause. Accuracy of a Reduced Order Model of Induction Machine Stability Studies. IEEE. Power. Syst. Pas-98 July/Aug. 1979.
14. J.D.Birdwell, G.G.Jones, and D.A.Reed. Sycopack Documentation. Electrical Engineering Department, The University of Tennessee, Knoxville. 1980.

APPENDIXES

APPENDIX A

COMPUTER HARDWARE DOCUMENTATION

The configuration of the computer system is shown in Figure 16. This system is completely self-contained with total program development capability. The computer was built by Blach and was described in his thesis [4]. It was extensively modified for this project, and the present configuration is described here.

In order to execute complex calculations with ease in a reasonable time, a floating point processor was added in the CPU board.

A Soroc Technology video terminal communicates with the processor through the Heathkit H11-5 serial interface, which has been configured to respond to addresses 177560 through 177564. The receiver interrupt vector is located at address 60, and the transmitter interrupt vector is located at address 64. The baud rate is set at 9600.

A Datel Corporation A-D converter is used to interface the analog transducers to the digital computer. The analog converter has 16 A-D channels expandable to 32 channels, an internal or external clock interrupt capability, and individual or auto-increment channel scan. The A-D converter needs +15 volt and -15 volt power supplies. Slot 3 of backplane 2 has been wired to supply these values.

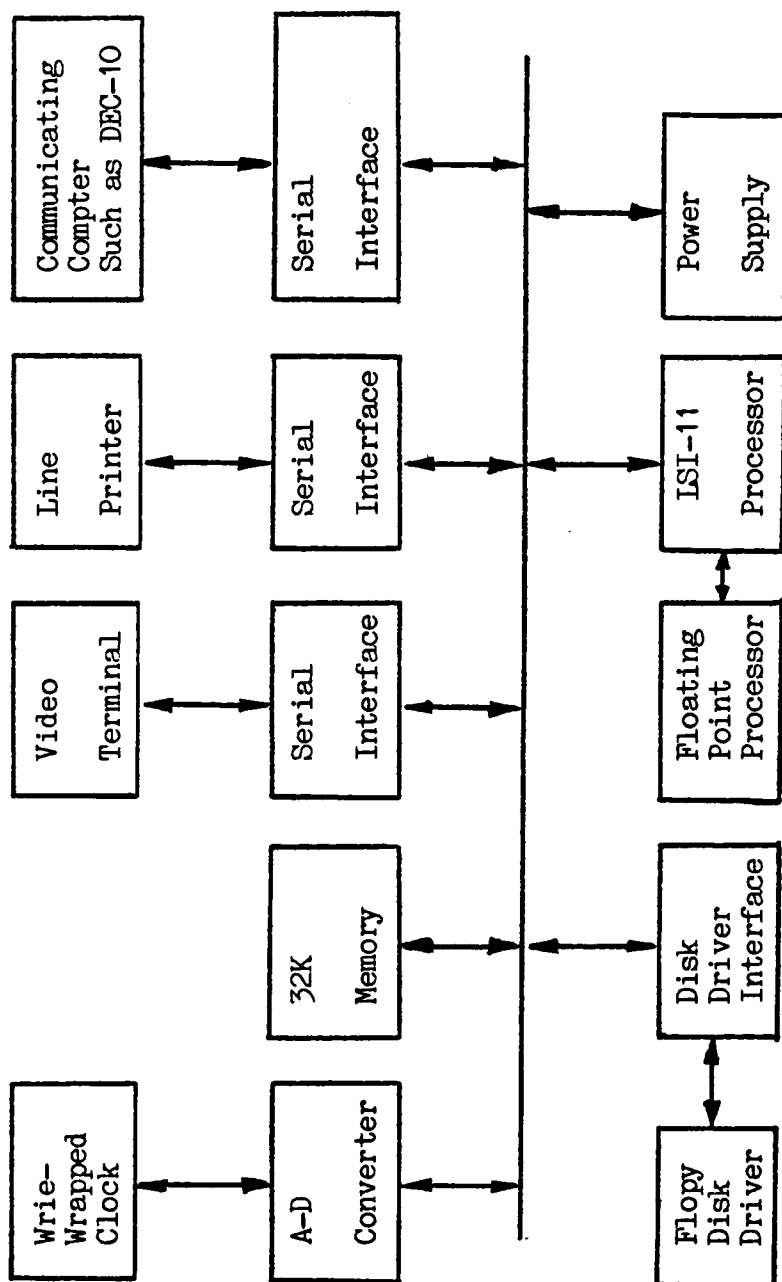


Figure 16. Computer System Architecture.

The status addresses of the A-D converter are 170400 and 170402. The addresses of the interrupt vectors for end of single channel conversion and end of a channel scan are 360 and 364, respectively. The clock interrupt vector address is 370. On the A-D converter board there is room for two D-A converters and 4 digital outputs. The addresses for these outputs are 170404 and 170406. The D-A converters are not implemented on this version of the board. The detailed specification is in reference [8].

An Advanced Electronics Design AED 6200 LS floppy disk drive is used as the primary system mass storage device. With the appropriate software this disk drive will read from and write on diskettes written in any double density format. The process of writing timing marks on a diskette is called formatting. The AED disk drive can write timing marks on a diskette which correspond to any formatting scheme. Using this capability, a double density format which stores over 600,000 bytes on a diskette has been used. The addresses of the AED 6200 LS range from 164000 through 164006; the interrupt vector address is 170.

A line printer, model 810, manufactured by Texas Instruments, is used as a hard copy device. It is linked to the processor through a modified DLV-11 serial interface board which has standard addresses for the registers and the interrupt vector. They are 177514, 177516 and 200,

respectively. The baud rate is set at 2400.

A DLV-11 communication interface board is mounted in slot 7 of backplane 1. The address of the registers and interrupt vectors are 176500 through 176506, 300 and 304, respectively. In order to be compatible with the hardwired line to the UTCC DEC-10, the baud rate is set at 2400. It can link to the DEC-10, the IPAL PDP11/40 or other devices by switching to the corresponding connector.

A 32 K memory board manufactured by Intel is used; its model is 05-0848. Since there are 4 K of memory on the CPU board, and the top 4 K of memory is used as device addresses, only the range from 4 K to 28 K of the memory has been enabled by switches on the board.

Two bus extender boards, model MLSI-TEV-40340, are used to link backplane 1 and backplane 2. In order to transmit proper daisy-chained bus signals, the board labeled ONE must be in backplane 1 slot 8 and the board labeled TWO must be in backplane 2 slot 1.

A bus terminator, model M9400, is used to match the bus characteristic impedance. It is in the last slot in backplane 2.

All these boards are mounted on two H9270 backplanes. The assembly diagram of the computer system is shown in Figure 17 and described in Table 9. The device register and interrupt vector addresses are shown in Table 10.

Back-plane 1

D	C	B	A
Slot 1		Slot 2	
Slot 4		Slot 3	
Slot 5		Slot 6	
Slot 8		Slot 7	

Back-plane 2

D	C	B	A
Slot 1		Slot 2	
Slot 4		Slot 3	
Slot 5		Slot 6	
Slot 8		Slot 7	

Figure 17. The Back-planes and Their Slot Assignment.

Table 9

The Back-plane Assignment

Device	The Place Mounted On	
	Back-plane No.	The Slot No.
LSI-11 Processor	Back-plane 1	Slot 1 and 2
Disk Driver Interface	Back-plane 1	Slot 3 and 4
32 K Memory Board	Back-plane 1	Slot 5
Video Terminal Interface	Back-plane 1	Slot 6
Communicate Interface	Back-plane 1	Slot 7
Bus Extender Board 1	Back-plane 1	Slot 8
Bus Extender Board 2	Back-plane 2	Slot 1
Line Printer Interface	Back-plane 2	Slot 2
A-D Converter	Back-plane 2	Slot 3 and 4
Time Clock for A-D	Back-plane 2	Slot 5 and 6
Spare	Back-plane 2	Slot 7
Terminator Board	Back-plane 2	Slot 8

Table 10

The Device Register and Interrupt Vector Addresses

Board Name	Status Address	Interrupt Vector Address
Disk Driver	164000,164002	170, 172
Interface	164004,164006	174, 176
Video Terminal	177560,177562	60, 62
Interface	177564,177566	64, 66
Communicating	176500,176502	300, 302
Interface	176504,176506	304, 306
Line Printer	177514	200
Interface	177516	202
A-D Converter	170400,170402	370,372,360
Interface	170404,170406	362,364,366

In order to start up the LSI-11 processor after turning on the power, two bus signals are needed. They are called BDCOK and BHALT. When the BHALT signal is active (logical 0), the processor will halt normal program execution and invoke ODT microcode. This can be used to diagnose user programs under ODT. When BDCOK changes from logical 0 to logical 1, it will trigger the processor to execute the power restart microcode.

These two signals are generated in a wire-wrapped small home-made card. It also contains a line frequency clock. Each time the AC line voltage crosses zero and goes positive, the clock generates a signal which is linked to bus signal BEVENT. When BEVENT is active, the processor is interrupted and enters a service routine via vector address 100, which contains a line frequency clock service routine address for the operating system. The clock has the highest priority on the bus, so that when starting the processor this line frequency clock must be turned off; otherwise, the operating system can not be started. Also, when running all the programs in this work, turn the clock off. The circuit diagram is shown in Figure 18.

The computer control panel and its output panel are shown in Figure 19. The procedure to start the computer is described as follows:

Turn the power supply switch to the ON position; turn

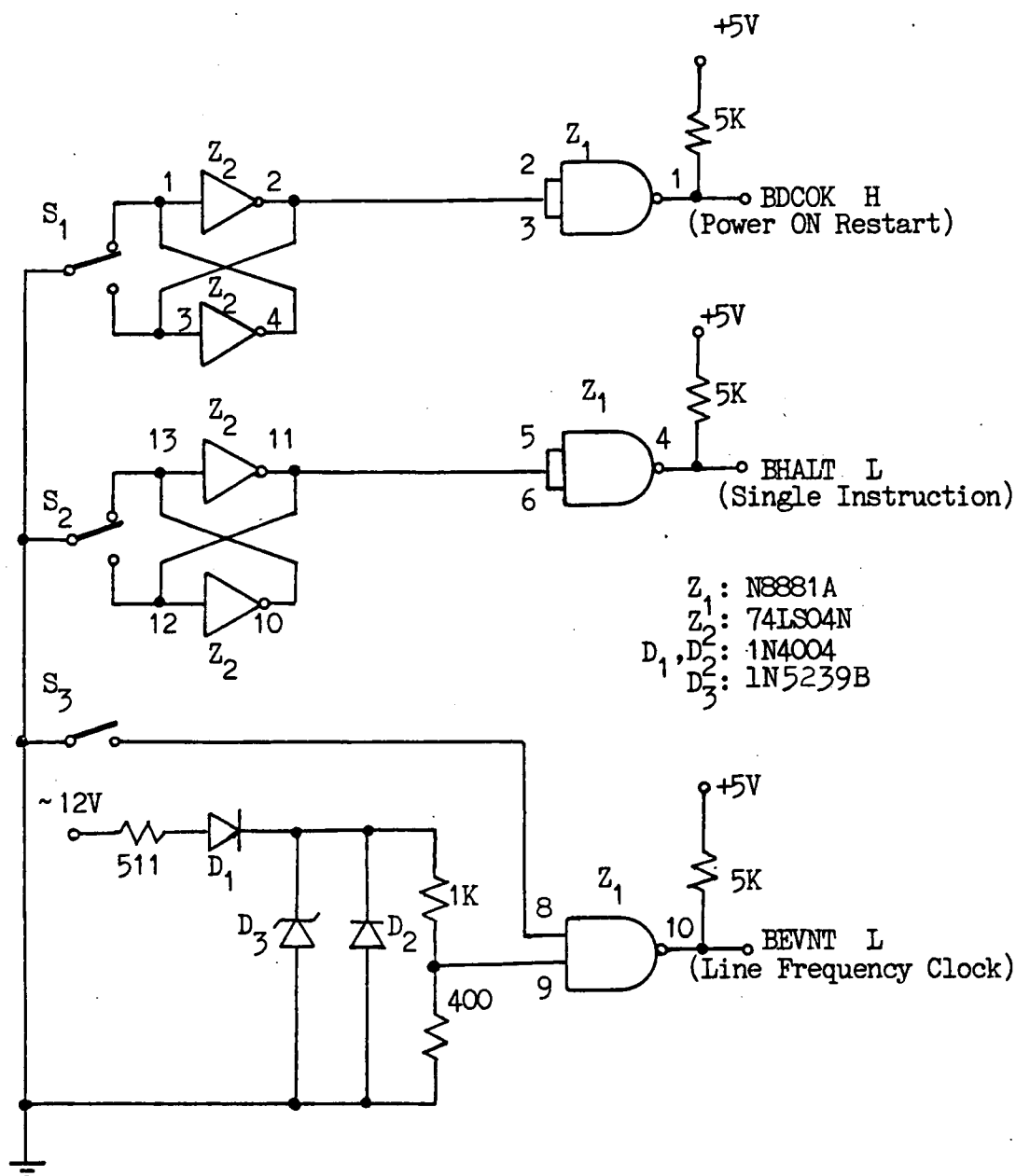
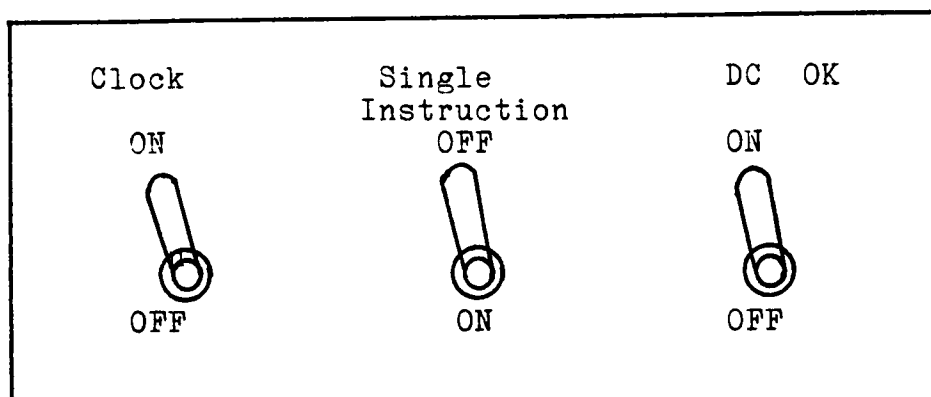


Figure 18. Power Restart, Single Instruction and Line Frequency Clock Circuit.

CONTROL PANEL



OUTPUT PANEL

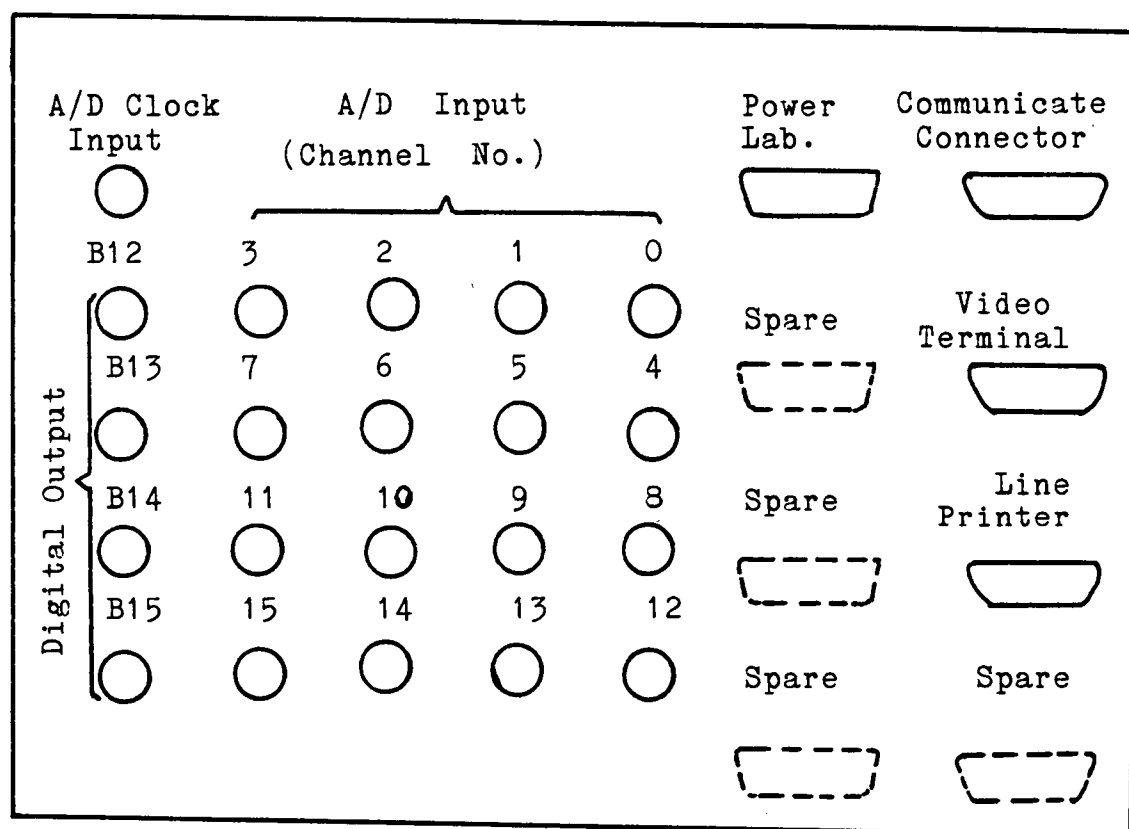


Figure 19. Diagrams of the Control Panel and Output Panel

the time clock switch to the OFF position; turn the single instruction switch to the OFF position, and turn the DC OK switch to the ON position. Afterwards turn on all the peripheral devices including the disk drive. Push the INITIAL button on the disk drive to the UP position and release it; it will return to the low position. Then type a capital G on the video terminal. After a couple of seconds, the operating system will be active and some system message will appear on the screen. At this point the computer is ready to execute any commands. When turning off the computer, follow the reverse procedure. In order to avoid destroying the data on the disk, make sure that the floppy disk has been removed before turning off the disk drive.

APPENDIX B

SOFTWARE DESCRIPTION

In this appendix all softwares in both the LSI-11 and the DEC-10 computers are presented; the detailed ways to run these programs are described. The programs cover the collection of experimental data and simulation of the fifth order model.

SOFTWARE FOR DATA COLLECTION ON LSI-11 COMPUTER

The program for collecting data is named MODE11.MAC, and is written in PDP-11 assembly language. The main difference between program MODE11 and MAIN5 in reference [4] is that the three new commands ZR, BN and BR were added for collecting transient response of the motor. MODE11 runs under the RT-11 operating system and can collect, store and display the data. MODE11 uses 13 commands to communicate with the operator through the video terminal. The detailed operating procedures are as follows:

When the program MODE11 is started, it prints out the message:

```
FILE NAME FOR OUTPUT
*
```

where the * is the prompt from the operating system command string interpreter. After the prompt *, the program

expects a valid RT-11 file name. A valid name is either of the form

FILENM.EXT=<CR>

or

<CR>

where the <CR> represents the carriage return, FILENM stands for the six character name of the file in which the data will be stored on the disk, and the EXT is the extension of the file name. In the first case the file name has been opened and is ready to store data. In the second case no file is opened, and no data will be stored. If there is any error in the file name specification typed by the operator, the program will output an error message. After succeeding in opening the file on disk, the program outputs a ? on the console to notify the operator that the program is waiting for one of the 13 commands to be typed. An explanation of the 13 commands follows.

The ZR command collects and stores the deviation of the transducer from zero reference point. After execution, the program outputs another ? to wait for another input command. An example is

? ZR<CR>
?

The nice feature of ZR is that it can be used to enlarge the measurable range. For instance, the torque

transducer has a zero knob to adjust the output value for zero input signal. Suppose the zero knob is adjusted so that the transducer output value is equal to negative full scale for zero input signal. After using ZR command the positive measurable range is doubled.

The BN command performs two functions: One is to turn on or off the transistor switches to generate the step load function; another is to collect the transient response of the motor to this load change at the same time. The load of the motor first steps up until all the transistor switches are turned on, and then steps down until all the transistor switches are turned off.

A crystal clock, whose frequency is 30 Hz, is used to generate the sampling time signal. Each $1/30$ second, the clock interrupts the computer and triggers an A-D conversion. The A-D converter works in auto-increment scan mode. It collects sampling data for each channel 64 times at its maximum rate, then averages them and stores the average values. The computer then stops the A-D converter and waits for another clock interrupt signal.

Since the motor will reach steady state within 1 second, the time duration for each step is chosen as 3 seconds. Since the sample time is $1/30$ second, the number of data points for each step is 90, and for 5 transistor switches, 11 steps can be generated. The total data set

contains 990 data points. Figure 20 shows the typical load torque generated by BN. The amplitude of the load torque for each step can be changed by changing the load resistance R_1 and R_2 of the transistor switch shown in Figure 6. An example is

```
? BN<CR>
?
```

Note: After typing BN command, the operator needs to wait nearly half minute to allow the computer to collect data before getting any response on the console.

The IR command has a similar function to BN except the program does not turn on or off the transistor switches. An example is

```
? IR<CR>
?
```

Note: Nearly half a minute elapses before getting any response from the console.

The C command converts, stores and displays a line of data. The data is one sample of all data channels. An example is

```
? C<CR>
LINE  VOLTS    AMPS    WATTS    RPM    TORQUE
0001   0223.    0008.89  0895.    1180.    0027.46
? C<CR>
LINE  VOLTS    AMPS    WATTS    RPM    TORQUE
0002   0223.    0008.89  0890.    1180.    0027.50
?
```

The AV command is the same as C except it gets the data value from the average of 64 samples. An example is

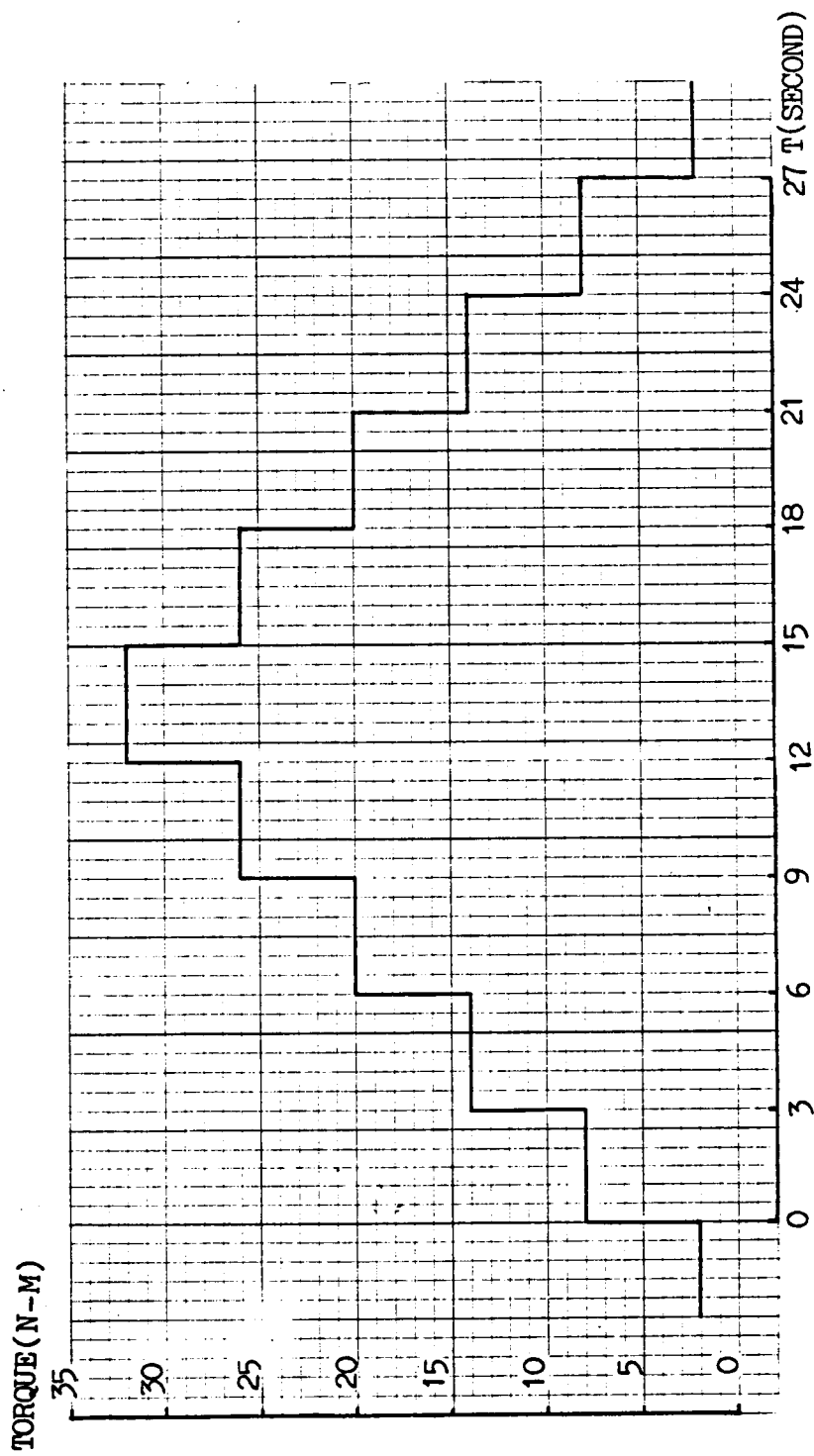


Figure 20. The Typical Load Torque Generated by Command BN.

```

? AV<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0003    0223.    0009.00  0910.    1180.    0026.99
? AV<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0004    0223.    0008.98  0894.    1179.    0027.80
?

```

The R command recollects a specified line of data, and stores and displays it. The R command has the form of

```
R=NNN<CR>
```

where NNN is a one to three decimal number which stands for the line number to be recollected. An example is

```

? R=10<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0010    233.    0013.69  4547.    1133.    0489.96
? R<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0010    223.    0013.52  4540.    1133.    0489.20
?

```

Note: The second R command has no line number specification. In this case the default line number is equal to the line number last collected.

The AR command is the same as R except it collects data by averaging 64 samples. An example is

```

? C <CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0012    0224.    0008.89  0896.    1180.    0027.46
? AR<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0012    0224.    0008.85  0893.    1180.    0027.40
? AR=6<CR>
LINE    VOLTS    AMPS    WATTS    RPM    TORQUE
0006    0223.    0008.80  0850.    1181.    0027.31

```

If a line number for which no data exists is specified by an R or AR command, the program will output an error

message.

The T command outputs the specified lines of data on the screen. The format of the T command is

T=NNN,NNN,. . .,NNN<CR>

where NNN is a one to three decimal number.

When NNN is omitted, the default number is the one last specified. An example is

```
? T=4,0,123<CR>
LINE   VOLTS   AMPS   WATTS   RPM   TORQUE
0004   0222.   0008.82  0397.   1180.  0027.32
LINE   VOLTS   AMPS   WATTS   RPM   TORQUE
0000   0223.   0008.80  0873.   1181.  0027.00
NO SUCH LINE
?
```

The A command displays all the lines of data taken. This command is very useful for reviewing data before storing it on disk. An example is

```
? A<CR>
LINE   VOLTS   AMPS   WATTS   RPM   TORQUE
0000   0231.   0012.21  3882.   1137.  0421.35
0001   0231.   0012.10  3880.   1137.  0421.12
0002   0231.   0012.70  3910.   1136.  0422.01
?
```

The S command stores the data on the specified file in disk. If the S command is successful, the program will restart. If a failure occurs, an error message will be written on the console. Usually an incorrect FILENM.EXT or no FILENM.EXT will be the cause of the failure. In the event of failure, the operator should use the FD command to check the file name. If no file name is opened, then a

file name should be specified by using FS command. If the file name is present but no an = follows the file name, a file by that name is present on the disk and opened for reading, but not for storing. Therefore a FS command is needed to respecify the file name. An example is

```
? S<CR>
DATA STORED IN FILE FILENM.EXT=
FILE NAME FOR OUTPUT
*
```

The FS command is used to specify an output device and file name for storing the collected data. When it is executed, it calls the command string interpreter of the operating system, which writes a * on the console. a valid RT-11 output file name should be typed in. The general form of the file name specification is

OUTFIL.EXT[N]=INPFIL.EXT

Note: The file name before the = sign is for storing data and the N, if it is a positive decimal number, is used to specify the block number in disk for storing. If the N is equal to -1, then the maximum available blocks will be assigned to that file name. If the N is omitted, then only half of the maximum available blocks will be assigned to that file. The file name after a = sign is for reading data. If the part of the line up through the = sign is omitted, then that file name is used for reading data. Therefore when specifying the file name it is very

important to put an = symbol or a < symbol after the file name. More information can be found in reference [9]. An example is

```
? FS<CR>
FILE NAME FOR OUTPUT
* MOTOR.DAT<CR>
? CSI-F-FILE NOT FOUND
* MOTOR.DAT=
?
```

The FD command is used to examine the output file name. If there is one present, then executing the FD command will display the file name on the console. If there is no output file opened, FD responds with a line feed. An example is

```
? FD<CR>
MOTOR.DAT=
?
FS<CR>
FILE NAME FOR OUTPUT
*<CR>
? FD<CR>

? FS<CR>
FILE NAME FOR OUTPUT
* MOTOR1.DAT=,CR>
? FD<CR>
MOTOR1.DAT=
?
```

The Q command is used to end the program and return to the monitor. Note: Execution of this command immediately returns control of the video terminal to the monitor; any data collected will not be stored. An example is

```
? Q<CR>
.
```

where the . is the prompt of the monitor. A summary of

commands for MODE11 is listed in Table 11.

SOFTWARE FOR CONVERTING THE DATA FORMAT TO ASCII

The data collected by the program MODE11 is stored in a binary format, and must be converted to ASCII format in order to print it out or transmit it to another computer. When the program is run, it asks the operator to type in the file name for data input and output. The file name has the same form as previously mentioned. An example is

```
* OUTFIL.EXT[N]=INPFIL.EXT<CR>
```

where the * is the prompt of the executing program. The default of the extension for input is DAT and for output is MAC. A control C will kill the program and give control back to the monitor of the operating system.

SOFTWARE FOR COMMUNICATION

The software for communication with the DEC-10 or another device is contained in the program COMUT.MAC. It controls the bidirectional transfer of ASCII data across an asynchronous serial interface to another device, such as the DEC-10 computer. When COMUT runs, the program will prompt with a * on the console. Then the operator needs to enter a valid RT-11 file name, which has the same form as mentioned in the previous section. To receive a file from another computer, the file specification is

Table 11

Summary of Commands for the Program MODE11

Command	Meaning
ZR	Collect and store the zero deviation values of the transducers
BN	Turn on or off step by step the transistor switches, collect and store data
IR	Collect and store 999 data sets
C	Take in another data set
AV	Take and average the incoming data, which is an average of 64 data samples
R=NNN	Repeat the conversion specified by the NNN or the last one if no NNN specified
AR=NNN	The same as R except the data which is equal to an average value
T=NNN,...,NNN	Display the data sets specified by NNN
A	Display all of the collected data
S	Store all collected data
Q	Exit the MODE11
FD	Display the file specification on the video terminal
FS	Specify a new file name for store data

OUTFIL.EXT[N]=

To transmit a file to the other computer, the file specification is

INPFIL.EXT

After the program is successful in transmitting the file, the operator can use the RT-11 console as if it were a video terminal directly attached to the other computer. As an example, to transfer the file MOTOR.DAT to the DEC-10, the following exchange occurs.

.R COMUT<CR>	(RT-11)
<u>FILE NAME</u>	(RT-11)
*MOTOR.DAT<CR>	(RT-11)
<CR>	(DEC-10)
.R PIP<CR>	(DEC-10)
*MOTOR1.DAT=TTY:<CR>	(DEC-10)
^P	(RT-11)
.	
.	(TRANSMIT FILE)
.	
^Z	(DEC-10)
<u>*</u>	(DEC-10)
.	(DEC-10)
<u>STOP TRANSMITTING</u>	(RT-11)
.	(DEC-10)

where ^p is the control P, and the underlined words are produced by the computer.

After typing ^p, the file named MOTOR.DAT in the LSI-11 will be transferred to the DEC-10 under the filename MOTOR1.DAT. During the transfer, the transmitted characters will be echoed on the video terminal after the DEC-10 receives them. When the LSI-11 program encounters the last character in the file, it will transmit a control

Z to the DEC-10 to inform PIP that the end of file is present. The DEC-10 then closes the file MOTOR1.DAT. There might be a sizable delay between the last character being sent and the control Z being echoed on the terminal. This is due to the null characters that RT-11 uses to fill up the last block between the last character and the last word of the block. The delay should be ignored. Since the maximum number of characters per second which can be transmitted to the DEC-10 is 30, a time delay occurs after each character transmitted by COMUT to meet that limit. During transmission COMUT monitors the busy signal from the DEC-10. If it receives a bell signal, COMUT stops transmitting for 3 seconds. It also monitors for control S and control Q characters and responds properly.

For transmitting data to other devices, such as the PDP-11/40, the limit on the number of characters per second is much higher. The operator may enter a control A, which will halve the delay time which occurs after each transmitted character. This procedure can be continued as desired until the delay time generated by COMUT equals zero. On the other hand if more delay is desired after each transmitted character, the operator may enter a control B to double the delay time. The maximum delay time is 2 seconds.

After finishing the transfer of a file to the DEC-10,

the operator can use the DEC-10 computer to process the data. In order to transfer control of the video terminal back to RT-11 the operator should type a control N after logging off the DEC-10, and COMUT will ask for a new file name to be transferred. A control C will then stop COMUT and return the terminal to RT-11 monitor status. An example of receiving a file named ABC1.FOR from the DEC-10 is

<u>.R</u> COMUT	(RT-11)
<u>FILE NAME</u>	(RT-11)
*ABC1.FOR=<CR>	(RT-11)
<CR>	(DEC-10)
.TYPE ABC1.FOR^V	(DEC-10)
.	(RECEIVE FILE)
.	
^W	(RT-11)
<u>STOP RECEIVE, FILE STORED</u>	(RT-11)
.K	(DEC-10)
.DISC	(DEC-10)
<u>BYE!</u>	(DEC-10)
^N	(RT-11)
<u>FILE NAME</u>	(RT-11)
*^C	(RT-11)
.	(RT-11)

where ^V, ^W, ^N and ^C are control V, control W, control N and control C; the underlined characters are generated by the computers.

The control V is an internal command to COMUT; after receiving it the program transmits a carriage return to the DEC-10 to end the TYPE command and starts receiving the file from the DEC-10. During the transfer of the file, all the received characters are echoed on the terminal. When the DEC-10 finishes the transfer, the operator either types

in a control W, or inserts a control W at the end of the file before transferring it in order to close the received file on the RT-11 disk. A summary of the control words is listed in Table 12.

SOFTWARE FOR SIMULATING THE MODEL ON DEC-10

The program that simulates the fifth order model is written in the FORTRAN language, and is named MODE32.FOR. The input variable for the model is the load torque of the induction motor; therefore, before running MODE32, a file called FOR07.DAT must be prepared, in which the torque data are stored. In our case, the torque data are collected by experiment using the LSI-11. When MODE32 runs, it reads the FOR07.DAT file to get the input variables, and also asks the operator to input the parameters of the induction motor from the terminal. After that the program calculates the initial values before calling subroutine DQSYK to solve the nonlinear differential equations. Since the motor-generator unit will reach steady state within 1 second for one step load torque change, the model is simulated for 1.3 seconds on the computer. The output time interval is 1/30 second in order to match the experimental sampling time. All the programs are listed in the appendix C, and the subroutines used to solve differential equations are described in Reference 14.

Table 12

Summary of Control Words for the Program COMUT

Control Words	Functions
Control N	Terminate communication with the other computer; return control of console to LSI-11 and restart the COMUT
Control P	Start transmitting a file to the other computer
Control Z	Close the transmitted file on the other computer under PIP
Control V	Start receiving a file from the other computer
Control W	Close the received file in LSI-11 and restart the COMUT
Control A	Half the delay time which occurs after each character is transmitted
Control B	Double the delay time which occurs after each character is transmitted
Control C	Exit from COMUT

APPENDIX C

SOFTWARE LISTINGS

```

; !***** PROGRAM NAME: MODE11.MAC *****!
; !
; !     THIS PROGRAM IS USED TO COLLECT DATA
; ! ABOUT AN INDUCTION MOTOR IN THE POWER LAB.
; ! FIVE KINDS OF DATA WILL BE COLLECTED.
; ! THEY ARE A.C. LINE VOLTAGE, PHASE CURRENT,
; ! TOTAL INPUT ACTIVE POWER, MOTOR SPEED, AND
; ! TORQUE OF THE MOTOR GENERALED. THE SIGNALS
; ! ARE ANALOG , AND THEY ARE LINKED TO A/D
; ! CHANNEL 0-4. IT USES 13 COMMANDS TO CONTROL
; ! THE LOAD OF THE A.C. MOTOR AND COLLECT THE 5
; ! DATA. THEY ARE AS FOLLOWS:
; !
; !     ZR: COLLECT AND STORE THE DEVIATION
; !     VALUE OF THE TRANSDUCERS
; !     BN: OUTPUT DIGITAL STEPS TO CONTROL
; !     THE LOAD OF THE A.C MOTOR AND COLLECT
; !     DATA AT THE SAME TIME
; !     IR: IT HAS THE SAME FUNCTION AS "BN" EX-
; !     ECPT IT DOES NOT OUTPUT DIGITAL SIGNAL
; !     C: TAKE IN ONE SET OF DATA
; !     AV: TAKE IN ONE SET OF DATA BY AVERAGE
; !     VALUES
; !     R: RECOLLECTA SET OF EXISTED DATA
; !     AR: THE SAME AS "R" EXECPT BY AVERAGE
; !     T: DISPLAY SPECIFIED SETS OF DATA
; !     A: DISPLAY ALL EXISTING DATA SETS
; !     S: STORE ALL EXISTING DATA SETS
; !     Q: EXIT FROM THIS PROGRAM "MODE11"
; !     FD: DISPLAY THE FILE NAME SPECIFIED
; !     FS: ASSIGN A FILE NAME FOR STORE DATA
; !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
; TITLE POWER SYSTEMS LAB
; COMMA    =54
; EQ       =75
; GAIN     =0           ; THE A/D BOARD HAS A FIXED GAIN
; POINT    =56         ; DECIMAL POINT
; SPACE    =40         ; SPACE
; CR       =15         ; CARRAGE RETURN
; LF       =12         ; LINE FEED
; LETA     =101
; LETZ     =132

```

```

MODE0 =0           ;SINGLE CONVERSION
MODE1 =1           ;SCAN CONVERSION MODE
MODE2 =2           ;PACER CONVERSION MODE
MODE3 =3           ;EXTERNAL PACER
AUTOON=4           ;AUTO CHANNEL INCREMENT
AUTOOF=0           ;AUTO CHANNEL OFF
PACINT =10         ;PACER INTERRUPT ENABLE
ADDO =0            ;VOLTAGE CHANNEL
CDSTAT =170400
GNCHAD =170402
ADDATA =170402
DOUT=170404
.MCALL .CLOSE,.EXIT,.REGDEF,.DEVICE
.MCALL .CSIGEN,.TTYIN,.TTYOUT,.WRITW,.READW,.PRINT
.CSECT MAIN
MAIN: CLR @#LINE      ;START THE STORAGE POINTER AT ZERO
      .DEVICE #DEVARE,#DEVLST
      JSR PC,FILE      ;GO AND ASK FOR THE FILE NAME
INB:  MOV #ADDLST+2,ADDLST
      MOV #-1,ADDLST+2
      MOV #INBUF,R1     ;INPUT A STRING OF CHARACTERS
      CLR R2
      MOVB #'?',R0      ;PUT A QUESTION MARK IN R0
      .TTYOUT           ;SEND IT TO THE TTY
IN:   .TTYIN           ;FROM THE KEYBOARD AND STORE IN
      MOVB R0,(R1)+     ;STORE THE STRING ON THE LIST
      CMPB R0,#LF       ;IS THE LAST CHARACTER A LINE FEED?
      BNE IN           ;BACK IF NOT
      JSR PC,PARSE
      BCS PARERR        ;IF THE PARSE FAILED, FIND OUT WHY
      CLR R0
      MOV #JMPTAB,R1
      MOV @#NUMCMD,R2   ;THERE ARE NINE COMMANDS
      CLR R0            ;START AT THE BEGINNING OF THE LIST
$CMDLP: CMP CMDREG,CMDLST(R0) ;CHECK THE CMDREG AGAINST
                        ; EACH MEMBER OF THE CMDLST
      BEQ $GC ;IF WE HAVE A MATCH FIND OUT WHERE IT IS
      TST (R0)+         ;NEXT COMMAND
      SOB R2,$CMDLP
      .PRINT #BADLET
      BR INB
PARERR: CMP @#COMWRD,#1 ;SEE IF THE FIRST LETTER WAS A CR
      BEQ INB          ;SOFT ERROR, GO AND TRY AGAIN
      .PRINT #PIE      ;PRINT PARSER INPUT ERROR
      BR INB
$GC:  ADD R0,R1         ;ADD THE OFFSETT TO THE BASE ADDRESS
      MOV (R1),R1       ;MOVE THE JUMP ADDRESS TO R1
      JMP (R1)          ;R1 POINTS TO THE ADDRESS
                        ; OF THE SERVICE ROUTINE

```

```

*****
;*
;* THIS ROUTINE CALLS THE DATA STORE SUBROUTINE
;* THE DATA IS STORED IN AN UNFORMATTED FORMS AND MUST
;* BE REFORMATTED BY ASC2.SAV BEFORE IT IS PRINTED OR SENT
;* TO THE DEC-10/IBM 370.
;*
*****

```

```

ST:      JSR PC,STORE
        BCC $STM      ;IF NO ERROR OCCURED THEN RESTART
        .PRINT #WARN   ;AN ERROR OCCURED, TELL THE OPERATOR
        JMP INB        ;DO NOT CLEAR DATA PIONTER
$STM:    .PRINT #STMESS ;TELL THE OPERATOR THE DATA STORED
        BR MAIN ;GO BACK AND RESTART THE PROGRAM

```

```

*****
;*
;* THIS ROUTINE IS USED TO OUTPUT THE DIGITAL SIGNAL TO
;* CONTROL ELECTRICAL SWITCHES AND AT THE SAME TIME COLLECT
;* DATA,WHICH WILL CONVERT LINES OF DATA. A LINE OF DATA
;* CONSISTS OF TAKING FIVE DATA POINTS AND STORING THEM IN
;* MEMORY. LINE TELLS HOW MANY SETS OF FIVE DATA POINTS
;* WE HAVE TAKEN. ALN IS WORD THAT TELLS CNVRT WHICH
;* LINE OF DATA THE PROGRAM IS ON.
;*
*****

```

```

BN1:    MOV      #BAVER,370      ;PREPARE INTRRUPT FOR A/D
        MOV      #340,372
        CLR      FLAG           ;FLAG FOR SWITCH ON OR OFF
        CLR      LINE          ;START LINE 0
        MOV      #PATERN,TEMP4  ;SWITCH STEP PATTERN
2$:      MOV      @TEMP4,DOUT     ;OUTPUT SWITCH
        MOV      #90.,TEMP5     ;90 SAMPLES EACH SWITCH STEP
        MTPS     #0             ;ALLOW INTERRUPT
        MOV      #PACINT!MODE3,CDSTAT ;SET A/D START CHAN.
1$:      BR       .              ;WAIT A/D FINISH
        MOV      LINE,ALN       ;STORE DATA
        JSR      PC,$ENTRY
        INC      LINE           ;BUMP A LINE
        DEC      TEMP5          ;IS 90 DATA FOR ONE STEP
        BNE      1$            ;NO,CONTINUE
        INC      FLAG           ;GOTO NEXT STEP
        CMP      FLAG,#11.      ;DOES FINISH?
        BEQ      CN11           ;YES
        CMP      FLAG,#6        ;STEP DOWN OR UP?
        BLO      3$             ;DOWN
        SUB      #2,TEMP4
        BR       2$

```

```

3$:      ADD      #2,TEMP4      ;UP
        BR       2$

;*****
;*
;* THIS ROUTINE CONVERTS ONE LINE OF DATA , IT TAKES *
;* 64 DATA SETS THEN AVERAGES THEM, ALSO IT WILL STORE *
;* AND DISPLAY THIS DATA SET *
;*
;*****
CN:      MOV LINE,ALN      ;PUT THE LINE NUMBER TO
        ; BE CONVERTED INTO ALN
        JSR PC,CNVRT
        INC @#LINE      ;INCREMENT THE LINE NUMBER
        .PRINT #HEADER  ;PRINT THE LINE HEADER
        JSR PC,TYPE      ;DISPLAY THE PAST CONVERSION
CN11:    CLR      CDSTAT    ;NO MORE A/D
        JMP      INB      ;GO OUT

;*****
;*
;* THIS ROUTINE IS USED TO COMPENSATE THE ZERO DEVA- *
;* TION VALUES OF THE TRANSDUCERS, IT COLLECTS AND STORE*
;* THESE VALUES AT ZERO INPUT SIGNALS, THE FINAL OUTPUT *
;* VALUES ARE THE DIFFERENCE OF THE REAL INPUT & "ZERO" *
;* INPUT SIGNAL *
;*
;*****

ZERO:    MOV      #BAVER,370      ;PREPARE A/D INTERRUPT
        MOV      #340,372
        MTPS     #340
        MOV      #MODE3!PACINT,CDSTAT;ALLOW AUTO-INCRESE INT.
        MTPS     #0
        BR       .              ;WAIT A/D FINISH
        MOV      #TEMP,R0        ;GET ADDR. FOR READ DATA
        MOV      #CON1,R1        ;ZERO VALUE STORE ADDR.
        MOV      #5,R2
1$:      MOV      (R0)+,(R1)+      ;STORE THEM
        SOB      R2,1$           ;FINISHED?
        BR       CN11           ;YES, GO OUT

;*****
;*
;* THIS ROUTINE IS USED TO CONVERT 999 LINES OF *
;* DATA.EACH LINE OF DATA IS AVERAGE VALUES OF 64 DATA *
;*
;*****

INER:    CLR      LINE          ;START AT LINE 0
        MOV      #340,372      ;PREPARE A/D INTERRUPT

```

```

        MOV     #BAVER,370
        MTPS    #0
        MOV     #MODE3!PACINT,CDSTAT      ;ALLOW A/D TO WORK
1$:      BR      .                        ;WAIT A/D FINISH
        MOV     LINE,ALN                  ;STORE A LINE DATA
        JSR     PC,$ENTRY
        INC     LINE                      ;BUMP A LINE
        CMP     LINE,#999.                ;IS FINISHED?
        BLO     1$                        ;NO
        BR      CN11                      ;YES,GO OUT

```

```

;*****
;*
;* FI AND FD ARE THE FILE SPEC AND FILE DISPLAY PROGRAMS *
;*
;*****

```

```

FI:      JSR PC,FILE      ;ASK FOR THE FILE SPEC
        JMP INB
FD:      JSR PC,FILPRT    ;PRINT THE FILE SPEC
        JMP INB
QU:      .EXIT

```

```

;*****
;* AV IS ROUTINE THAT AVERAGES DATA. IT IS A SIMPLE AVERAG *
;* ROUTINE THAT USES INTERRUPT TO DIVIDE A SECOND INTO 256 *
;* EQUAL PARTS AND KEEP A RUNNING SUM OF THE FIVE A-D DATA *
;* CHANNELS. AT THE END OF THE 1 SECOND SUMS ARE DIVIDED BY*
;* 256. $ENTRY IS AN ENTRY POINT INTO CNVRT THAT STORE THE *
;* DATA IN MEMORY.
;*****

```

```

AV:      JSR PC,AVER      ;AVER IS THE AVERAGING ROUTINE
        MOV     @#LINE,@#ALN          ;PUT THE CONVSERION NUMBER
        JSR PC,@#$ENTRY              ;IN ALN
        INC     LINE
        .PRINT  #HEADER
        JSR     PC,TYPE
        JMP     INB
FILPRT:  .PRINT  #FB
        RTS     PC
FILE:    .PRINT  #FILNM
        MOV     SP,SPSAVE
        .CSIGEN #DEVBUF,#DFTEXT,#0,#FB
        MOV     SPSAVE,SP
        RTS     PC
SPSAVE:  0
PARSE:   CLR     R3
        CLR     TENNUM
        MOV     #-1,R2              ;THESE TWO REGISTERS ARE POINTERS

```

```

        CLR CMDREG          ;THE CMDREG MUST BE ZERO TO START
STATE1: INC R2              ;MOVE THE CHARACTER POINTER ONE BYTE
        CMPB INBUF(R2),#SPACE ;SKIP OVER ALL LEADING SPACES
        BEQ STATE1          ;IF SPACES WAIT TILL NO SPACES
        CMPB INBUF(R2),#CR   ;SEE IF A CARRIAGE RETURN
        BNE $443            ;IF IT IS, TELL THE CALLER
        JMP @#CREXIT
$443:   JSR PC,LETTER        ;CHECK TO SEE IF A CAPITAL
        ; LETTER BETWEEN A AND Z
        BCC $444            ;IF NOT TELL THE OPERATOR
        JMP @#ERROR
$444:   MOVB INBUF(R2),CMDREG ;STORE IN THE CMDREG LOW BYTE
STATE2: INC R2              ;NEXT BYTE PLEASE,YUMMY
        CMPB INBUF(R2),#SPACE ;IF THE NEXT BYTE IS A SPACE
        ;GO AND WAIT TILL SOMETHING ELSE
        BEQ $1              ;GO AND WAIT
        CMPB INBUF(R2),#EQ   ;SEE IF AN EQUAL SIGN
        BEQ $SPA
        CMPB INBUF(R2),#CR   ;SEE IF A 'CR'
        BNE $12.3           ;BRANCH TO EXIT ON EQUAL
        JMP $EXIT1
$12.3:  JSR PC,LETTER        ;SEE IF A LETTER
        BCS ERROR
        MOV INBUF(R2),CMDREG+1
$1:     INC R2              ;NEXT CHARACTER PLEASE
        CMPB #SPACE,INBUF(R2) ;WE WANT TO SKIP SPACES
        BEQ $1              ;SKIP SPACES
        CMPB #EQ,INBUF(R2)   ;IS IT AN EQUAL SIGN?
        BEQ $SPA
        CMPB #CR,INBUF(R2)
        BEQ $EXIT1
        BNE ERROR          ;NO MUST BE AN ERROR
$SPA:   INC R2              ;NEXT CHARACTER
        CMPB INBUF(R2),#SPACE ;SKIP SPACES
        BEQ $SPA
        CMPB INBUF(R2),#COMMA ;CHAECK FOR COMMAS
        BEQ CMA
        CMPB INBUF(R2),#CR   ; CHECK FOR CARRIAGE RETURN
        BEQ $EXIT          ;GO TO EXIT ROUTINE
N1:     INC @#TENNUM        ;WE HAVE ENCOUNTERED ONE NUMBER
        JSR PC,CHECK        ;SEE IF THE CHARACTER IS A NUMBER
        BCS ERROR          ;IF THE CARRY IS SET, NOT A NUMBER
        MOVB $NUM,R3        ;PUT THE NUMBER IN R3
N2:     INC R2
        JSR PC,CHECK        ;SEE IF THE CHARACTER IS A NUMBER
        BCS $ECOM1         ;IF NOT CHECK FOR COMMAS AND 'CR'
        MOV R3,R0           ;SET UP FOR MULTIPLY
        MUL #10.,R0         ;MULTIPLY THE FIRST DIGIT BY TEN
        CLR R3
        MOVB $NUM,R3        ;NUMBER GOES TO R3

```

```

      ADD R1,R3          ;R3=10.*N1+N2
      INC R2             ;NEXT CHARACTER
N3:   JSR PC,CHECK
      BCS $ECOM1         ;CHECK FOR CARRIAGE RETURN OR COMMA
      MOV R3,R0
      MUL #10.,R1
      CLR R3
      MOVB $NUM,R3
      ADD R1,R3          ;R3=((10.*N1)+N2)*10.+N3
$ECOM: INC R2             ;NEXT CHARACTER
$ECOM1: CMPB INBUF(R2),#COMMA ;CHECK FOR COMMA
      BEQ $STLST
      CMPB INBUF(R2),#CR   ;CHECK FOR CARRIAGE RETURN
      BEQ $EXIT
      BR ERROR
$STLST: CMP #9.,@TENNUM ;DO WE HAVE MORE THAN TEN NUMBERS?
      BEQ $EXIT ;IF WE DO GO AND EXIT, AND FORGET THE
                  ;REST OF THE PARSE. WE HAVE NO ROOM
                  ;FOR ANY MORE NUMBERS ANYWAY

      MOV ADDLST,R0
      MOV R3,(R0)+        ;THE FIRST WORD OF THE ADDLST
                        ; IS THE POINTER INTO THE LIST
      MOV R0,ADDLST      ;STORE THE POINTER
      BR CMA
CMA:  INC R2             ;NEXT CHARACTER
      CMPB INBUF(R2),#COMMA
      BEQ CMA            ;LOOP ON COMMAS
      CMPB INBUF(R2),#CR
      BEQ $EXIT1
      JSR PC,CHECK       ;SEE IF A NUMBER
      BCC N1             ;IF A NUMBER GO TO N1
      BR ERROR
$EXIT1: MOV ADDLST,R0
      MOV #-1,(R0)+
      RTS PC
$EXIT:  MOV ADDLST,R0
      MOV R3,(R0)+        ;STORE THE LAST NUMBER ON THE LIST
      MOV #-1,(R0)+      ;TERMINATE THE LIST
      RTS PC

```

```

;*****
;*
;*  THE ROUTINES BELOW SUPPORT THE PARSER
;*  ERROR   TELL THE CALLER SOMETHING
;*           WAS WRONG WITH THE COMMAND STRING
;*  CREXIT  TELLS THE CALLER THAT THE FIRST
;*           WAS A CARRIAGE RETURN
;*  CHECK   TELLS THE PARSER IF THE SYMBOL IS A
;*           NUMBER AND IF SO RETURNS WITH IT
;*  LETTER  TELLS THE PARSER IF THE SYMBOL IS A
;*

```

```

; *          CAPITAL LETTER          *
; *                                     *
; *****

CREXIT: MOV #1,COMWRD ;TELL THE CALLER THE FIRST CHARACTER
          ;WAS A CARRIAGE RETURN
          SEC
          RTS PC          ;RETURN TO CALLER
ERROR:   MOV #2,COMWRD ;TELL THE CALLER WHAT THE ERROR IS
          SEC          ;SET THE ERROR BIT
          RTS PC

; *****
; *                                     *
; *          ALL REFERENCES TO ERROR ARE THROUGH          *
; *          JUMPS AND BRANCHES                          *
; *          A RTS RETURNS TO THE MAIN PROGRAM          *
; *                                     *
; *****

LETTER:  CMPB #LETA,INBUF(R2)      ;IF LESS THAN NOT A LETTER
          BGT $LER
          CMPB #LETZ,INBUF(R2)    ;IF GREATER THAN NO A LETTER
          BLT $LER
          CLC          ;NO ERROR, TELL THE CALLER
          RTS PC
$LER:    SEC
          RTS PC
CHECK:   MOV R4,-(SP)      ;PUSH R4 ON TO THE STACK AND SAVE IT
          CLR R4          ;NO BYTE ADD OR SUBTRACT OPERATIONS
          MOVB INBUF(R2),R4    ;PUT THE CHARACTER IN R4
          SUB #60,R4        ;MAKE THE ASCII CHARACTER A NUMBER
          BMI $E          ;IF THE RESULT WAS NEGATIVE,
          ; IT WAS NOT A NUMBER
          CMPB R4,#9.      ;IF R4 IS GREATER THAN 9,
          ; IT WAS NOT A NUMBER
          BGT $E
          MOV R4,$NUM      ;PUT THE NUMBER IN ITS STORE LOCATION
          MOV (SP)+,R4     ;RESTORE R4
          CLC          ;NO ERROR WAS DETECTED
          RTS PC
$E:      MOV (SP)+,R4
          SEC          ;AN ERROR WAS DETECTED
          RTS PC
CNVRT:   MOV #TEMP,R3      ;PUT THE TEMPORY STORAGE BUFFER IN R3
          MOV #5,R4        ;WE WILL SCAN 5 CHANNELS
          MOV #AUTOON!MODE1,CDSTAT ;AUTO INCREMENT AND SCAN
          MOV #GAIN!ADD0,@#GNCHAD ;START SCAN AT CHANNEL ZERO
          MOV @#ADDATA,@#TEMP ;START THE CONVERSION
TS:      TST @#CDSTAT      ;SEE IF DONE

```

```

        BPL TS          ; IF NOT LOOP
        MOV @#ADDATA,(R3)+      ; STORE THE CONVERSIONS
        SOB R4,TS          ; FIVE CONVERSIONS
$ENTRY: MOV #ARRAY,R4 ; PUT ADDRESS OF THE DATA ARRAY IN R4
        MOV @#ALN,R0 ; PUT THE LINE NUMBER IN THE MULTIPLIER
        MUL #12.,R0          ; RETURNS THE ADDRESS IN R0
        ADD R1,R4 ; PRODUCES THE STORE ADDRESS OF THE DATA
        MOV @#ALN,(R4)+ ; STORE THE LINE NUMBER FOR LATER
        MOV #5,R0
        MOV #TEMP,R3
        MOV      #CON1,R2
ARR:     SUB      (R2)+,(R3)
        MOV (R3)+,(R4)+ ; WRITE THE DATA INTO THE ARRAY
        SOB R0,ARR      ; LOOP TILL DONE
        RTS PC

;*****
;*
;*      R0 COUNTS HOW MANY TIMES WE HAVE INTERRUPTED,*
;*      TEMP IS USED AS A DOUBLE PRECISION STORAGE *
;*      AREA. AFTER THE ROUTINE HAS FINISHED,      *
;*      THE NUMBERS ARE DIVIDED BY 64 AND COMPRESSED*
;*      TO FIVE STORAGE LOCATIONS STARTING AT      *
;*      TEMP.                                       *
;*
;*****

AVER:   MOV      #64., R0      ; CLEAR THE INTERRUPT COUNTER
        MOV #370,R5          ; R5 POINTS TO THE INTERRUPT
        MOV #AVSER,(R5)+; VECTOR. PUT IN THE LOCATION
                                ; OF THE SERVICE ROUTINE.
        MOV #340,(R5) ; THE ROUTINE WILL BE UNINTERRUPTABLE
        MOV #10.,R5      ; EACH OF THE 5 A-D CHANNELS
                                ; REQUIRES TWO WORDS
        MOV #TEMP2,R4 ; LOCATIONS TO BE CLEARED START AT TEMP
$CLLP:  CLR (R4)+          ; CLEAR EACH ONE OF THE TEN LOCATION
        SOB R5,$CLLP
        MTPS #340          ; MAKE THE ROUTINE INTERRUPT PROOF
        MOV #PACINT!MODE3,@#CDSTAT ; TURN ON THE INTERRUPT
        MTPS #0            ; INTERRUPT, AND AWAY WE GO!
HANG:   BR HANG           ; GURGLE, GURGLE, GURGLE!
        RTS      PC
AVSER:  CLR CDSTAT
        DEC R0            ; INCREMENT THE INTERRUPT COUNT
        BEQ      $DONE1 ; REACH 64 TIMES
        JSR      PC,$CNVRT ; GO TO COLLECT A SET DATA
        MOV      #PACINT!MODE3,CDSTAT ; NO MORE A/D CONVERT
        RTI
$CNVRT: MOV      #TEMP2,R3      ; GET STORE ADDR.
        MOV      #5,R4         ; COLLECT 5 DATA

```

```

        MOV     #AUTOON!MODE1,CDSTAT      ;START CONVERT
        MOV     #GAIN!ADD0,GNCHAD
        MOV     ADDATA,R1                 ;BEGIN TO CONVERT
$TS:    TST     CDSTAT ;FINISHED?
        BPL     $TS                       ;NO
        MOV     @#ADDATA,R1 ;STORE THE CONVERSIONS
        ADD     R1,(R3)+                  ;ADD IN THE LOWER PART
        ADC     (R3)+                    ;ADD IN THE CARRY BIT
        TST     R1                       ;IS MINUS?
        BPL     1$                       ;NO , PLUS
        ADD     #177777,-2(R3) ;SIGN EXTENSION
1$:     SOB     R4,$TS                    ;FIVE CONVERSIONS
        RTS     PC

```

```

;*****
;*
;* THIS IS A SUBROUTINE FOR CONVERTING A SET OF DATA
;* THE VALUE IS EQUAL TO THE AVERAGE VALUE OF
;* 64 DATA POINTS
;*
;*****

```

```

BAVER:  MOV     #64.,R0 ;SET A/D TIMES TO 64
        MOV     #10.,R5 ;CLEAR STORE AREA
        MOV     #TEMP2,R4
1$:     CLR     (R4)+
        SOB     R5,1$
2$:     JSR     PC,$CNVRT                ;GO TO CONVERT
        DEC     R0                      ;DECREASE A/D TIMES
        BNE     2$
        MOV     #PACINT!MODE3,CDSTAT    ;REACH 64 TIMES
                                           ;NO MORE CONVERT

```

```

;*****
;*
;* THE $DONE ROUTINE TAKES THE DOUBLE PRECISION NUMBERS
;* IN TEMP2 AND DIVIDES THEM BY 256 DECIMAL. THE ROUTINE
;* STARTS BY TAKING THE MOST SIGNIIFICANT PART
;* OF THE LAST CONVERSION AND PUTTING IT INTO R0 AND THE
;* LEAST SIGNIFICANT PART IN R1,THEN WORKING DOWN THE
;* LIST BACKWARDS, STORING THE RESULTS IN TEMP AS IT GOES
;*
;*****

```

```

$DONE1: ADD     #2,(SP)
        MTPS    #0                      ;MAKE THE ROUTINE INTERRUPTABLE
        MOV     #TEMP2+20.,R5 ;PUT THE TOP OF TEMP2 PLUS ONE
        ;WORD IN R5. THE AUTO DECREMENT WILL POINT IT

```

```

                                ; TO THE CORRECT LOCATION.
MOV #TEMP+10.,R4 ; PUT THE TOP OF TEMP PLUS ONE WORD
                                ; IN R4.
MOV #5,R3 ; WE HAVE FIVE CONVERSIONS TO DIVIDE BY 64.
$SHIFT: MOV -(R5),R0 ; PUT THE MOST SIGNIFICANT PART IN R0
MOV -(R5),R1 ; PUT THE LEAST SIGNIFICANT PART IN R1
ASHC #-6,R0 ; DIVIDE THE PAIR BY 256.
MOV R1,-(R4) ; PUT IN TEMP
SOB R3,$SHIFT ; LOOP TILL ALL FIVE VALUES ARE DONE
RTI ; RETURN FROM WHENCE WE CAME
TYPE: MOV R0,-(SP)
MOV LINE,R0 ; SEE IF THE NUMBER IS OUT OF BOUNDS
DEC R0 ; BACK UP THE LINE POINTER
CMP ALN,R0
BLE $14 ; A GOOD NUMBER
JMP FATAL
$14: MOV #ARRAY,R5
MOV @#ALN,R0
MUL #12.,R0
ADD R1,R5 ; CALCULATES THE POSITION OF THE VALUES
                                ; IN THE ARRAY
TST (R5)+ ; ADD 2 TWO R5 FOR THE PGM
MOV #LIST,R4 ;
MOV #CONST,R3 ;
MOV @#ALN,R0 ; CONVERT THE LINE NUMBER INTO ASCII
JSR PC,DECAS
JSR PC,SPA
AVOLTS: JSR PC,MLT ; CONVERT THE VOLTAGE TO AN ASCII LIST
JSR PC,NORML
JSR PC,DECAS
MOVB #POINT,(R4)+ ; PUT DECIMAL POINT ON THE STACK
JSR PC,SPA ; ADD FOUR SPACES TO THE STACK
ACUR: JSR PC,MLT ; CONVERT THE CURET TO AN ASCII LIST
ASHC #-1,R0
JSR PC,NORML
JSR PC,DECAS ; THE CURRENT BY TWO
MOVB #POINT,(R4)+ ; PUT A DECIMAL POINT ON THE STACK
JSR PC,FRAC
JSR PC,SPA
AWAT: JSR PC,MLT
JSR PC,NORML
JSR PC,DECAS
MOVB #POINT,(R4)+
JSR PC,SPA
ASPEED: JSR PC,MLT
JSR PC,NORML
JSR PC,DECAS
MOVB #POINT,(R4)+
JSR PC,SPA
ATOR: JSR PC,MLT

```

```

        JSR PC,NORML
        JSR PC,DECAS
        MOVB #POINT,(R4)+
        JSR PC,FRAC
        CLRB (R4)+
        .PRINT #LIST      ;PRINT THE NUMBERS
$12:    MOV (SP)+,R0
        RTS PC
FATAL:  .PRINT #NSL
        BR $12
MLT:    MOV (R5)+,R0
        MUL #5,R0
        MOV R1,R0 ;R0 NOW CONTAINS LEAST SIGNIFICAT PART
        MUL (R3)+,R0 ;PUT IN THE CONVERSION SCONSTANT
        RTS PC
NORML:  ASHC #6,R0 ;PUT INTERGER PART OF THE NUMBER IN R0
        ;THE FRACTIONAL PART GOES IN R1
        RTS PC
DECAS:  MOV R1,-(SP)
        MOV R5,-(SP) ;PUSH R0 AND R5 ONTO THE STACK
        TST R0
        BPL 1$
        MOVB #'-, -1(R4)
        NEG R0
1$:     MOV #THT0,R5 ;MOVE THE ADDRESS OF THE CONVERSION
        ; TABLE TO R5
        MOV #4,R2 ;NUMBER OF LOOPS
        MOV R0,R1 ;PUT THE INTEGER PART IN R1
        CLR R0 ;CLEAR OUT THE UPPER PARTS OF THE NUMBER
LP:     DIV (R5)+,R0 ;DIVIDE OUT THE THOUSANDS, ECT
        ADD #60,R0 ;MAKE AN ASCII CHARACTER
        MOVB R0,(R4)+ ;STORE ON THE LIST
        CLR R0 ;CLEAR OUT THE UPPER PART OF THE NUMBER
        SOB R2, LP
        MOV (SP)+,R5
        MOV (SP)+,R1
        RTS PC ;RETURN FROM WHINCE WE CAME
SPA:    MOV R2,-(SP) ;SAVE R2
        MOV #4,R2 ;FOUR SPACES
SPA1:   MOVB #SPACE,(R4)+ ;FOUR SPACES ON THE LIST
        SOB R2,SPA1
        MOV (SP)+,R2
        RTS PC
FRAC:   MOV R1,R0 ;MOVE THE FRACTIONAL PART TO R0
        CLC
        ROR R0 ;MAKE R0 A POSITIVE NUMBER
        MUL #2000.,R0 ;CONVERT FRACTION TO A WHOLE NUMBER
        MOV R0,R1 ;PUT THE WHOLE NUMBER IN R1
        CLR R0 ;NOTHING IN THE LEAST PART
        DIV #100.,R0 ;DIVIDE OUT THE HUNDREDS

```

```

      ADD #60,R0          ;MAKE AN ASCII CHARACTER
      MOVB R0,(R4)+       ;STUFF ON THE LIST
      CLR R0
      DIV #10.,R0         ;GET NEXT PART
      ADD #60,R0
      MOVB R0,(R4)+
      RTS PC
AVRE:  TST @#LINE
      BEQ ERR
      MOV #ADDLST+2,R0
      MOV (R0),R0         ;R0 NOW POINTS TO THE CONVERSION
                          ; TO BE REPEATED
      BMI $AVDC           ;IF R0 IS NEGATIVE,
                          ; REPEAT THE LAST CONVERSION
      CMP R0,LINE         ;SEE IF NUMBER ACTUALLY EXISTS
      BGE ERR             ;TELL THE OPERATOR SO
      MOV R0,ALN
$AVCNV: JSR PC,AVER
        JSR PC,$ENTRY     ;STORE THE CONVERSION IN MEMORY
        .PRINT #HEADER    ;PRINT THE LINE HEADER
        JSR PC,TYPE       ;TYPE OUT THE PRINTED LINE
        JMP INB
$AVDC:  MOV LINE,ALN      ;PUT THE NEXT CONVERSION NUMBER IN ALN
        DEC ALN           ;BACK ALN UP TO THE PRESENT CONVERSION
        BR $AVCNV         ;GO AND THE NEXT SET OF DATA PIONTS
RE:     TST @#LINE
      BEQ ERR
      MOV #ADDLST+2,R0
      MOV (R0),R0         ;R0 NOW POINTS TO THE
                          ; CONVERSION TO BE REPEATED
      BMI $DC             ;IF R0 IS NEGATIVE,
                          ; REPEAT THE LAST CONVERSION
      CMP R0,LINE         ;SEE IF NUMBER ACTUALLY EXISTS
      BGE ERR             ;TELL THE OPERATOR SO
      MOV R0,ALN
$CNV:   JSR PC,CNVRT
        .PRINT #HEADER    ;PRINT THE LINE HEADER
        JSR PC,TYPE       ;TYPE OUT THE PRINTED LINE
$RT:    JMP INB
$DC:    MOV LINE,ALN      ;PUT THE NEXT CONVERSION NUMBER IN ALN
        DEC ALN           ;BACK ALN UP TO THE PRESENT CONVERSION
        BR $CNV           ;REPEAT THE LAST CONVERSION
ERR:    .PRINT #RPT
        JMP INB
TY:     .PRINT #HEADER    ;PRINT THE LINE HEADER
        MOV #ADDLST+2,R0  ;THE SECOND LOCATION IN
                          ; ADLST CONTAINS THE FIRST NUMBER
        MOV (R0)+,ALN     ;PUT THE FIRST LINE
                          ; - NUMBER IN ITS STORE LOCATION
        BPL $8

```

```

MOV LINE,ALN
DEC ALN ;BACK UP AND TYPE THE PREVIOUS CONVERSION
JSR PC,TYPE
$5: JMP INB
$8: JSR PC,TYPE ;DISPLAY A LINE
$6: MOV (R0)+,ALN ;NEXT NUMBER
    BMI $5 ;FINISHED IF NEGATIVE
    JSR PC,TYPE ;DISPLAY THE LINE
    BR $6
ALL: TST @#LINE ;SEE IF WE HAVE ANY CONVERSIONS
    BEQ $NCNYT ;WE HAVE NOT MADE ANY
    .PRINT #HEADER ;PRINT THE LINE HEADER
    MOV LINE,R0 ;PUT HOW MANY LINES IN R0
    CLR ALN
$AGN: JSR PC,TYPE ;GO TYPE OUT THE NEXT LINE
    INC ALN ;NEXT LINE
    SOB R0,$AGN
$NM: JMP INB
$NCNYT: .PRINT #RPT ;TELL THE OPERATOR THAT
        ; HE HAS NO CONVERSIONS
        JMP INB ;RETURN TO THE MAIN PROGRAM
STORE: TST @#LINE ;SEE IF WE HAVE ANY CONVERSIONS?
    BNE WC
    JMP ERR ;TELL THE OPERATOR THAT
WC: MOV @#LINE,R0 ;GET THE LINE NUMBER
    ;WE ARE GOING TO CALCULATE THE
    ;ADDRESS OF THE NEXT ENTRY

    MUL #12.,R0
    MOV #999.,ARRAY(R1) ;999 IS THE DELIMITER
    MOV #5,R2 ;ALSO STORE FIVE ZEROS ON THE ARRAY
    MOV R1,R3
    ADD #ARRAY+2,R3 ;R3 CONTAINS THE ADDRESS
    ; OF THE NEXT LOCATION
HR: CLR (R3)+ ;CLEAR THE NEXT FIVE WORDS
    SOB R2,HR ;TIGHT LOOP
    MOVB #'E,(R3)+
    MOVB #'N,(R3)+
    MOVB #'D,(R3)+
    MOVB #'?',(R3)+
    MOV R3,R1
    ADD #16.,R1 ;SIX MORE WORDS ON THE ARRAY
    CLR R3 ;CLEAR OUT THE UPPER PART
    MOV #ARRAY,R2 ;R2 IS THE BUFFER POINTER
    ; AND INITIALLY CONTAINS
    ;THE ADDRESS OF ARRAY
    CLR R3 ;R3 IS THE BLOCK COUNTER
WR: .WRITW #AREA,#0,R2,#256.,R3 ;WRITE TO THE DISK
    BCS $WRERR ;AN ERROR OCCURED ON THE WRITE
    ADD #512.,R2
    INC R3

```

```

        CMP      R2,R1
        BLOS     WR
        .CLOSE #0      ;CLOSE CHANNEL 0
        CLR @#LINE     ;LETS START OVER
        RTS PC
$WRERR: SEC          ;LET THE CALLER KNOW THERE WAS AN ERROR
        RTS PC
DEVARE: .WORD 0,0
DEVLST: .WORD CDSTAT,0,0,0
AREA:   .WORD 0,11,0,0,0,0
TEMP:   .WORD 0,0,0,0,0,0
TEMP2:  .BLKW 10.
TEMP3:0
TEMP4:0
TEMP5:0
TEMP6:0
TEMP7:0
FLAG:0
PATERN: .WORD 170000,160000,140000,100000,40000,0
THT0:   .WORD 1000.,100.,10.,1.
CONST:  .WORD 60.,4,1600.,200.,50.
CON1:   .BLKW 5
TENNUM: .WORD 0
CON1PT: .WORD 0
STMESS: .ASCII/DATA STORED IN FILE /
FB:     .BLKB 82.
HEADER: .ASCII/LINE      VOLTS      AMPS      WATTS      RPM/
        .ASCIZ/          TORQUE/
BADLET: .ASCIZ/BAD COMMAND STRING/
RPT:    .ASCIZ/NO CONVERSIONS/
WARN:   .BYTE CR,LF
        .ASCII/AN ERROR HAS OCCURED ON THE OUTPUT DEVICE/
        .BYTE CR,LF
        .ASCII/THE DATA THAT YOU JUST TOOK WAS NOT STORED!/
        .BYTE CR,LF,0,0
FILNM:  .BYTE CR,LF,LF
        .ASCIZ/FILE NAME FOR OUTPUT/
NOCMD:  .ASCIZ/ILLEGAL COMMAND?/
BNUM:   .ASCIZ/ILLEGAL NUMBER/
NSL:    .ASCIZ/NO SUCH LINE/
PIE:    .ASCIZ/COMMAND STRING ERROR/
        .EVEN
NUMCMD: .WORD 13.
CMDLST: .ASCIZ/T/
        .ASCIZ/S/
        .ASCIZ/R/
        .ASCIZ/C/
        .ASCIZ/Q/
        .ASCIZ/A/
        .ASCII/FS/

```

```
.ASCII/FD/
.ASCII/AV/
.ASCII/AR/
.ASCII /BN/
.ASCII/ZR/
.ASCII/IR/
JMPTAB: .WORD TY,ST,RE,CN,QU,ALL,FI,FD,AV
        .WORD AVRE, BN1,  ZERO, INER
COMWRD: .WORD 0
CMDREG: .WORD 0
$NUM:   .WORD 0
LINE:   .WORD 0
ALN:    0
DFTEXT: .WORD 0,0,0,0
INBUF:  .BLKW 40
DEVBUF: .BLKW 2000
LIST:   .BLKW 100
ADDLST: .WORD ADDLST
        .BLKW 10.
ARRAY:  .BLKW 6000.
.END MAIN
```



```

; THE END OF DATA
; INDICATOR (999) WAS ENCOUNTERED
; IN THE
; DATA STREAM.
MOV #999.,INBUF ; FORCE IN THE END OF DATA
; INDICATOR
MOV #INBUF+2,R0 ; WE NEED TO CLEAR OUT THE
; NEXT FIVE WORDS

MOV #5,51
CLLP: CLR (R0)+ ; CLEAR OUT THE NEXT FIVE LOCATIONS
      SOB R1,CLLP
      BR $EN ; BRANCH AND SET END OF DATA
; AND WRITE OUT
R1OR2: CMPB @#FIVTWO,#2 ; SEE WEATHER A ONE OR TWO
      BLT ONE ; IF TRUE A HARD ERROR
TWO: .PRINT #NFO ; TELL THE OPERATOR THAT NO FILE
; WAS OPEN FOR
; OUTPUT
      JMP ENTRY ; RESTART THE PROGRAM
ONE: .PRINT #HARD ; HARD ERROR, TERMINATE PROGRAM
      .EXIT
      JMP ENTRY
$NER: CMP INBUF,#999. ; SEE IF THE END
      BNE NTDN
$EN: BIS #200,RSTAT ; SET DONE BIT
NTDN: JSR PC,ASCRT
      JSR PC,WR
      BCC TRST ; IF WE DID NOT HAVE AN ERROR
; CONTINUE
      TSTB FIVTWO ; LETS SEE WHAT KIND OF ERROR
; FIVTWO = 0 ATTEMPT TO WRITE
; PAST AN END OF FILE
; 1 HARDWARE ERROR ON
; CHANNEL
; 2 NO OUTPUT CHANNEL
; OPEN
      BGT W1OR2 ; IF GREATER THAN 0 MUST BE A
; ONE OR TWO
      .PRINT #ATWEOF ; TELL THE OPERATOR AND GIVE
; HIM THE CHANCE TO
; TRY AGAIN. IF THIS ERROR
; OCCURS I HAVEN'T
; THE FOGGIEST IDEA OF WHAT
; COULD BE WRONG!
; GOOD LUCK
      JMP ENTRY ; GO BACK AND RESTART
W1OR2: CMPB FIVTWO,#2 ; SEE IF A HARD ERROR OR NO FILE
      BLT WONE ; IF BRANCH SUCCEDS IT WAS A ONE
WTWO: .PRINT #NCHOP ; TELL THE OPERATOR THAT NO
; CHANNEL WAS OPEN

```

```

        BR TRST                ;CONTINUE THE EXECUTION OF
                                ; THE PROGRAM
WONE:   .PRINT #HARD           ;A HARD ERROR ON THE CHANNEL
        .EXIT                 ;BY, BY, AMERICAN PIE
TRST:   TSTB RSTAT
        BPL MAIN
        .PRINT #DNE
        .CLOSE #0             ;MAKE CHANNEL ZERO PERMNET
        .CLOSE #3
        JMP ENTRY             ;REPEAT THE FORMATTING TILL DONE
ASCRT:   MOV RO,-(SP)          ;SAVE THE GENERAL
                                ; PURPOSE REGISTERS
        MOV R1,-(SP)
        MOV R2,-(SP)
        MOV R3,-(SP)
        MOV R4,-(SP)
        MOV #CON1,@#CON1PT
        MOV #INBUF,R5
        MOV #LIST,R4          ;
        MOV #CONST,R3         ;
        MOV (R5)+,R1           ;CONVERT THE LINE NUMBER INTO ASCII
        JSR PC,DECAS
        JSR PC,SPA
AVOLTS: JSR PC,MLT             ;CONVERT THE VOLTAGE TO AN
                                ; ASCII LIST
        JSR PC,NORML
        JSR PC,CORR           ;CORRECT FOR OFFSETS
        JSR PC,DECAS
        MOVB #POINT,(R4)+      ;PUT THE DECIMAL POINT
                                ; ON THE STACK
        JSR PC,SPA             ;ADD FOUR SPACES TO THE STACK
ACUR:   JSR PC,MLT             ;CONVERT THE CURRET TO AN
                                ; ASCII LIST
        ASR R1
        ROR RO                ;DOUBLE PRECISION SHIFT ON R1,RO
        JSR PC,NORML
        JSR PC,CORR
        JSR PC,DECAS          ;THE CURRENT BY TWO
        MOVB #POINT,(R4)+      ;PUT A DECIMAL POINT ON
                                ; THE STACK
        JSR PC,FRAC
        JSR PC,SPA
AWAT:   JSR PC,MLT
        JSR PC,NORML
        JSR PC,CORR           ;CORRECT FOR OFFSETS
        JSR PC,DECAS
        MOVB #POINT,(R4)+
        JSR PC,SPA
ASPEED: JSR PC,MLT
        JSR PC,NORML

```

```

      JSR PC,CORR      ;CORRECT FOR OFFSETS
      JSR PC,DECAS
      MOVB #POINT,(R4)+
      JSR PC,SPA
ATOR:  JSR PC,MLT
      JSR PC,NORML
      JSR PC,DECAS
      MOVB #POINT,(R4)+
      JSR PC,FRAC
      MOVB #CR,(R4)+  ;PUT A CARRIAGE RETURN
      MOVB #LF,(R4)+  ;AND LINE FEED ON THE LIST
      CLRB (R4)+
      .PRINT #LIST    ;PRINT THE NUMBERS
      MOV (SP)+,R4
      MOV (SP)+,R3
      MOV (SP)+,R2
      MOV (SP)+,R1
      MOV (SP)+,R0    ;RESTORE THE GENERAL PURPOSE
                      ; REGISTERS
      RTS PC
CORR:  MOV CON1PT,R2  ;PUT THE ADDRESS OF THE CORRECTION
                      ; LIST IN R2
      ADD (R2)+,R0    ;ADD IN THE LEAST SIGNIFICANT PART
                      ; OF THE CORECTION
      ADC R1          ;ADD IN THE CARRY
      ADD (R2)+,R1    ;ADD IN THE MOST SIGNIFICANT PART
                      ; OF THE CORRECTION
      TST R1          ;WE ALLOW NO NEGATIVE NUMBERS
      BPL NOTN
      CLR R0          ;THE NUMBERS WERE LESS THAN ZERO
                      ; SO MAKE THEM ZERO
      CLR R1
NOTN:  MOV R2,CON1PT  ;SAVE THE ADRESS FOR THE NEXT
                      ; CORRECTION
      RTS PC
MLT:   MOV #5,R1      ;PUT THE VOLTS PER BIT IN R1
      MOV (R5)+,R0
      JSR PC,MUL
      MOV (R3)+,R1    ;PUT IN THE CONVERSION SCONSTANT
      JSR PC,MUL      ;MULTIPLY
      RTS PC
NORML: MOV #6,R2      ;PUT IN THE NUMBER OF SHIFTS
NML:   ASL R0          ;DO A DOUBLE PRESSION SHIFT
                      ; ON R0 AND R1
      ROL R1
      SOB R2,NML
      RTS PC
DECAS: MOV R0,-(SP)
      MOV R5,-(SP)    ;PUSH R0 AND R5 ONTO THE STACK
      MOV R1,R0

```

```

MOV #THT0,R5      ;MOVE THE ADDRESS OF THE
                  ; CONVERSION TABLE TO R5
MOV #4,R2          ;NUMBER OF LOOPS
LP:  MOV (R5)+,R1    ;DIVIDE OUT THE THOUSANDS, ECT
     JSR PC,DIV
     ADD #60,R1      ;MAKE AN ASCII CHARACTER
     MOVB R1,(R4)+   ;STORE ON THE LIST
     SOB R2, LP
     MOV (SP)+,R5
     MOV (SP)+,R0
     RTS PC ;RETURN FROM WHENCE WE CAME
SPA:  MOV R2,-(SP)   ;SAVE R2
     MOV #4,R2       ;FOUR SPACES
SPA1:  MOVB #SPACE,(R4)+ ;FOUR SPACES ON THE LIST
     SOB R2,SPA1
     MOV (SP)+,R2
     RTS PC
FRAC:  MOV #10.,R1   ;FIRST PART OF FRACTION
     JSR PC,MUL
     ADD #60,R1      ;MAKE AN ASCII CHARACTER
     MOVB R1,(R4)+   ;STORE ON LIST
     MOV #10.,R1     ;NEXT DIGIT
     JSR PC,MUL
     ADD #60,R1
     MOVB R1,(R4)+
     RTS PC
RD:    MOV R0,-(SP)   ;SAVE THE GENERAL
                  ; PURPOSE REGISTERS
     MOV R1,-(SP)
     MOV R2,-(SP)
     MOV R3,-(SP)
     MOV R4,-(SP)
     MOV (R5)+,R2     ;R2 NOW CONTAINS THE NUMBER
                  ; OF WORD TO BE READ
     MOV (R5)+,R4     ;R4 IS CALLERS BUFFER ADDRESS
     MOV LSAV,R3      ;GET THE BYTE OFFSET AND
                  ; PUT IT INTO R3
     TST RSTAT        ;IF RSTAT IS MINUS WE HAVE READ
                  ; A BLOCK
     BMI $NR ;NEGATIVE WE HAVE MADE THE FIRST READ
     JSR PC,READ      ;GO READ A BLOCK
     BCC $NR          ;DID WE HAVE AN ERROR?
     JMP ERR1         ;I GUESS WE DID, THAT'S TO BAD
$NR:   CMP R3,#512.    ;SEE IF WE ARE AT THE END OF
                  ; THE BUFFER
     BLT $SK1        ;BACON LETTICE AND TOMATO YUMMY
     JSR PC,READ
     BCC $SK1
     JMP ERR1
$SK1:  CMP BUFF(R3),#999. ;ARE WE AT THE END OF

```

```

                                ; THE BUFFER?
BLT $LP
JMP LAST                       ; AT THE END
$LP:  MOV BUFF(R3),(R4)+        ; LETS MOV THE DATA TO
                                ; THE CALLERS BUFFER
ADD #2,R3                      ; BUMP R3
CMP R3,#512.                   ; END OF THE BUFFER?
BLT $SK2
JSR PC,READ
BCC $SK2
JMP ERR1
$SK2: SOB R2,$LP               ; LOOP TILL ALL THE WORDS ARE READ
MOV R3,LSAV                    ; RESORE THE CALLER
$ED1:  MOV (SP)+,R4
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
CLC                             ; NO ERRORS
RTS R5                          ; R5 IS THE LINK
ERR1:  MOV (SP)+,R4
MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
SEC                             ; A SET CARRY SAYS ERROR
RTS R5                          ; R5 IS THE LINK
READ:  .READW #AR1,#3,#BUFF,#256.,BLOCK
BCC $SK4
MOVB @#52,FIVTWO               ; SAVE THE SYSTEM
                                ; ERROR BYTE
JMP RETERR
$SK4:  INC BLOCK
BIS #100000,RSTAT
CLR R3
CLC
RTS PC
RETERR: SEC
RTS PC
LAST:  MOV #999.,(R4)+
MOV #5,R1
LAST1: CLR (R4)+
SOB R1,LAST1
BR $ED1
WR:    MOV R0,-(SP)             ; SAVE THE GENERAL
                                ; PURPOSE REGISTERS
MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
MOV R4,-(SP)

```

```

MOV WRSAB,R2
CLR R1          ;CLEAR OUT THE LIST BYTE COUNTER
$WLP:  TSTB LIST(R1) ;SEE IF WE AT THE END OF THE LIST
      BEQ LWRITE
      CMP R2,#512.  ;SEE IF WE HAVE FILLED UP THE
                  ; WRITE BUFFER

      BLT $SK7
      JSR PC,WRITB
      BCS $WRERR    ; IF WE HAD A WRITE ERROR
                  ; THEN REPORT
$SK7:  MOVB LIST(R1),WBUF(R2) ;TRANSFER ONE WORD
      CMPB (R1)+,(R2)+      ;ADD 1 TO R1 AND R2
      BR $WLP
WRITB:  .WRITW #AR1,#0,#WBUF,#256.,WBLOCK
      BCC $SK6
      MOVB @#52,FIVTWO;SAVE THE SYSTEM ERROR BYTE
      SEC          ;THERE WAS AN ERROR OF SOME TYPE
      RTS PC
$SK6:  INC WBLOCK
      CLR R2      ;CLEAR OUT THE BUFFER COUNT
      CLC        ;NO ERRORS
      RTS PC
LWRITE: TSTB RSTAT
      BPL RSTORE  ; IF RSTAT IS NEGATIVE, WE NEED
                  ; TO PADD WITH ZEROS
PADD:  CLRB WBUF(R2)
      INC R2      ;INC R2, THIS PADD THE END OF
                  ; THE BUFFER WITHZEROS
      CMP R2,#512. ; IF R2<512 KEEP GOING TILL DONE
      BLE PADD
      JSR PC,WRITB
      BCS $WRERR  ; IF WE HAD A WRITE ERROR THEN REPORT
RSTORE: MOV R2,WRSAB
      MOV (SP)+,R4
      MOV (SP)+,R3
      MOV (SP)+,R2
      MOV (SP)+,R1
      MOV (SP)+,R0
      CLC        ;WE HAD NO ERRORS
      RTS PC
$WRERR: MOVB @#52,FIVTWO      ;SAVE THE SYSTEM
                  ; STATUS WORD
      MOV (SP)+,R4
      MOV (SP)+,R3
      MOV (SP)+,R2
      MOV (SP)+,R1
      MOV (SP)+,R0
      SEC          ; TELL THE CALLER THERE
                  ; WAS AN ERROR
      RTS PC      ;RETURN FROM THE HINTHER LANDS

```

```

UNSTAK:CON1:      .WORD 0,-3,154632,-1,0,-78.,0,-10.
CON1PT: .WORD 0
FIVTWO: .WORD 0
LSAV:   .WORD 0
WRSV:   .WORD 0
INBUF:  .WORD 0,0,0,0,0,0,0,0
BLOCK:  .WORD 0
WBLOCK: .WORD 0
RSTAT:  .WORD 0
AR1:    .WORD 0,0,0,0,0,0,0
AREA:   .WORD 0,0,0,0,0,0,0
TEMP:   .WORD 0,0,0,0,0,0,0
THTO:   .WORD 1000.,100.,10.,1.
CONST:  .WORD 60.,4,1600.,200.,50.
WBUFF:  .BLKW 256.
BUFF:   .BLKW 256.
ATWEOF: .ASCIZ/ATTEMPT TO WRITE PAST UNIT END OF FILE/
NCHOP:  .ASCII/NO OUTPUT DEVICE WAS SPECIFIED./
        .BYTE CR,LF
        .ASCII/THE DATA IS NOT BEING STORED IN A/
        .ASCII/ REFORMATTED FORM/
        .BYTE CR,LF
        .ASCII/          !!!!!WARNING!!!!!!/
        .BYTE CR,LF,0,0
$99:    .ASCII/ AN END OF FILE WAS ENCOUNTERED ON THE/
        .ASCII/ UNIT BEFORE/
        .BYTE CR,LF
        .ASCII/THE END OF DATA NUMBER (999/
        .ASCII/). 999 FORCED AS END OF DATA/
        .BYTE CR,LF,0,0
NFO:    .ASCIZ/NO FILE WAS OPEN FOR INPUT/
HARD:   .ASCIZ/HARD ERROR ON UNIT/
FILENM: .ASCIZ/FILE NAMES FOR FORMATTING/
DNE:    .ASCIZ/DONE/
.EVEN
LINE:   .WORD 0
SPSAV:  .WORD 0
ALN:    .WORD 0
DFTEXT: .RAD50 "DAT"
        .RAD50 "MAC"
        .RAD50 "MAC"
        .RAD50 "MAC"
LIST:   .BLKW 256.
DEVBUF: .BLKW 2000
.END ENTRY

```

```

;!!!!!!! ***** PROGRAM NAME: COMUT.MAC ***** !!!!!!!!
;!      THIS A COMMUNICATING PROGRAM BETWEEN LSI-11 AND !
;! OTHER COMPUTER, THERE ARE 8 CONTROL WORDS TO INTERFACE!
;! THE OPERATER TO CONTROL THE PROGRAM. THEY ARE :
;!      C/N: COME BACK TO LSI-11 FROM OTHER COMPUTR
;!      C/P: START TRANSMITTING A FILE FROM LSI-11 TO
;!      OTHER COMPTER
;!      C/Z: END TRANSMITTING A FILE FROM LSSI-11 TO
;!      OTHER COMPUTER
;!      C/V: START RECEIVING A FILE FROM OTHER COMPUTER
;!      TO LSI-11
;!      C/W: END RECIVING A FILE FROM OTHER COMPUTER
;!      TO LSI-11
;!      C/A: HALF TRAN. DELAY TIME AFTER A CHARACTER
;!      HAS BEEN TRANSMITTED
;!      C/B: DUBLE TRAN. DELAY TIME AFTER A CHARACTER
;!      HAS BEEN TRANSMITTED
;!      C/C: EXIT FROM "COMUT"
;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
.TITLE COMUT
.MCALL .READW,.CLOSE
.MCALL .GTJB,.INTEN,.SYNCH
.MCALL .WRITW
.MCALL ..V2....REGDEF,.FETCH,.LOOKUP,.PRINT
.MCALL .EXIT
..V2..
.REGDEF
ERRWD=52
.MCALL .CSIGEN
..V2..
.REGDEF
.CSECT
CNTRC=03
CNTRA=01
CNTRV=26
CNTRW=27
CNTRB=02
BEL=7
CNTRLR=20
CR=15
LF=12
PSW=340
CNTRLN=16
CNTRLZ=32
INTE=100
XON=21
XOFF=23
RECSTA=176500
RECBUF=176502
TMXSTA=176504

```

```

TMXBUF=176506
RCSTA=177560
RCBUF=177562
XCSTA=177564
XCBUF=177566

START:  MOV #LUPG01,@#34          ;PUT IN TRAP VECTOR
        MOV #PSW,@#36
        MOV #JOB,R5              ;ADDRESS OF THE JOB NUM IN R5
        .GTJB #AREAS,R5 ;R5 POINTS TO THE JOB STATS
        MOV (R5),SYNBLK+2        ;PUT THE JOB #IN PLACE
ANFILE: CLR STATUS                ;RESET THE PROGRAM POINTERS
        CLR ST3                  ;ST3 IS WHERE THE NUMBER OF WORDS
        MOV #7777,FLAG2 ;RT-11 STORES DATA,SETS TRANSMIT
                                ;DELAY FLAG
        CLR FLAG                 ;FLAG IS RECEIVE MODE
        CLR FLAG1                ;FLAG1 IS END RECEIVE
        CLR FLAG3                ;FLAG3 IS TRANSMIT FLAG
        .PRINT #MESS1
        .CSIGEN #FREE,#DEFAULT, #0
        CLR R0
        MOV #-1,R1              ;THE INITIAL BLOCK IS -1+1=0
        CLR R2
        CLR R3
FAIL:   BCC STARTD              ;LOOK UP SUCEEDED
LUPG01: CLR RCSTA                ;NO INTERRUPT
        CLR RECSTA              ;NO INTERRUPT
        .INTEN 0                ;ENABLE THE INTERRUPTS
        .SYNCH #SYNBLK
        BR SYFAIL                ;SYN. FAIL
        MOV #INTE,RCSTA          ;ENABLE INTERRUPT
        MOV #INTE,RECSTA         ;ENABLE INTERRUPT
        MOV @#ST5,R1            ;RECALL R1
        .READW #AREA3,#3,#WRDBUF,#256.,R1 ;READ 256 CHAR.
        BCC WRITE
        BIS #200,@#STATUS
WRITE:  MOV R0,ST3               ;SAVE R0 FOR LATER USE
        ASL ST3                  ;MAK ST3 INTO BYTES
        RTS PC                   ;THE SYNC DOES THE RTI
SYFAIL: JSR PC,DELAY            ;WAIT FOR A WHILE
        .PRINT #MSG5             ;PRINT MESSAGE
        JMP LEAVE                ;TRY AGAIN
NLD:   TSTB @#STATUS            ;SEE IF THE LAST BLOCK
        BMI RET ;IF SET YES
        CMP R3,ST3              ;SEE IF THE LAST CHARACTER
        BGE LASTB               ;YES GO TO LAST BLOCK
        CMPB WRDBUF(R3),#LF      ;SEE IF IT IS A LINEFEED
        BNE SKIP                ;IT WAS NOT DONT INC R3
        CMPB TEMP2,#CR           ;PRECIVIOUS ONE IS CR?
        BNE SKIP                ;NO
        INC R3                   ;INC THE WORD POINTER OVER IT
SKIP:   MOVB WRDBUF(R3),TMXBUF   ;TMX THE WORD

```

```

        MOVW    WRDBUF(R3),TEMP2          ;SAVE IT
        INC R3                          ;BUMP R3
        CLR     TMXSTA ;STOP TRAN. FOR A WHILE
        MOV     #DELY2,(SP)              ;GO TO DELAY
STUFF:   RTI      ;GO TO SLEEP
LASTB:   CLR R3 ; SET THE WORD BUF POINT TO ZERO
        CLR @#TMXSTA ;NO INTERUPTS DURING THE READ
        INC R1 ;INC THE BLOCK COUNT
        MOV R1,@#ST5 ;STORE R5 FOR THE SYNC
        JSR PC,TR ;READ THE NEXT BLOCK
        MOV #INTE,TMXSTA ;ENABLE THE ENTERUPT
        RTI      ;GO TO SLEEP
RET:     CLR @#TMXSTA ;NO MORE TMX INTERUPTS
        MOV     #END2,(SP)              ;GO TO END
        MOVW #CNTRLZ,TMXBUF ;SEND AN END OF FILL TO PIP
        MOV ST2,@#304 ;
        RTI      ;
STARTD:  MFPS -(SP) ;SAVE THE OLD PSW
        MTPS #340 ;NO INTERRUPTS
        MOV @#60,-(SP)
        MOV @#62,-(SP) ;SAVE THE TTY INTERRUPT VECTORS
        MOV @#XCSTA,-(SP) ;SAVE THE TTY TRSMIT INT STA
        MOV @#RCSTA,-(SP) ;SAVE THE TTY RECVER INT STA
        MOV SP,TEMSP ;SAVE STAKE POINTER
        CLR @#XCSTA ;SET STA TO 0
        CLR @#RCSTA ;SET STA TO 0
        MOV #INTE,RCSTA ;ENABLE THE KEYBOARD INT
        MOV #RCSER,@#60 ;PUT THE ADDRESS OF THE SERVICE
        MOV #340,@#62 ;AND PS STATUS IN VECTOR
        MOV #INTE,RECSTA
        MOV #XONOFF,@#300
        MOV #340,@#302
        MTPS #0
HANG:    NOP
        BR HANG ;SEE IF BUSY
RCSER:   MOV @#RCBUF,R4 ;STORE CHARA
        BIC #177600,R4 ;CLEAR OUT THE UPPER BITS
        CMPB #CNTRLP,R4 ;SEE IF WE WANT TO DUMP TO TEN
        BEQ SAVE
        CMPB #CNTRLN,R4 ;SEE IF WE WANT TO LEAVE THE TEN
        BNE 7$
        MOV #LEAVE,(SP) ;GO TO LEAVE
        RTI
7$:      CMP #CNTRV,R4 ;CONTROL V STARTS RECEIVE DATA
        BNE 1$
        MOV #1,FLAG ;SET FIRST TIME FLAG
        MOV #DATA,R3 ;R3 IS ADDR. OF DATA BUFFER
        MOV #STORE,@#300 ;SET TRAP VECTOR FOR RECEIVE
        CLR TEMP ;TEMP CONTAINS NO. OF STORE BLOCK
        MOVW #CR,TMXBUF ;START REC. BY END TYPE COMMAND

```

```

RTI
1$: CMP      #CNTRW,R4      ;CONTROL W ENDS REC.
   BNE      2$
   TST      FLAG          ;IS RECEIVE MODE?
   BEQ      11$           ;NO REC. MODE
   CLR      RECSTA        ;NO MORE RECIVE
   MOVB     #CNTRC,TMXBUF  ;END TRAN.
   MOV      #XONOFF,300   ;SET VECTOR IN TRAN.
   MOV      #STORE3,(SP)  ;GO TO END SERVICE
   RTI
11$: MOV      #WRONG,(SP)   ;PRINT MESSAGE
   RTI
2$: TST      FLAG3         ;IS TRANSMITTING NOW?
   BEQ      3$            ;NO
   CMP      #CNTRLZ,R4    ;CONTROL Z ENDS TRAN.
   BNE      3$
   CLR      TMXSTA        ;NO MORE TRAN.
   MOV      #END2,(SP)    ;GO TO END SERVICE
   RTI
3$: CMP      #CNTRA,R4     ;CNTR A IS TO HALF TRAN. DELAY
   BNE      5$            ;NO
   TST      FLAG2         ;IS LARGEST TIME DELAY
   BLE      9$            ;YES
   CMP      FLAG2,#1      ;YES,DOES REACH MIN. TIME
   BEQ      8$            ;YES,THEN DO NOT REDUCE TIME
   ASR      FLAG2         ;DIVIDE TIME BY 2
8$: RTI
9$: MOV      #77777,FLAG2
   RTI
5$: CMP      #CNTRB,R4     ;CONTROL M IS ADD TRAN. DELY
   BNE      6$
   ASL      FLAG2         ;MULTIPLY TIME BY 2
   RTI
6$: TSTB     @#TMXSTA      ;TRAN. FINISHED ?
   BPL      NO
   MOV      R4,TMXBUF     ;TRAN. A CHAR.
NO: RTI
SAVE: MOV      #1,FLAG3    ;SET RECEIVE FLAG
      MOV      @#304,ST2
      MOV      #NLD,@#304 ;NEXT CHARACTER ROUTINE
      JMP      NLD
TR:  TRAP
      RTS      PC
LEAVE: .CLOSE #0          ;END RESEIVE
      .CLOSE #3          ;END TRAN.
LEAVE1: CLR     @#TMXSTA   ;NO MORE DEC TEN INTERUOPTS
      CLR     @#RECSTA    ;
      MOV     TEMSP,SP    ;RESTORE STAKE POINTER
      MOV     (SP)+,RCSTA ;PUT THE OLD INTERRUPT ENA BACK IN
      MOV     (SP)+,XCSTA ;PUT THE OLD INTERRUPT ENA BACK IN

```

```

MOV (SP)+,@#62 ;PUT THE OLD INTE VECTOR
MOV (SP)+,@#60 ;BACK IN THE TTY VECTOR
MTPS (SP)+ ;RESTORE THE OLD PSW
JMP ANFILE
XONOFF: MOV @#RECBUF,@#ST4 ;STORE
BIC #177600,@#ST4 ;CLEAR OUT UPPER BITS
CMP #BEL,ST4 ;IS BELL SIGNAL
BNE 1$
CLR TMXSTA ;PAUSE TRAN.
MOVB ST4,XCBUF ;ECHO
MOV #DELY,(SP) ;DELAY 5 SEC.
RTI
1$: CMP #XON,@#ST4 ;SEE IF AN XON
BEQ ON ;YES GO TURN ON
CMP #XOFF,@#ST4
BEQ OFF
MOV @#ST4,@#XCBUF ;ECHO THE CH
RTI ;RETURN
OFF: CLR @#TMXSTA
RTI
ON: MOV #INTE,@#TMXSTA
RTI
DELY2: MOV FLAG2,WAIT3 ;DELAY
1$: DEC WAIT3
BNE 1$
BR OUT
DELY: MOV #77,WAIT1 ;SET DELAY TIME
DELY3: MOV #7777,WAIT2 ;DELAY .1 SEC.
1$: DEC WAIT2
BNE 1$
TST WAIT1
BMI OUT
DEC WAIT1
BR DELY3
OUT: MOV #INTE,TMXSTA ;ALLOW TRAN.
BR .
STORE: MOVB RECBUF,(R3) ;PICK UP A CHAR.
MOVB (R3),XCBUF ;ECHO
BIC #200,(R3) ;MASK BIT 7
CMPB (R3),#LF ;IS THE RESPONSE OF CR
BNE 1$ ;NO
MOV #STORE1,@#300 ;YES,GO TO RECIVE
1$: RTI
STORE1: MOVB RECBUF,(R3) ;PICK UP DATA
MOVB (R3),XCBUF ;ECHO
CMPB (R3)+,#CNTRW ;IS END TRAN. ?
BNE 1$ ;NO
CLR RECSTA ;YES,NO MORE RECEIVE
MOVB #CNTRC,TMXBUF ;STOP TRAN.
MOV #-1,FLAG ;SET END RECEIVE

```

```

MOV B #0,-1(R3) ;RUBOUT C/W
1$: CMP R3,#DATA+512. ;REACH ONE BLOCK?
BLO 2$ ;NO
MOVB #XOFF,TMXBUF ;YES,PAUSE TRAN.
MOV #STORE2,(SP) ;GO TO STORE A BLOCK
RTI
2$: TST FLAG ;IS END RECEIVE
BPL 3$ ;NO
MOV #STORE3,(SP) ;GO TO FILL A BLOCK & STORE
3$: RTI
STORE3: MOV #-1,FLAG ;SET LAST STORE
CLR FLAG1
1$: CMP R3,#DATA+512. ;REACH A BLOCK
BHS STORE5 ;YES
MOVB #0,(R3)+ ;FILL ZERO
BR 1$
STORE2: MOV #77777,WAIT2 ;WAIT FOR A WHILE
1$: DEC WAIT2
BNE 1$
STORE5: CLR RECSTA ;NO INTERRUPT
CLR RCSTA
MOV R3,TEMP1 ;SAVE R3
.WRITW #AREA0,#0,#DATA,#256.,TEMP;STORE DATA IN DISK
BCS WTERR ;WRITE DISK ERR
INC @#TEMP ;BUMP BLOCK NO.
MOV TEMP1,R3 ;RESTORE R3
MOV #DATA+512.,R1 ;
MOV R3,R2 ;SAVE
MOV #DATA,R3 ;RE-RECEIVE
2$: CMP R2,#512.+DATA ;IS ANY MORE DATA LEFT
BLOS 1$ ;NO
TST FLAG ;IS END TRAN.?
BGE 3$ ;NO
MOV #-1,FLAG1 ;YES,SET END FLAG
3$: MOVB (R1)+,(R3)+ ;COLLECT IT
DEC R2
BR 2$ ;ANY MORE ?
1$: TST FLAG1 ;END TRAN.?
BMI STORE3 ;YES
MOV #INTE,RCSTA ;ENABLE INTRRUPT
MOV #INTE,RECSTA
MOVB #XON,TMXBUF ;RESTART REC.
TST FLAG ;IS LAST BLOCK
BGE 4$ ;NO
.CLOSE #0 ;END RECEIVE
JSR PC,DELAY ;WAITE FOR A WHILE
.PRINT #MSG1 ;OUTPUT END MESSAGE
JMP LEAVE ;AGAIN
4$: BR .
WTERR: JSR PC,DELAY ;WAITE FOR A WHILE

```

```

        .PRINT  #MSG3
        JMP     LEAVE
END2:   TSTB     TMXSTA  ;DONE?
        BPL     END2
        MOV     #CNTRLZ, TMXBUF  ;END TRAN.
        .CLOSE  #3
        JSR     PC, DELAY          ;WAITE FOR A WHILE
        .PRINT  #MSG2
        BR      .
WRONG:  JSR     PC, DELAY          ;WAIT FOR A WHILE
        .PRINT  #MSG4
        BR      .
DELAY:  MOV     #0, WAIT1
1$:     DEC     WAIT1
        BNE     1$
        RTS     PC
MSG1:   .ASCII//<15><12><12>/STOP RECIVE, FILE STORED/
        .ASCIIZ//<15><12>//
MSG2:   .ASCIIZ//<15><12><12>/STOP TRANSMIT/<15><12>//
MSG3:   .ASCII//<15><12><12>/WRITE DISK ERR!/
        .ASCIIZ/ TRY AGAIN OR LOG OFF !!!<15><12>//
MSG4:   .ASCII//<15><12>/NO FILE RECEIVED !!!
        .ASCII//<15><12>/TYPE CONTROL V TO RECEIVE A/
        .ASCIIZ/ FILE !!!
MSG5:   .ASCIIZ//<15><12>/SYNCH FAIL, TRY AGAIN!!!
        .EVEN
TEMP:   0
TEMSP:  0
TEMP1:  0
TEMP2:  0
WAIT1:  0
WAIT2:  0
WAIT3:  0
FLAG3:  0
FLAG2:  0
FLAG1:  0
FLAG:   0
AREA0:  .BLKW 10
ST1:    0
ST2:    0
ST3:    0
ST4:    0
ST5:    0
SYNBLK: 0
        .WORD  0
        .WORD  0
        .WORD  0
        .WORD  5
        .WORD -1,0
STATUS: .WORD  0

```

```
JOB:      .BLKW 10
MESS1:    .ASCIZ//<15><12>/FILE NAME/<15><12>/
AREA3:    .WORD 0,0,0,0,0,0,0,0,0
AREA5:    .WORD 0,0
           .WORD 0
           .WORD 0
           .WORD 0
           .WORD 0
DFAULT:   .RAD50 "MAC"
           .RAD50 "FOR"
           .WORD 0,0,0
WRDBUF:   .BLKW 512.
DATA:     .BLKW 270.
FREE:     .BLKW 2000.
           .END START
```

```

C ***** PROGRAM NAME: MODE32.FOR*****
C
C THIS PROGRAM IS USED TO SIMULATE AN INDUCTION MOTOR
C IN THE POWER SYSTEM LAB. THE INPUT OF THE MODEL IS
C LOAD TORQUE OF THE MOTOR, THE OUPUTS OF THE MODEL
C ARE THE ACTIVE & REACTIVE POWER DELIVERED TO MOTOR
C AND THE SLIP OF THE MOTOR
C
C DOUBLE PRECISION IO,PAO,PRO,TLO,VO,D1,TOQ,PA,PR,T,
2 TN,R,X(5),DX(5),R1,R2,L1,L2,M,WMO,J,D,DAMP,IC
3 ,H,DELT,P(5),DP(5),PN(5),TABLE(20,5),B(5)
4 ,A1,A2,A3,A4,A5,A6,A7,A8,V1,TOQ1,TOQ2,SP,
5 TL(40),A,Z(4,4),Y(4),ZT(4,4),COND,WORK(4),IPVT(4)
REAL EP(5),HMINA,HMAXA,EMAX
INTEGER KD(5),KQ(5),N1
C
C PREPARE PARAMETER FOR CALLING DQSYK ROUTINE
C
C MXSTEP =20000
C DO 12 I=1,5
C EP(I) = - DMAX1(1.D-5,1.D-5*DABS(X(I)))
12 KD(I) = -1
C N = 1
C TN = 0.D0
C T = 0.D0
C DELT=1.D0/3.D1
C H = DELT/10.D0
C HMINA = H / 65536.D0
C HMAXA = H
C
C CALCULATE STARTING VALUE OF THE MODEL,THE NOTATIOND
C USED AS FOLLOWINGS:
C R1=RS (RS IS STATOR RESISTANCE OF EACH PHASE)
C R2=RR (RR IS ROTOR RESISTANCE OF EACH PHASE)
C L1=3/2*LS (LS:SELF-INDUCTANCE OF EACH STATOR PHASE)
C L2=3/2*LR (LR:SELF-INDUCTANCE OF EACH ROTOR PHASE)
C M=3/2*MRS (MRS:MAX. MUTUAL INDUCTANCE BETWEEN EACH
C ROTOR AND STARTOR PHASES)
C W (W IS ELECTRICAL ANGULAR FREQUENCE)
C WM (WM IS MECHANICAL ANGULAR FREQUENCE)
C R (R IS NUMBER OF POLE PAIRS)
C J (J IS INERTIA OF THE WHOLE SYSTEM)
C D (D IS DAMPING COEFFICIENT)
C TOQ (TOQ IS THE LOAD TOQUE OF THE MOTOR)
C V (V IS THE PHASE VOLTAGE OF POWER SUPPLY)
C
C INPUT VOLTAGE AND SPEED FOR THE MOTOR

```

```

601      WRITE(5,601)
        FORMAT('  SP,V0=?')
        READ(5,*) WMO,V0
        J=.2D0
        L1=.052D0
        R1=.5D0
        R2=.65D0
        D=.05D0
        TOQ2=32.D0
        PRO=325.D1
        PA0=5.D3
        L2=L1
        M=.96D0*L1
        R=3.D0
        V1=V0
        WRITE(6,602) R1,R2,L1,L2,M,D,J,WMO,V0
        WRITE(5,602) R1,R2,L1,L2,M,D,J,WMO,V0
602      FORMAT('  R1,R2,L1,L2,M,D,J,SP,V0=',7(F6.5,2X),
2         4(F8.3,2X))
        WMO=WMO*2.D0*3.14156D0/60.D0
        W=2.D0*3.14156D0*60.D0
        WRITE(5,303) PA0,PRO,TOQ2,WMO,W
        WRITE(6,303) PA0,PRO,TOQ2,WMO,W
303      FORMAT('  PA0,PRO,TOQ2,WMO,W=',5(F14.5,3X))
C
C      CALCULATE X1 TO X5 FOR PREPARING CALL DQSYK
C
        D1=L1*L2-M*M
        Z(1,1)=-R1*L2
        Z(1,2)=R2*M
        Z(1,3)=R*M*M*WMO+W*D1
        Z(1,4)=R*L2*M*WMO
        Z(2,1)=R1*M
        Z(2,2)=-R2*L1
        Z(2,3)=-R*L1*M*WMO
        Z(2,4)=D1*W-R*L1*L2*WMO
        Z(3,1)=-D1*W-R*M*M*WMO
        Z(3,2)=-R*L2*M*WMO
        Z(3,3)=-R1*L2
        Z(3,4)=R2*M
        Z(4,1)=R*L1*M*WMO
        Z(4,2)=-D1*W+R*L1*L2*WMO
        Z(4,3)=R1*M
        Z(4,4)=-R2*L1
        Y(1)=-L2*V0
        Y(2)=M*V0
        Y(3)=0
        Y(4)=0
        CALL MINV (4,4,4,Z,ZT,COND,WORK,IPVT)
        IF ((COND+1.D0).EQ.COND) GO TO 111

```

```

      CALL MMUL (4,4,4,4,4,1,ZT,Y,X)
C
C      READ IN DATA OF TORQUE, THE DATA IS STORED
C      IN DISK UNDER FILE NAME FOR07.DAT. THE TOTAL
C      DATA SETS ARE 40
C
      READ(7,*) (TL(I),I=1,40)
      X(5)=WMO
C
C      PRINT OUT TORQUE FOR CHECK
C
      WRITE(6,334) TL
      WRITE(5,334) TL
334    FORMAT(' TL=',5(F8.3,3X))
      A2=2.54D-2*9.8D0/2.2043D0
C
C      CONVERT TORQUE TO NT-M UNIT
C
      DO 202 I=1,40
202    TL(I)=TL(I)*A2/2.D0
      WRITE(6,299) TL
299    FORMAT(' TL=',5F14.5)
C
C      CALL SUBROUTINE DQSYK TO SOLVE DIF. EQUATION
C
      CALL DQSYK (5,T,X,DX,KD,EP,IFLAG,H,HMINA,HMAXA,
2      DELT,1.4D0,MXSTEP,KSTEP,KEMAX,EMAX,KQ,PN,TABLE)
      GO TO 1200
1100    CALL DQSYK1 (5,T,X,DX,KD,EP,IFLAG,H,HMINA,HMAXA,
2      DELT,1.4D0,MXSTEP,KSTEP,KEMAX,EMAX,
3      KQ,PN,TABLE)
1200    GO TO (1001,1002,1003,1004,1005,1006,1007,1008)
2      ,IFLAG
1001    DO 13 I=1,5
13      EP(I) = - DMAX1(1.D-5,1.D-5*DABS(X(I)))
      V0=V1
      TOQ=TL(N)+(TL(N+1)-TL(N))*(T-N*DELT+DELT)/DELT
1002    DX(1)=(-R1*L2*X(1)+R2*M*X(2)+W*D1*X(3)
2      +R*M**2*X(3)*X(5)+R*L2*M*X(4)*X(5)+L2*V0)/D1
      DX(2)=(R1*M*X(1)-R2*L1*X(2)-R*L1*M*X(3)*X(5)
2      +W*D1*X(4)-R*L1*L2*X(4)*X(5)-M*V0)/D1
      DX(3)=(-W*D1*X(1)-R*M**2*X(1)*X(5)-R*L2*M*X(2)*X(5)
2      -R1*L2*X(3)+R2*M*X(4))/D1
      DX(4)=(R*L1*M*X(1)*X(5)-W*D1*X(2)+R*L1*L2*X(2)*X(5)
2      +R1*M*X(3)-R2*L1*X(4))/D1
      DX(5)=(R*M*(X(2)*X(3)-X(1)*X(4))-D*X(5)-TOQ)/J
      GO TO 1100
1003    N = T/DELT + 1.5D0
      PA=(R1*(X(1)**2+X(3)**2)+R2*(X(2)**2+X(4)**2)
2      +R*M*X(5)*(X(2)*X(3)-X(1)*X(4)))

```

```

      PR=(W*(L1*(X(1)**2+X(3)**2)+L2*(X(2)**2+X(4)**2)
2      +2.D0*M*(X(1)*X(2)+X(3)*X(4)))
      SP=X(5)*60.D0/6.2832D0
      WRITE (6,270)  N,T,PA,PR,V0,TQ0,SP
270    FORMAT ('  N,T,PA,PR,V0,TQ0,SP=',I 6,3X,F7.5,
2      3X,6(F14.5,3X))
      SP=(1200.D0-SP)/1200.D0
      TQ01=TQ0/TQ02
      PA=PA/PA0
      PR=PR/PRO
      WRITE (5,305) PA,PR,TQ01,SP
      WRITE(6,305) PA,PR,TQ01,SP
305    FORMAT('  PAZ=',F10.6,5X,'PRZ=',F10.6,5X,'TQ0Z=',
2      F10.5,5X,'SLIP=',F10.5)
      IF (N .LT. 1001) GO TO 1100
      GO TO 1004
1005    WRITE (6,501) MXSTEP,T
501    FORMAT(' MXSTEP=',I7,' TOO SMALL AT T =',F10.4,
2      ' MXSTEP INCREASED BY 1000 ')
      MXSTEP = MXSTEP + 1000
      GO TO 1100
1006    EP(KEMAX)=32.*EMAX*EP(KEMAX)
      GO TO 1100
1007    HMINA=.1*H
      GO TO 1100
1008    STOP 222
1004    STOP 777
111    STOP 111
      END

```

C***** PROGRAM NAME: FILTER.FOR*****

C

C THIS PROGRAM IS USED TO ESTIMATE THE ROTATIONAL
C INERTIAL AND DAMPING FACTOR OF THE MOTOR
C THE METHOD USES AN EXTENDED KALMAN FILTER

C

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

      DOUBLE PRECISION X(35),T1,DX(35),FF(7,7),HH(2,7),R,
2     TL,P(7,7),V1(7,7),V2(7,7),DELT,H1,FN(35),
3     TABLE(20,35),TOQ(100),V(100),PA(100),SP(100),V0,SP0
4     ,D1,CD,WK(2),IP(2),V3(7,7),V5(2,7),V6(2,2),V7(2,2)
5     ,PA0,PA1,PA2,PRO,PR1,PR,TL0,TL2,SP1,RR(2,2),Q(7,7)
6     ,SL,EP1(35),R1,R2,L1,L2,M,
7     H(2),KM(7,2),V10(7),V11(7),T,TEND,W,V8(7,2)
      REAL EP(35),HMINA,HMAXA,EMAX
      INTEGER KD(35),IFLAG,KQ(35),N1,N

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C

C X1-5 ARE MODEL VARIABLES,X6 IS J,X7 IS D,
C X8-35 ARE P MATRIX

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

C

READ IN EXPERIMENTAL DATA OF V(VOLTAGE),PA(ACTIVE
POWER),SP(SPEED) AND TOQ(TOURE)
THEN CONVERT THE UNITS

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

      N=1
      PA0=5.D3
      PRO=3.25D3

```

```

      TL0=32.D0
      L1=0.052D0
      L2=L1
      M=0.96D0*L1
      R1=0.5D0
      R2=0.65D0
      R=3.D0
      W=120.D0*3.14159D0
      DO 171 I=1,99
      READ(6,*) N1,V(I),PA1,PA1,SP(I),TOQ(I)
      PA(I+1)=PA1*5.D0
      SP(I)=(SP(I))*6.2832D0/6.D1
      TOQ(I)=(TOQ(I))*2.54D-2*9.8D0/2.2043D0
789   FORMAT(' V,PA,SP,TOQ=',4F15.5)
171   CONTINUE
      DO 780 I=1,100
780   TOQ(I)=TOQ(I)-TOQ(1)
      DO 222 I=1,100
222   WRITE(7,789) V(I),PA(I),SP(I),TOQ(I)
      D1=L1*L2-M*M

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      INITIALIZE THE STATE VARIABLESS
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 102 I=1,35
102   X(I)=0.D0
      X(7)=0.05D0
      X(6)=.02D0
      DO 103 I=1,5
      Q(I,I)=10.D0
103   P(I,I)=10.D0
      P(6,6)=1.D-5
      Q(6,6)=1.D-5
      Q(7,7)=1.D-5
      P(7,7)=1.D-5
      RR(1,1)=10.D0
      RR(2,2)=.4D5
      DO 81 I=1,7
81   WRITE(7,90)(P(I,J),J=1,7)
      DO 80 I=1,2
80   WRITE(7,91)(RR(I,J),J=1,2)
90   FORMAT(' P=',7F14.5)
91   FORMAT(' RR=',2F15.5)

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CALL SUBROUTINE TO FORM X(8-35) FROM
C      P MATRIX ( IT IS SYMMETRIC)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL CALTX(X,P)

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      PREPARE VALUE OF HH AND FF
C      HH IS THE DERIVITIVE OF H,FF IS DERIVITIVE OF F
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 100 I=1,7
      DO 100 J=1,7
100    FF(I,J)=0.D0
      FF(1,1)=-R1*L2/D1
      FF(1,2)=R2*M/D1
      FF(2,1)=R1*M/D1
      FF(2,2)=-R2*L1/D1
      FF(3,3)=-R1*L2/D1
      FF(3,4)=R2*M/D1
      FF(4,3)=R1*M/D1
      FF(4,4)=-R2*L1/D1
      DO 101 I=1,7
      DO 101 J=1,2
101    HH(J,I)=0.D0
      HH(1,5)=1.D0

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      PREPARE INITIAL VALUES FOR DQSYK
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1122    MXSTEP =10000
      DO 12 I=1,35
      EP1(I)=-DMAX1(0.0001D0,0.0001D0*DABS(X(I)))
      EP(I)=SNGL(EP1(I))
12      KD(I) = -1
      T = 0.D0
      TEND=.2D0
      DELT=1.D0/3.D1
      H1 = DELT/10.D0
      HMINA = H1 / 65536.D0
      HMAXA = H1

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CALL DQSYK TO SLOVE PROPAGATION EQUATION
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

      CALL DQSYK (35,T,X,DX,KD,EP,IFLAG,H1,HMINA,HMAXA,
2          DELT,TEND,MXSTEP,KSTEP,KEMAX,EMAX,
3          KQ,PN,TABLE)
      GO TO 1200
1100  CALL DQSYK1 (35,T,X,DX,KD,EP,IFLAG,H1,HMINA,HMAXA,
2          DELT,TEND,MXSTEP,KSTEP,KEMAX,EMAX,
3          KQ,PN,TABLE)
1200  GO TO (1001,1002,1003,1004,1005,1006,1007,1008)
2      ,IFLAG

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CALCULATE THE VALUE OF VOLTAGE AND
C      TORQUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

1001  DO 13 I=1,35
      EP1(I) = - AMAX1(0.0001D0,0.0001D0*DABS(X(I)))
13    EP(I)=SNGL(EP1(I))
      T1=T/DELT
      V0=V(N)+(V(N+1)-V(N))*T1
      TL=TOQ(N)+(TOQ(N+1)-TOQ(N))*T1

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CALCULATE DERIVITIVE OF X1-7
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

1002  DX(1)=(-R1*L2*X(1)+R2*M*X(2)+W*D1*X(3)
2      +R*M**2*X(3)*X(5)+R*L2*M*X(4)*X(5)+L2*V0)/D1
      DX(2)=(R1*M*X(1)-R2*L1*X(2)-R*L1*M*X(3)*X(5)
2      +W*D1*X(4)-R*L1*L2*X(4)*X(5)-M*V0)/D1
      DX(3)=(-W*D1*X(1)-R*M**2*X(1)*X(5)-R*L2*M*X(2)*X(5)
2      -R1*L2*X(3)+R2*M*X(4))/D1
      DX(4)=(R*L1*M*X(1)*X(5)-W*D1*X(2)+R*L1*L2*X(2)*X(5)
2      +R1*M*X(3)-R2*L1*X(4))/D1
      DX(5)=(R*M*(X(2)*X(3)-X(1)*X(4))-X(7)*X(5)-TL)/X(6)
      DX(6)=0.D0
      DX(7)=0.D0

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      COMPUTE DERIVITIVE OF P MATRIX (DX(8-35))
C      FIRST COMPUTE THE VALUE OF FF MATRIX
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL CALTFF(FF,X,L1,L2,M,R1,R2,W,R,TL,D1)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      RENEW P MATRIX FROM X(8-35)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL CALTP(P,X)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CALCULATE THE VALUE OF DERIVITIVE OF P MATRIX
C      .
C      T
C      P= FF * P + P * FF + Q
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL MMUL(7,7,7,7,7,7,FF,P,V1)
      CALL XYT(7,7,7,7,7,7,P,FF,V3)
      CALL MADD(7,7,7,7,7,V1,V3,V2)
      CALL MADD(7,7,7,7,7,V2,Q,V1)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      CONVERT V1,WHICH HAS THE VALUE OF P DOUT,
C      TO DX(8-35)
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL CALTX(DX,V1)
      GO TO 1100

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      OUTPUT ACTIVE AND REACTIVE POWER,SPEED
C      AND TORQUE
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
1003      IF(N.EQ.1) GO TO 50
          IF(T.EQ.0.DO) GO TO 1100
50      PA1=(R1*(X(1)**2+X(3)**2)+R2*(X(2)**2+X(4)**2)
2      +R*M*X(5)*(X(2)*X(3)-X(1)*X(4)))

```

```

      PR=(W*(L1*(X(1)**2+X(3)**2)+L2*(X(2)**2+X(4)**2)
2    +2.D0*M*(X(1)*X(2)+X(3)*X(4)))
      SP1=X(5)*60.D0/6.2832D0
      WRITE (7,270)  N,T,PA1,PR,V0,TL,SP1
270    FORMAT ('  N,T,PA,PR,V0,T00,SP=',I 6,3X,F7.5,
2    3X,6(F14.5,3X))
      SL=(1200.D0-SP1)/1200.D0
      TL2=TL/TL0
      PA2=PA1/PA0
      PR=PR/PR0
      WRITE (5,305) PA2,PR,TL2,SL
      WRITE(7,305) PA2,PR,TL2,SL
305    FORMAT('  PA%=',F10.6,5X,'PR%=',F10.6,5X,'T00%=',
2    F10.5,5X,'SLIP=',F10.5)
      WRITE(7,21) X(6),X(7)
21    FORMAT('  J,D=',2F12.7)
      WRITE(5,21)X(6),X(7)

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C
C      CALCULATE GAIN MATRIX K
C      T      T      -1
C      K = P * HH *(HH * P * HH + R)
C

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

      CALL CALTP(P,X)
      CALL CALTH(HH,X,R1,R2,R,M)
      CALL XYT(7,2,7,7,7,2,P,HH,V8)
      CALL MMUL(2,7,2,2,7,7,HH,P,V5)
      CALL XYT(2,2,2,2,7,2,V5,HH,V6)
      CALL MADD(2,2,2,2,2,V6,RR,V7)
      CALL MINV(2,2,2,V7,V6,CD,WK,IP)
      IF((CD+1.D0).EQ.CD) GO TO 1777
      CALL MMUL(7,2,7,7,2,2,V8,V6,KM)

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

C
C      UPDATE P(+)
C      P(+)=(I- K * HH ) * P
C

```

```

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

```

```

      CALL MMUL(7,2,7,7,2,7,KM,HH,V1)
      DO 120 I=1,7
      DO 120 J=1,7
120    V2(I,J)=-V1(I,J)
      DO 121 I=1,7
121    V2(I,I)=1.D0+V2(I,I)

```

```

      CALL MMUL(7,7,7,7,7,7,V2,P,V1)
      CALL CALTX(X,V1)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      COMPUTE H VALUE AND UPDATE X(+)
C      X(+)= X + K * H
C      H HAS THE VALUE OF MEASUREMENT ERROR
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      T1=T/DELT
      H(1)=SP(N)+(SP(N+1)-SP(N))*T1-X(5)
      H(2)=PA(N)+(PA(N+1)-PA(N))*T1-PA1
      CALL MMUL(7,2,7,7,2,1,KM,H,V10)
      DO 130 I=1,7
130      X(I)=X(I)+V10(I)

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C      OUTPUT MATRIX P AND K
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      DO 188 I=1,7
188      WRITE(7,588) (V1(I,J),J=1,7)
      DO 189 I=1,7
189      WRITE(7,589) (KM(I,J),J=1,2)
588      FORMAT(' P=',7F18.5)
589      FORMAT(' K=',7F18.5)
      IF((N.EQ.1).AND.(T.EQ.0.DO)) GO TO 1100
      N=N+1
      IF(N.LT.25) GO TO 1122
      WRITE(5,502)
      WRITE(7,502)
502      FORMAT(' REACH 100 STEPS')
      GO TO 1004
1005      WRITE (6,501) MXSTEP,T
501      FORMAT(' MAX. STEP ADD 1000',3X,'STEP=',
2      I6,4X,'T=',F8.5)
      MXSTEP = MXSTEP + 1000
      GO TO 1100
1006      EP(KEMAX)=32.*EMAX*EP(KEMAX)
      GO TO 1100
1007      HMINA=.1*H1
      GO TO 1100
1008      STOP 222
1004      STOP 777
111      STOP 121
1777      WRITE(5,555)
      WRITE(7,555)
555      FORMAT(' MATRIX IS SINGULAR')

```

STOP 555
END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C

C COMPUTE X MATRIX FROM P MATRIX

C

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

 SUBROUTINE CALTX(X,P)

 DOUBLE PRECISION X(35),P(7,7)

 INTEGER N

 N=0

 DO 100 I=1,7

 DO 101 J=1,I

 K=N+J

101 X(7+K)=P(J,I)

100 N=N+I

 RETURN

 END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C

C COMPUTE P MATRIX FROM X VECTOR

C

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

 SUBROUTINE CALTP(P,X)

 DOUBLE PRECISION X(35),P(7,7)

 INTEGER N

 N=0

 DO 100 I=1,7

 DO 101 J=1,I

 K=N+J

 P(J,I)=X(7+K)

101 P(I,J)=P(J,I)

100 N=N+I

 RETURN

 END

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

C

C COMPUTE THE FF MATRIX

C

CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

 SUBROUTINE CALTFF(FF,X,L1,L2,M,R1,R2,W,R,TL,D1)

 DOUBLE PRECISION FF(7,7),X(35),D1,L1,L2,W,

2 M,R,R1,R2,TL

 FF(1,3)=W+R*M*M*X(5)/D1

 FF(1,4)=R*M*L2*X(5)/D1

```

FF(1,5)=R*M*(M*X(3)+L2*X(4))/D1
FF(2,3)=-R*L1*M*X(5)/D1
FF(2,4)=W-R*L1*L2*X(5)/D1
FF(2,5)=-R*L1*(M*X(3)+L2*X(4))/D1
FF(3,1)=-W-R*M*M*X(5)/D1
FF(3,2)=-R*L2*M*X(5)/D1
FF(3,5)=-R*M*(M*X(1)+L2*X(2))/D1
FF(4,1)=R*L1*M*X(5)/D1
FF(4,2)=-W+R*L1*L2*X(5)/D1
FF(4,5)=R*L1*(M*X(1)+L2*X(2))/D1
FF(5,1)=-R*M*X(4)/X(6)
FF(5,2)=R*M*X(3)/X(6)
FF(5,3)=R*M*X(2)/X(6)
FF(5,4)=-R*M*X(1)/X(6)
FF(5,5)=-X(7)/X(6)
FF(5,6)=- (R*M*(X(2)*X(3)-X(1)*X(4))-X(7)*X(5)-TL)
2 /X(6)/X(6)
FF(5,7)=-X(5)/X(6)
RETURN
END

```

VITA

Yu-Hua Wang was born in Shanghai, in the People's Republic of China, on June 12, 1946. He attended both elementary and high school in Shanghai, and received his Bachelor degree in August 1969 in the Physics Department of Fudan University in Shanghai, majoring in Electronics.

In February 1980, he was accepted as an exchange visiting scholar by the Electrical Engineering Department of The University of Tennessee, Knoxville and came to Knoxville in September 1980. He entered graduate school in September 1981 and received his Master of Science Degree in Electrical Engineering Department of The University of Tennessee, Knoxville in June 1982.