VECTOR CONTROLLED INDUCTION MOTOR DRIVE

USING TRANSPUTER PARALLEL PROCESSORS

by M. Sumner.

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy, May, 1990.

ACKNOWLEDGEMENTS

I would like express my gratitude to my supervisor, Dr. G.M. Asher for his guidance and support over the course of this project.

I would also like to thank the staff of the Department of Electrical and Electronic Engineering, University of Nottingham, for their technical assistance, and the Science and Engineering Research Council of Great Britain, for the project funding.

Finally, I would like to thank my friends and colleagues for their emotional support, when needed, over the last three years.

TABLE OF CONTENTS

ABSTRACT

This thesis describes the design and construction of a high performance induction motor drive, controlled by a network of parallel (INMOS) Transputer processors. The flexibility and high computational ability of the controller is demonstrated by the implementation of three forms of indirect vector control for the induction motor (here termed "V-Type, "V-Type with Current Feedback" and "I-Type") on two motor drive rigs. Results show that V-Type control with current feedback is superior, and that on-line parameter estimation (namely the rotor time constant) is required.

The controller has been expanded to incorporate two parameter identification strategies for assessment. The first, termed "Reactive Power Measurement", has proved successful in matching the controller value of rotor time constant to the actual machine value of rotor time constant. The second, termed "PRBS Injection with Cross-Correlation" has proved inconclusive and is the subject of on-going research.

The performance of the transputer parallel processing network for real time control is discussed. This assessment is felt to be significant since parallel architectures are likely to become increasingly exploited as the processors become cheaper, more powerful and flexible, and with enhanced system support.

LIST OF SYMBOLS

$v_a, v_b, v_c,$	instantaneous three phase stator voltages.
$i_a, i_b, i_c,$	instantaneous three phase stator currents.
$\underline{v}_s, \underline{i}_s,$	stator voltage/current vector defined in the stationary frame of reference
$\underline{i}_r,$	rotor current vector defined in the rotor frame of reference
$\underline{v}_{se}, \underline{i}_{se},$	stator voltage/current vector defined in the synchronously rotating frame of reference
$\underline{i}_{re},$	rotor current vector defined in the synchronously rotating frame of reference
$v_{sd}, v_{sq},$	stator voltage in a synchronously rotating d - q axis frame of reference.
$i_{sd}, i_{sq},$	stator current in a synchronously rotating d - q axis frame of reference.
$i_{rd}, i_{rq},$	rotor currents in a synchronously rotating d - q axis frame of reference.
$\Phi_{rd}, \Phi_{rq},$	d and q axis flux linking the rotor winding.
$T_e,$	electrically developed torque.
$p,$	number of pole pairs.
$M,$	stator/rotor mutual inductance.
$L_s, L_r,$	stator and rotor self-inductance.

σ,	leakage coefficient.
R_s, R_r,	stator and rotor resistance.
T_r,	rotor time constant.
i_{mr},	magnetising current.
w_e,	stator angular velocity.
w_r,	rotor angular velocity.
w_{slip},	slip angular velocity.
Θ_e, ε,	angular co-ordinates.
S,	differential operator.
s,	Laplace operator.

CHAPTER 1

INTRODUCTION

1.1)VARIABLE SPEED DRIVES

Up until the mid 80's industry traditionally relied upon the DC Motor drive for variable speed applications such as machine tool drives, paper and steel mills, traction drives and crane drives. The drive permits operation in all four quadrants of the torque-speed plane, gives good efficiency, and excellent dynamic response. The DC Machine itself suffers from several economic disadvantages when compared to the AC Induction motor: the induction motor is cheaper, more reliable, and requires less periodic maintenance. Moreover, the induction motor has a lower rotor inertia for a given power rating than the DC machine. The induction motor thus has the capability of higher dynamic performance for a given power rating. The advantages of the DC drive result from the simplicity of the power converter and control circuitry required to complete the variable speed drive.

The power converter employed by power DC drives is based on the line commutated thyristor bridge rectifier. The induction motor needs a converter capable of transforming fixed frequency ac into a variable frequency, variable voltage source at low losses. This originally necessitated the use of thyristors which cannot be commutated naturally. The extra components required for forced commutation increased the cost of the converter substantially.

The simplicity of the control circuitry derives from the fact that the DC motor has a simple control structure wherein the torque and air-gap flux of the machine are able, by the action on the commutator, to be controlled

independently. By contrast the induction motor has a complex control structure as the voltage, current, torque and speed are all interdependent, resulting in a highly coupled, non-linear, multivariable control problem.

The complexity of the original AC thyristor converters together with the complexity of the controller circuitry for induction motor control are the reasons why the DC drive was the automatic choice for high performance drives until relatively recently.

1.2)DEVELOPMENT OF AC INDUCTION MOTOR DRIVES

The development of AC drives over the last 10 years as variable speed drives has derived from the power switching devices developed over the same period as an alternative to thyristors [1],[2],[3]. The introduction of such devices as the Gate Turn Off Thyristor and high power Bipolar Junction and Field Effect Transistors for low to medium power applications has removed the need for auxiliary commutating circuits. The resulting reduction in cost of the power converter means that the AC drive has become more commonplace in variable speed drive applications.

Over the same period the problem of obtaining a fast torque response from the induction motor has received considerable attention [1],[4]. A control strategy for achieving this was derived originally by Blaschke [5] and developed by Leonhard [6],[7]. Termed "Vector" or "Field-Orientated" control, this method involves the transformation of the machine dynamics into those of a "pseudo-DC machine equivalent" in which a torque and field component of machine current are obtained. This method - described in chapter 2 - derives its transformations from the instantaneous rotor flux vector and allows the torque and flux of the induction motor to be controlled independently in a similar manner to the DC machine. The

development of microprocessor technology, as described in chapter 3, has provided devices capable of handling the computational tasks demanded by this strategy, and drives of comparable performance to equivalent DC drives have resulted [8],[9],[10]. Some of these are also described in chapter 3.

The area of vector control can be subdivided into two generic categories, each with its own advantages and disadvantages. The first, termed "direct" vector control, uses flux sensing coils mounted within the machine itself to detect the rotor flux vector. This provides accurate field orientation and results in excellent decoupling of the torque and field components of the motor current. The modifications to the machine and extra interface hardware are however expensive. The other method, termed "indirect" vector control, either uses a machine model to calculate the rotor flux vector, or imposes the vector control condition in a feed forward manner [9]. Both require an accurate knowledge of the machine parameters, (namely the rotor time constant) in order to provide accurate tracking of the flux vector. These parameters can vary during machine operation due to temperature and saturation effects, and the resultant loss of tracking can impair both transient and steady state performance. This problem is described in chapter 2.

1.3)PROJECT OBJECTIVES

The objective of this project was to construct a high performance induction motor rig controlled by a network of parallel (INMOS) Transputer processors. The rig therefore contains a controller of high computational ability and flexibility, which allows for multiple strategies to be investigated. The transputer network is easily expandable to cater for future research requirements in this area.

The indirect strategy for vector control was chosen

since this has been adopted by industry for commercial drives as it requires no alteration to the basic induction motor. No "standard" implementation of vector control exists and so the rig should be able to implement several control algorithms for evaluation, without major hardware changes. The requirement for on-line machine parameter identification is essential for high performance drives [11],[12],[13], and although several schemes for this have been proposed, none as yet have been adopted as a standard. The rig should be capable of testing several different schemes.

The INMOS Transputer was chosen as the basis of the control hardware for two reasons. Firstly, the control structure can be seen to be parallel in nature, particularly if extensive monitoring and fault finding capabilities are required, and this is discussed in chapter 3. The transputer was developed specifically as a device to be implemented in parallel processing networks. It is a highly efficient microcomputer in itself, but is easily interconnected with other transputers. Secondly it was desired to demonstrate the effectiveness of the transputer as a real time control device, and evaluate it and its associated software as a versatile development system for real time control problems. When the project was started there had been no work carried out in this area. The programming and interfacing of the transputer are discussed in chapters 5 and 4 respectively.

The project has investigated the implementation of three different forms of indirect vector control, here termed V-Type Control, V-Type control with Current Feedback, and I-Type Control. All these implemented forms are types of what are generally termed "feedforward indirect vector control" [9]. V-Type control employs only a speed feedback signal, and imposes stator voltages on the motor, calculated on the basis of the steady state voltage equations in the field orientated frame of

reference. V-Type control with current feedback uses current transducers to measure the stator currents. These are then transformed to the field orientated frame of reference and controlled again by impressed stator voltages. I-Type control uses a fast current loop around the power converter to provide a "current fed" converter. These control strategies are described in chapter 2.

Two induction motors have been used. The first machine is a low inertia squirrel-cage machine which demonstrates the potential high performance obtainable. The second was a wound rotor machine, here used only to demonstrate the flexibility of the control rig, but intended for future use in the area of rotor parameter identification. Additionally, two different schemes for on-line rotor time constant identification have been initially tested. The performance of the vector control strategies, the rotor time constant identification schemes, and of the transputer system itself are presented and discussed in chapters 6 and 7. The on-going nature of the project is also discussed.

CHAPTER 2

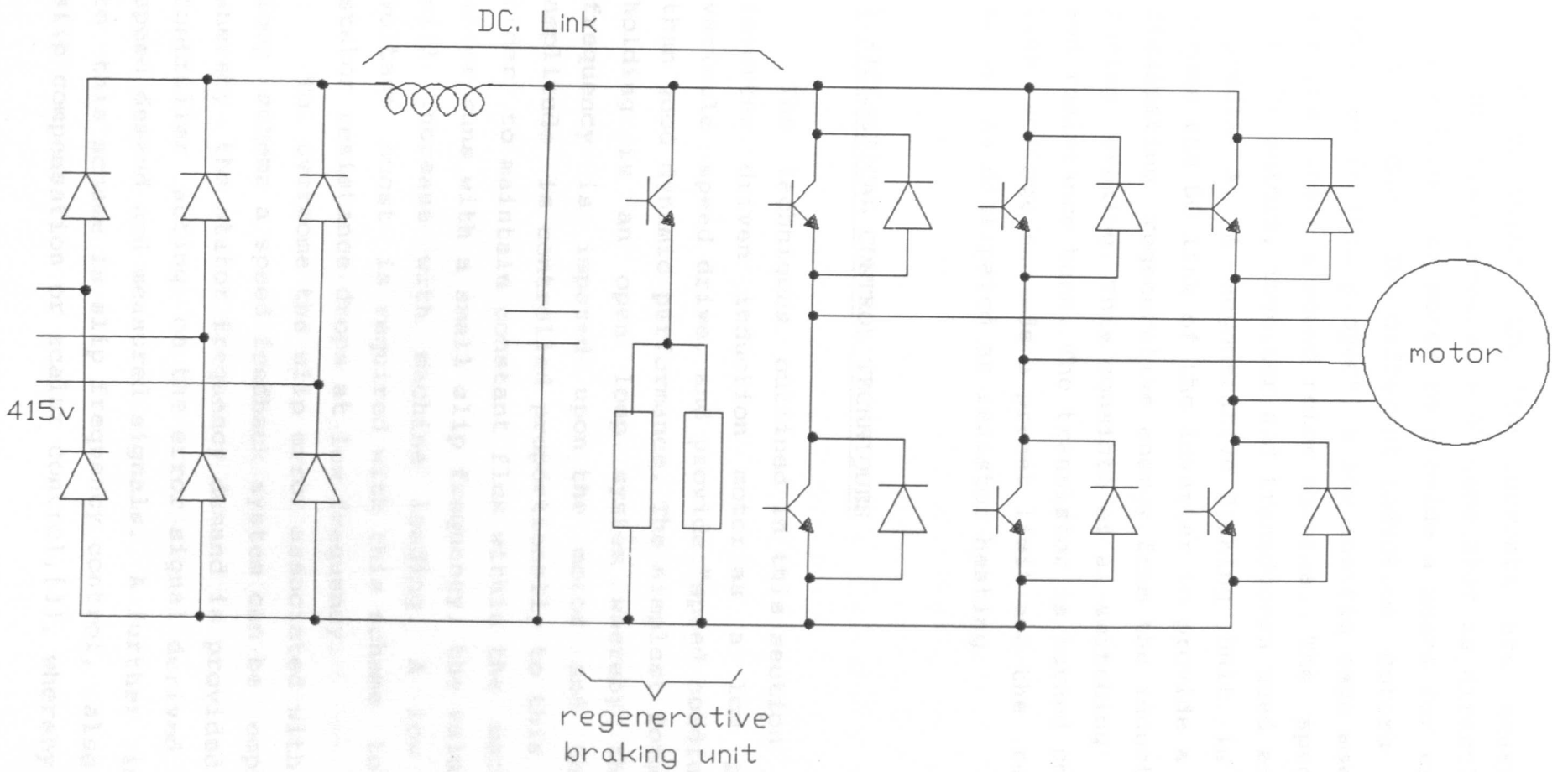
CONTROL OF INDUCTION MOTORS

2.1) INTRODUCTION

This chapter introduces the concept of Field Orientated or Vector Control for high performance induction motor drives, and compares it with operation of a DC variable speed drive system. The first section describes the typical inverter driven induction motor plant and its application to variable speed drives. The second section outlines classical control methods for "low performance" induction motor drives. Section three outlines the mathematical structure of vector control based on Generalised Machine Theory. Section four gives a brief description of how a vector control algorithm may be implemented as a practical induction motor drive. The final section introduces the problem arising from this form of control , the variation of the rotor time constant during machine operation, and outlines several of the mathematical techniques which may be employed for on line identification of that parameter.

2.2) THE VARIABLE SPEED DRIVE EMPLOYING AN INDUCTION MOTOR

The basis of the work presented in this thesis is an induction motor driven by a voltage source inverter as illustrated in Fig. 2.1. The base drive signals for the switching elements are derived from a Pulse Width Modulation strategy used by the controller and provide a three phase voltage source of variable amplitude, phase and frequency for the induction motor. A speed signal is derived from a speed encoder mounted on the shaft of the



Voltage Source Inverter Driven Induction Motor

Figure 2.1

induction motor, and line currents are measured using current transformers. A DC generator is directly coupled to the induction motor to provide a means for varying the load torque. Two different induction motors have been employed in this project, a low inertia cage machine, and a higher inertia wound rotor machine. The specifications for the motors, inverter and transducers used are given in Appendix A. A Regenerative Braking Unit is connected across the DC link of the inverter to provide a means for dissipating regenerative energy from the induction motor during braking. This consists of a switching transistor and resistance bank. The transistor is turned on when the link voltage exceeds a preset limit and the regenerative energy is dissipated as resistor heating.

2.3)CLASSICAL CONTROL TECHNIQUES

The techniques outlined in this section employ an inverter driven induction motor as a low performance variable speed drive, and provide "speed holding" rather than good dynamic performance. The simplest form of speed holding is an open loop system whereby the stator frequency is imposed upon the motor and the voltage amplitude is controlled proportionally to this value in order to maintain constant flux within the machine. The motor runs with a small slip frequency, the value of which will increase with machine loading. A low frequency voltage boost is required with this scheme to overcome stator resistance drops at low frequency.

To overcome the slip error associated with the open loop scheme a speed feedback system can be employed [3] whereby the stator frequency demand is provided by a PI controller acting on the error signal derived from the speed demand and measured signals. A further improvement to this scheme is slip frequency control, also known as slip compensation or scalar control,[3], whereby the speed

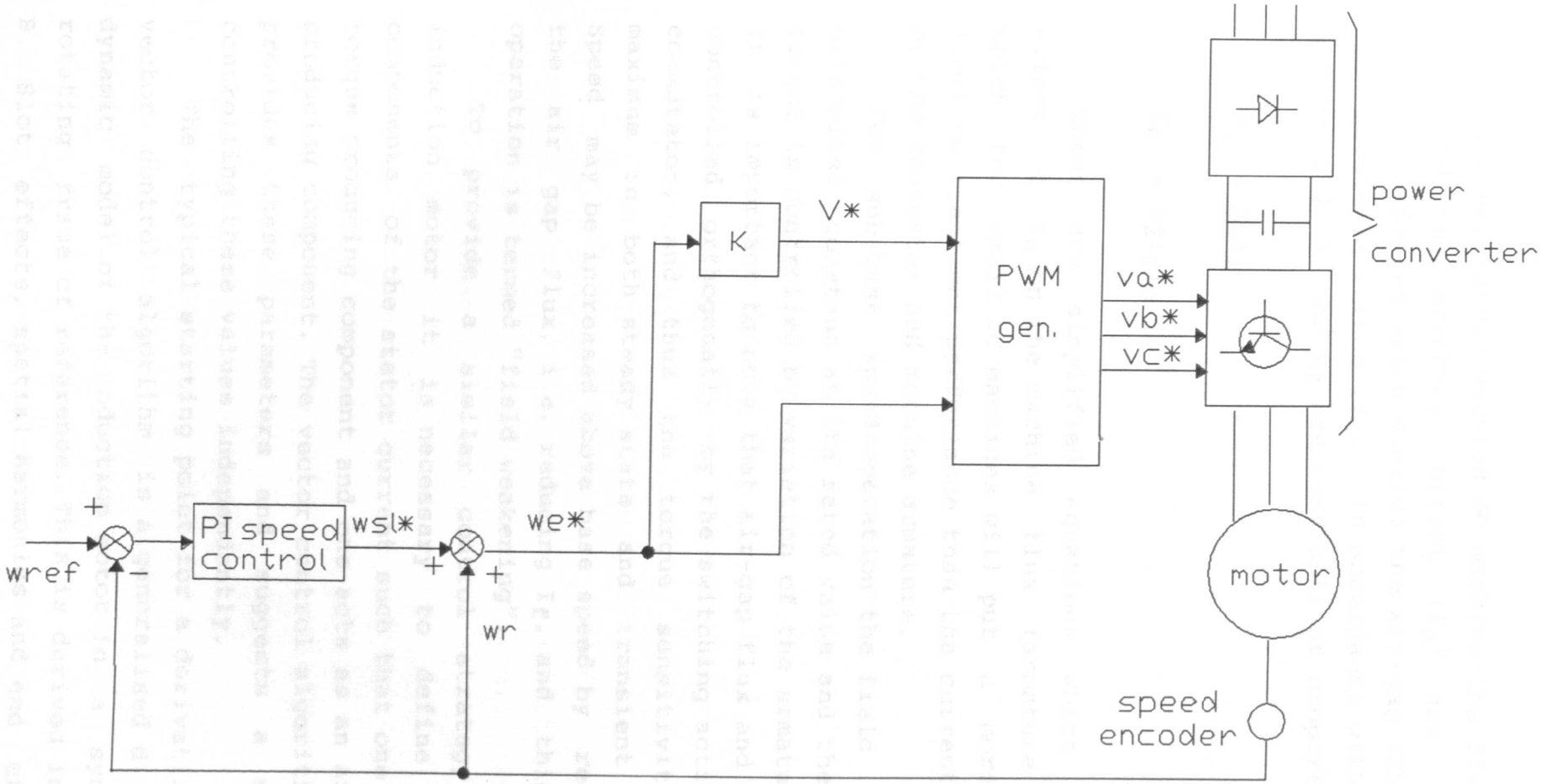
error controller provides a slip frequency reference value which is added to the measured rotor speed to provide a stator frequency reference value. This scheme is illustrated in Fig. 2.2. Again this system controls the stator voltage proportionally to the stator frequency, with a low voltage boost. The machine torque can be regarded as being proportional to the slip frequency and thus this system is in effect a torque controller within a speed controller.

The problem with these systems is that flux and torque are not closed loop controlled quantities and if the flux drifts from its preset value, the torque developed at a given slip will vary. Furthermore fluctuations in the machine parameters due to temperature effects or saturation will further impair the performance of the drive. For low performance drives these are minor considerations; however if dynamic performance is required then these problems must be overcome.

A control scheme which employs independent torque and flux control using the machine stator voltages and current measurements to synthesise torque and flux signals can be employed [3]; however the added complexity of the control circuitry does not provide a drive with dynamic performance comparable with a DC motor variable speed drive.

2.4)FIELD ORIENTATED OR VECTOR CONTROL

Vector control for induction machines was devised by Blaschke [5] and developed by Leonhard [6],[7],[8]. It is a technique for controlling independently the rotor flux and electrically developed torque of the machine. The resultant drive scheme becomes similar to that of a separately excited DC machine and transient response is proven to be superior to the "classical" induction motor drive strategies.



Slip Frequency Control for an Induction Motor.

Figure 2.2.

In a separately excited DC machine the field current (I_f) and the armature current (I_a) are independent control variables which control the air-gap flux (Φ_g) and the developed torque (T_e) , in accordance with equations 2.1 and 2.2. K_1 and K_2 are constants of proportionality.

$$\Phi_g = K_1 I_f \tag{2.1}$$

$$T_e = K_2 I_f I_a \tag{2.2}$$

These are simplified equations which ignore the effect of I_a on the machine flux (armature reaction), which for small DC machines will put a more important limit on transient performance than the current capability of the converter and machine armature.

For sub-base speed operation the field current is maintained constant at its rated value and the developed torque is controlled by variation of the armature current. It is important to note that air-gap flux and torque are controlled orthogonally by the switching action of the commutator, and thus the torque sensitivity remains maximum in both steady state and transient operation. Speed may be increased above base speed by reduction of the air gap flux, i.e. reducing I_f, and this mode of operation is termed "field weakening".

To provide a similar control strategy for the induction motor it is necessary to define orthogonal components of the stator current such that one acts as a torque producing component and one acts as an air-gap flux producing component. The vector control algorithm proposed provides these parameters and suggests a means for controlling these values independently.

The typical starting point for a derivation of the vector control algorithm is a generalised d - q axis dynamic model of the induction motor in a synchronously rotating frame of reference. This is derived in Appendix B. Slot effects, spatial harmonics and end effects are

ignored as are the harmonics of the voltage source. The equations derived in Appendix B reduce to the 4th order differential equation set given here :-

$$\begin{vmatrix} v_{sd} \\ v_{sq} \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} R_s+S\sigma L_s & -w_e\sigma L_s & SM/L_r & -w_eM/L_r \\ w_e\sigma L_s & R_s+S\sigma L_s & w_eM/L_r & SM/L_r \\ -MR_r/L_r & 0 & (R_r/L_r)+S & -(w_e-pw_r) \\ 0 & -MR_r/L_r & w_e-pw_r & (R_r/L_r)+S \end{vmatrix} \begin{vmatrix} I_{sd} \\ I_{sq} \\ \Phi_{rd} \\ \Phi_{rq} \end{vmatrix} \quad (2.3)$$

The d axis flux linking the rotor windings is :-

$$\Phi_{rd} = Mi_{sd} + L_r i_{rd} \quad (2.4)$$

The q axis flux linking the rotor windings is :-

$$\Phi_{rq} = Mi_{sq} + L_r i_{rq} \quad (2.5)$$

The torque equation for this system is :-

$$T_e = (2pM/3L_r)(i_{sq}\Phi_{rd} - i_{sd}\Phi_{rq}) \quad (2.6)$$

It can be seen that if the d axis of the rotating frame of reference is aligned to the rotor flux axis, then the d - q model simplifies significantly, particularly :-

$$\Phi_{rq} = 0 \quad (2.7)$$

$$|\Phi_r| = \Phi_{rd} \quad (2.8)$$

$$T_e = (2pM/3L_r)i_{sq}\Phi_{rd} \quad (2.9)$$

It should be noted that it is the total flux linking the rotor that is under consideration. This differs from the air - gap flux (which is used for the DC machine) by the rotor leakage. The rotor flux is used as it simplifies

dynamic equations used later, and is compensated for in the torque expression by the factor (M/L_r).

Putting $\Phi_{rq} = 0$ as demanded by the orientation described above, then row 4 of eqn. 2.3 becomes :-

$$w_e = pw_r + (MR_r/L_r)(i_{sq}/\Phi_{rd}) \qquad (2.10)$$

The second term on the right hand side is seen to represent the slip angular velocity. Considering row 3 of eqn. 2.3 gives :-

$$i_{sd} = ((1/M) + S(L_r/(MR_r)))\Phi_{rd} \qquad (2.11)$$

If a new term is introduced, the magnetising current vector i_{mr} where :-

$$i_{mr} = \Phi_r/M \qquad (2.12)$$

then eqn. 2.11 can be rewritten as :

$$i_{sd} = (L_r/R_r)d(i_{mr})/dt + i_{mr} \qquad (2.13)$$

For an induction motor operating with a constant rotor flux,

$$\Phi_{rd} = Mi_{mr} = Mi_{sd} \qquad (2.14)$$

and the torque equation becomes :-

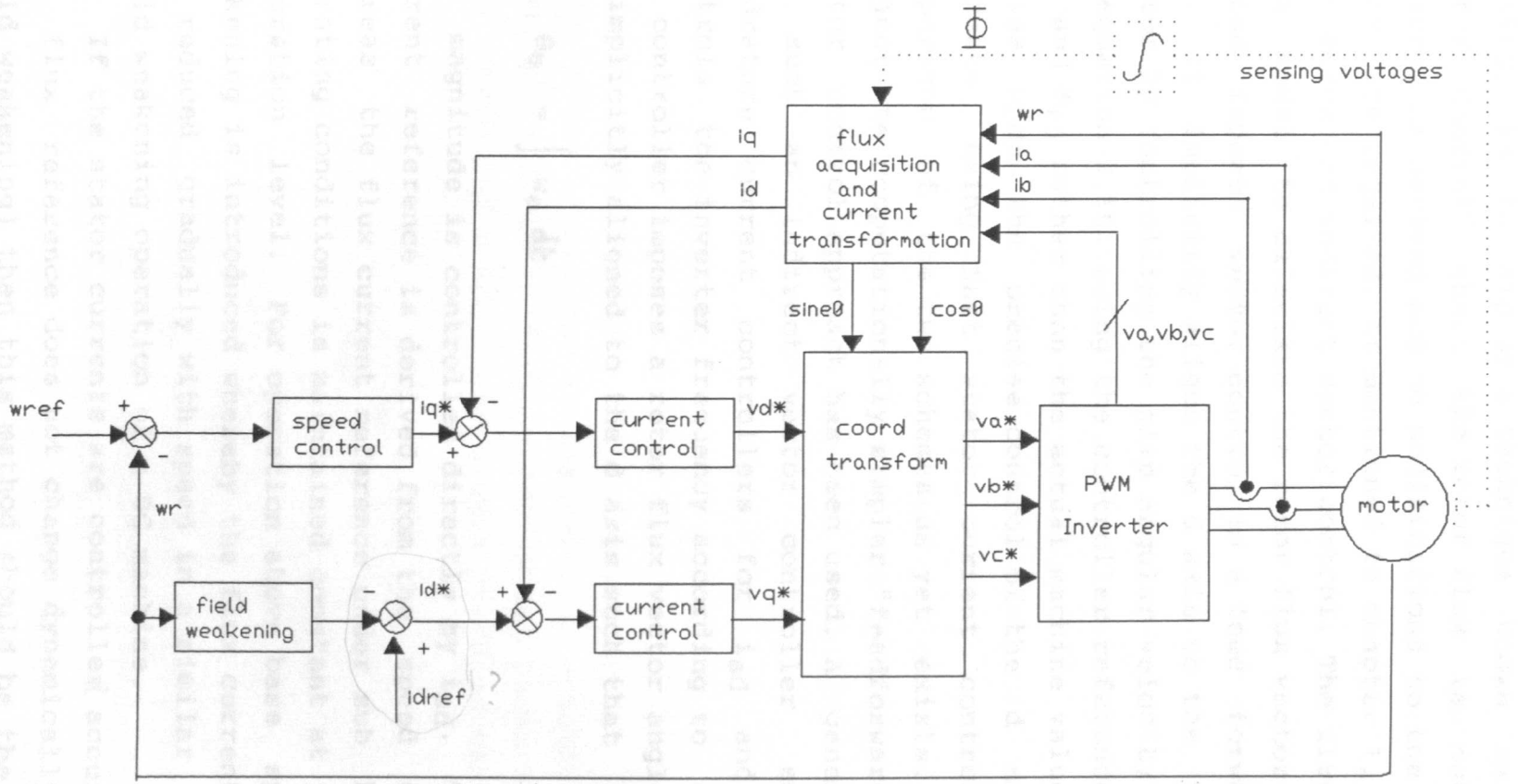
$$T_e = (2pM^2/3L_r)i_{sd}i_{sq} \qquad (2.15)$$

Equations 2.10 , 2.13 and 2.15 are the fundamental equations for vector control. Equation 2.13 is directly analogous to the field equation of a DC. machine where i_{sd} is the steady state field current and is directly proportional to the rotor flux. It is qualitatively analogous to the applied field voltage in the DC machine

and i_{mr} is analogous to I_f. Equation 2.15 corresponds to the torque equation of a DC. machine and thus i_{sq} is the torque producing component of the stator current. Under field weakening control eqn. 2.13 becomes important as the rotor flux, being proportional to i_{mr} through eqn. 2.12 responds to changes in i_{sd} with a significant time constant L_r/R_r.

2.5) IMPLEMENTATION OF VECTOR CONTROL

For implementation of vector control for induction motor drives some means for identifying the instantaneous position of the rotor flux vector is essential. Early experimentation with this technique employed flux sensing coils mounted in the stator slots [8] and provided a good flux measurement down to about 0.5Hz. This method however requires the use of modified or specially constructed machines, not considered generally practicable. A second method used in the early history of vector control used the stator voltages (either measured using sensing coils again or by using the reference values controlling a PWM generator) and by means of integration provided an estimate of the rotor flux vector [8]. This method however proved unreliable at low speed due to the cut off frequency of the integrators employed. Variations in the stator parameters due to temperature and saturation had a marked effect on the control scheme. Techniques employing direct sensing of the rotor flux signal are known as "Direct Vector Control". A general scheme for direct vector control using a voltage fed inverter is illustrated in Fig. 2.3. It is also possible to employ the same vector control scheme using a fast "local" current loop around the inverter.



Direct Vector Control of an Induction Motor

Figure 2.3.

An alternative and simpler approach to this control strategy is to employ a technique known as "Indirect Vector Control" where the rotor flux is not directly measured or derived and no modifications to the induction motor are required. As mentioned in chapter 1, there are two forms of indirect vector control. The first uses a flux model to calculate the rotor flux vector [6]. The second imposes vector control in a feed forward manner [9]. It implicitly aligns the d axis to the rotor flux vector by controlling the slip angular velocity according to equation 2.10. using the controller reference values of i_{sq} and Φ_{rd} rather than the actual machine values. It thus relies upon the precise control of the d and q axis currents using fast stator current controllers. No comparison of the two schemes as yet exists. For this project the computationally simpler "feedforward" indirect vector control approach has been used. A general scheme for such an indirect vector controller employs two quadrature current controllers for i_{sd} and i_{sq} and controls the inverter frequency according to eqn. 2.10. The controller imposes a rotor flux vector angle Θ_e which is implicitly aligned to the d axis such that

$$\Theta_e = \int w_e \, dt \qquad (2.16)$$

Its magnitude is controlled directly by i_{sd}. The torque current reference is derived from the speed controller, whereas the flux current reference under sub base speed operating conditions is maintained constant at just under saturation level. For operation above base speed field weakening is introduced whereby the flux current reference is reduced gradually with speed in a similar manner to field weakening operation of a DC machine.

If the stator currents are controlled accurately and the flux reference does not change dynamically (ie. no field weakening) then this method should be the same as if a flux model had been used.

The transformation of the instantaneous stator current values into field orientated d and q axis components is achieved using the equations given in Appendix C. This transformation will be here termed "demodulation". The equations employ matrix multiplication using the sine and cosine of the instantaneous flux vector angle, and this is abbreviated for the schematic diagrams by use of the equivalent complex operator $e^{j\theta e}$. The corresponding modulation routines (i.e. transformation of d and q axis values to instantaneous stator reference frame values) are also given in Appendix C, and are represented by the complex operator $e^{-j\theta e}$.

The precise control of the stator currents using a voltage source inverter can be achieved by one of three methods, each of which have been investigated experimentally. In this project these are termed V - Type control [9], V - Type control with current feedback [9],[10], and I - Type control [8].

2.5.1)V - Type Control

The basis of this control strategy is the steady state field orientated stator voltage equations derived from eqn. 2.3. [9]. If the differential terms are removed from rows 2 and 3 the d and q axis voltages can be defined as :-

$$v_{sd} = i_{sd}R_s - w_e\sigma L_s i_{sq} \qquad (2.17)$$

$$v_{sq} = i_{sq}R_s + w_e L_s i_{sd} \qquad (2.18)$$

The stator voltage demands are derived directly from these equations using the reference demands for i_{sd}, i_{sq} and w_e and imposed on the machine using a sinusoidal PWM generator which provides the required base drive signals to the inverter [14],[2]. This method does not employ closed loop current control, and is thus inherently sub-

optimal because it requires the precise knowledge of the stator and rotor parameters (which may change with temperature, frequency and saturation), and demands a correct calibration between the pwm voltage reference signals and the generated motor voltages. Its main advantage however is that it is easy to implement and requires only a speed feedback signal. The schematic for this controller is shown in Fig. 2.4.

2.5.2)V - Type Control with Current Feedback

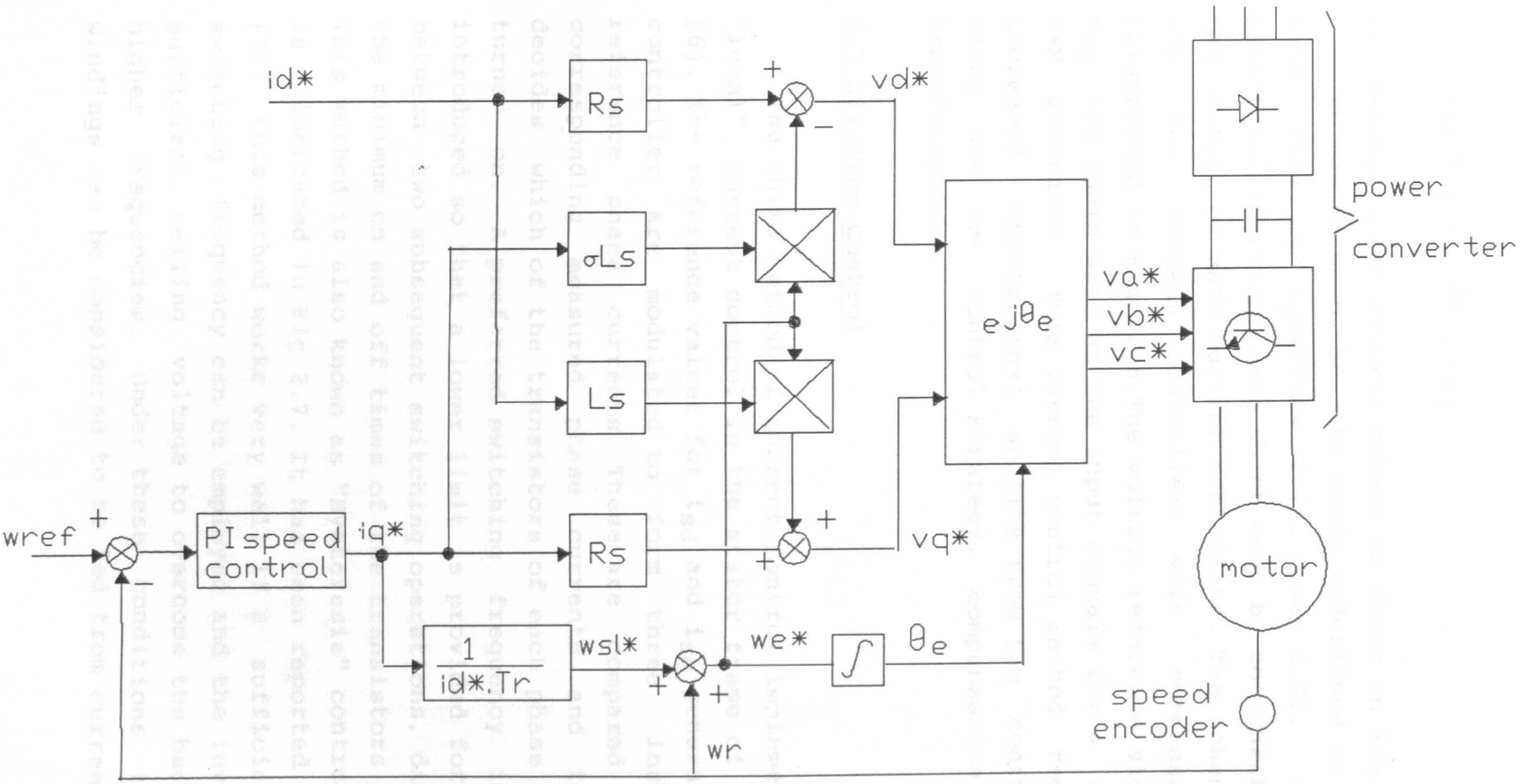
This method employs current feedback to provide closed loop control of the d and q axis currents and is based on the stator dynamic equations derived from eqn. 2.3 namely [10] :-

$$(\sigma L_s/R_s)d(i_{sd})/dt + i_{sd} = v_{sd}/R_s - (1-\sigma)(L_s/R_s)d(i_{mr})/dt + (\sigma L_s/R_s)w_e i_{sq} \qquad (2.19)$$

$$(\sigma L_s/R_s)d(i_{sq})/dt + i_{sq} = v_{sq}/R_s - (1-\sigma)(L_s/R_s)w_e i_{mr} + (\sigma L_s/R_s)w_e i_{sd} \qquad (2.20)$$

The current controllers themselves were designed using Laplace techniques and employed proportional plus integral control. The controllers were designed on the basis of the following equations :-

$$v_{sd} = (R_s + S\sigma L_s)i_{sd} \qquad (2.21)$$



Indirect Vector Control using Impressed Voltages and Open Loop Stator Dynamics Compensation (V Type Control)

Figure 2.4.

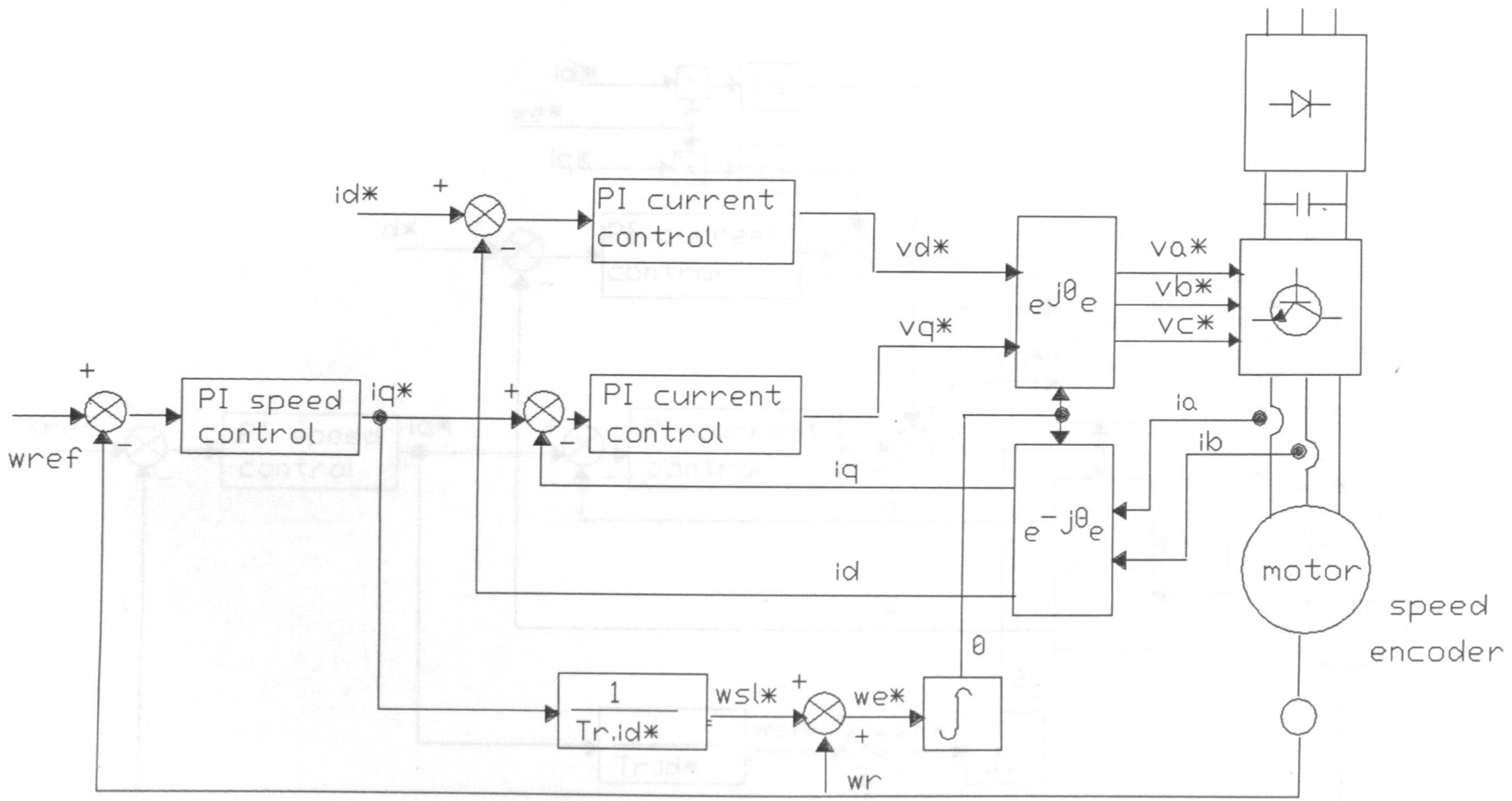
$$v_{sq} = (R_s + S\sigma L_s)i_{sq} \qquad (2.22)$$

The overall motor control scheme is shown in Fig 2.5.

It can be seen that the terms underlined on the right hand side of equations 2.19. and 2.20. constitute additional coupling terms which must be compensated for at the output of each current controller. The schematic for the two current controllers with compensation is illustrated in Fig. 2.6.The voltage reference signals v_{sd}, v_{sq} are again used as the input signals for a sinusoidal PWM generator. This current control method requires an increased computational ability from the controller to carry out the control strategy, compensation and the transformations.

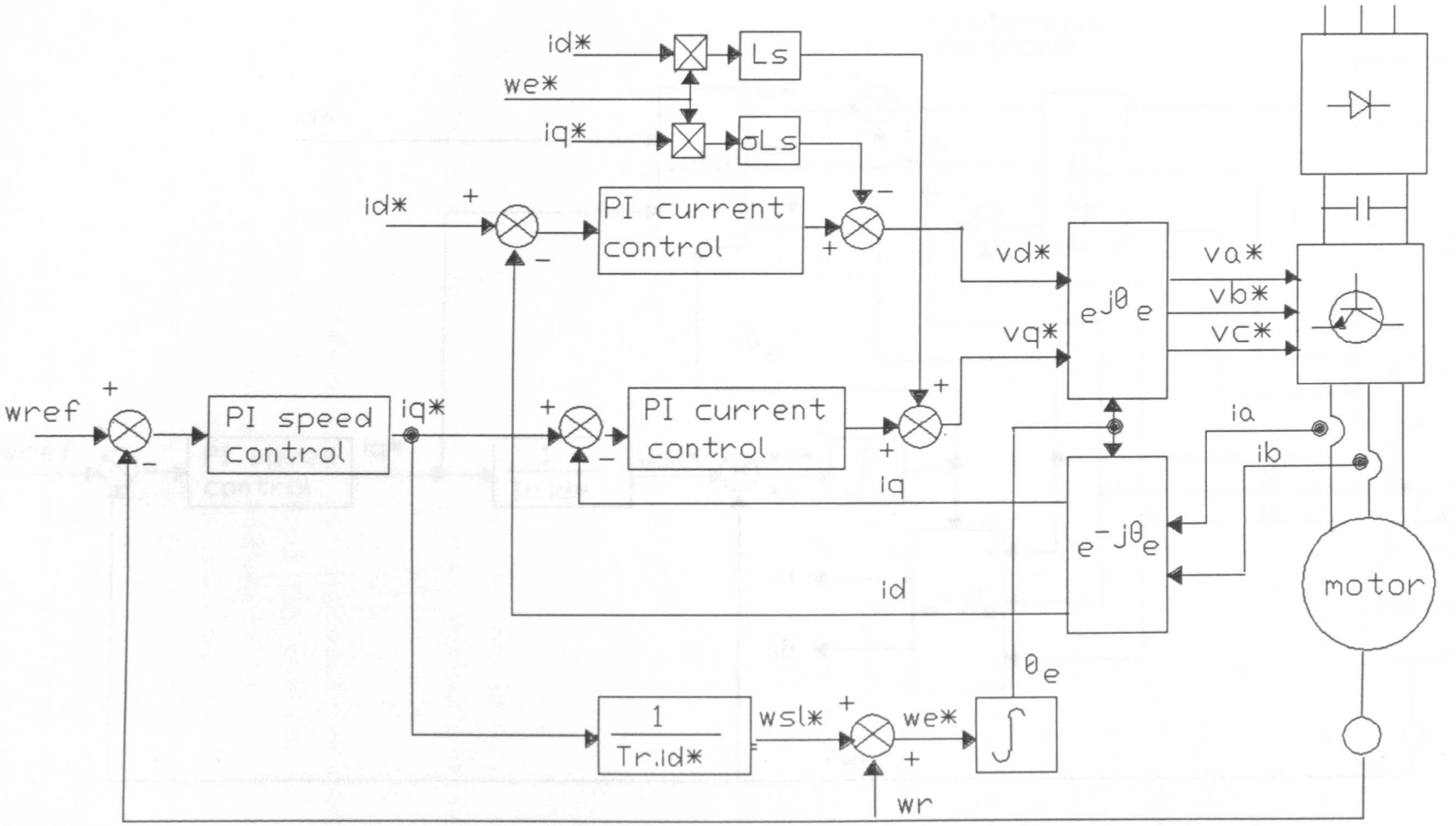
2.5.3)I-Type Control

The third method of current control implements a fast "local" current control in the stator frame of reference [6]. The reference values for i_{sd} and i_{sq} generated by the controller are modulated to form three instantaneous reference phase currents. These are compared with the corresponding measured phase currents and the result decides which of the transistors of each phase should be turned on. A preferred switching frequency is usually introduced so that a lower limit is provided for the time between two subsequent switching operations, dictated by the minimum on and off times of the transistors employed. This method is also known as "Hysteresis" control [6] and is illustrated in Fig 2.7. It has been reported that [6], [15] this method works very well if a sufficiently high switching frequency can be employed and the inverter has sufficient ceiling voltage to overcome the back EMF at higher frequencies. Under these conditions the stator windings can be considered to be fed from current sources.



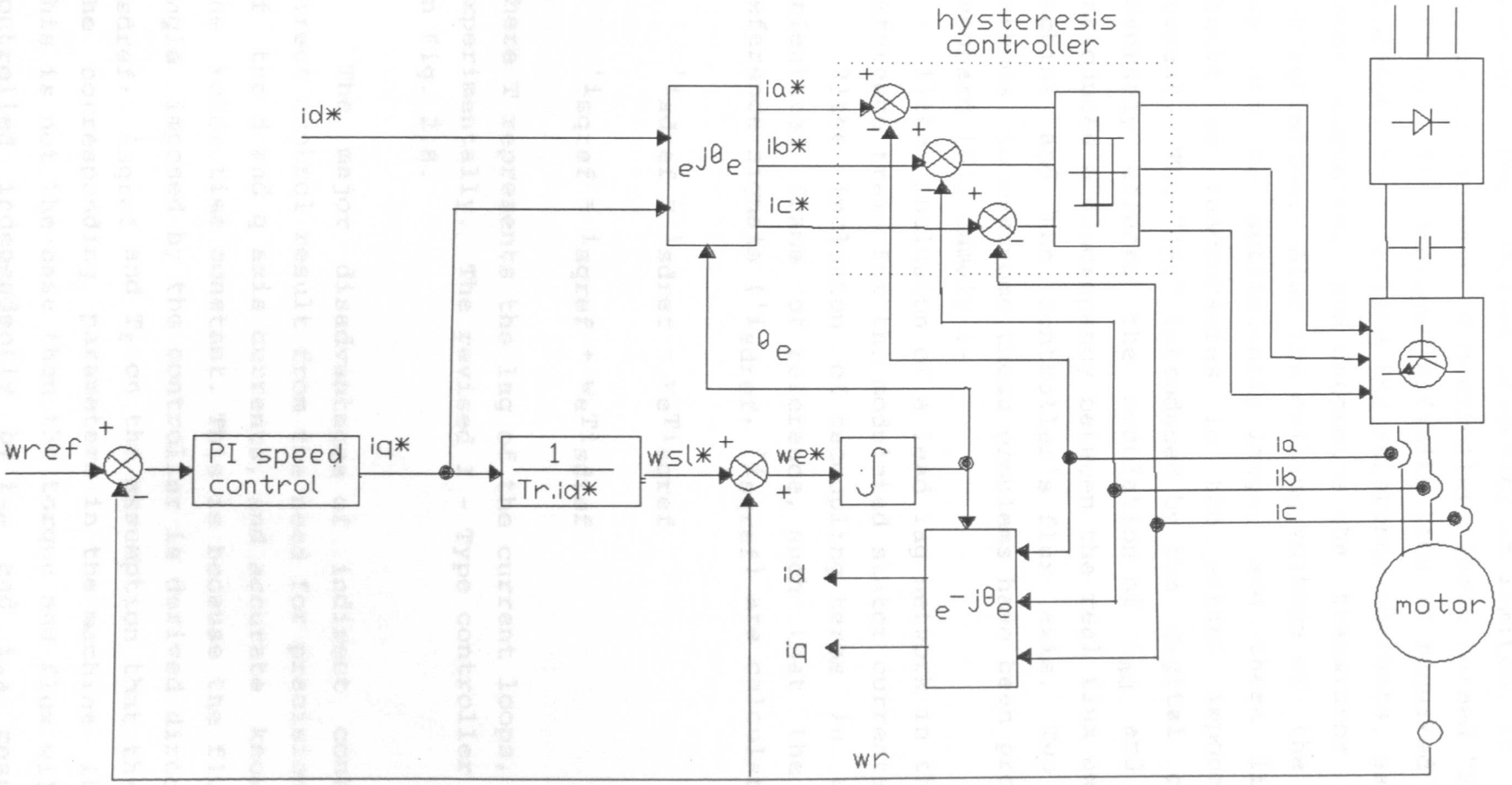
Indirect Vector Control Using Impressed Voltages and Control of Field and Torque Current Components (V Type with Current Feedback)

Figure 2.5.



V Type with Current Control and d-q axis Compensation

Figure 2.6.



Indirect Vector Control Using Impressed Stator Currents (I Type)

Figure 2.7.

However the carrier frequency is limited. If a purely digitally implemented controller, (here termed "Bang-Bang" control) is employed a finite time is required for the processor to calculate the reference currents, sample the motor currents, and determine the transistor switching configuration. Also the ceiling voltage of the inverter may not be sufficiently large, and these limitations result in inaccuracies in the actual imposed stator current. The "lag" introduced by the digital controller adversely affects the modulation of i_{sd} and i_{sq} and introduces a discrepancy between the real flux axis in the machine and the controller's flux axis. Two possible methods to overcome these problems have been proposed by Leonhard [6], namely :-

1)the inclusion of a lead lag network in the stator reference frame for the modulated stator currents

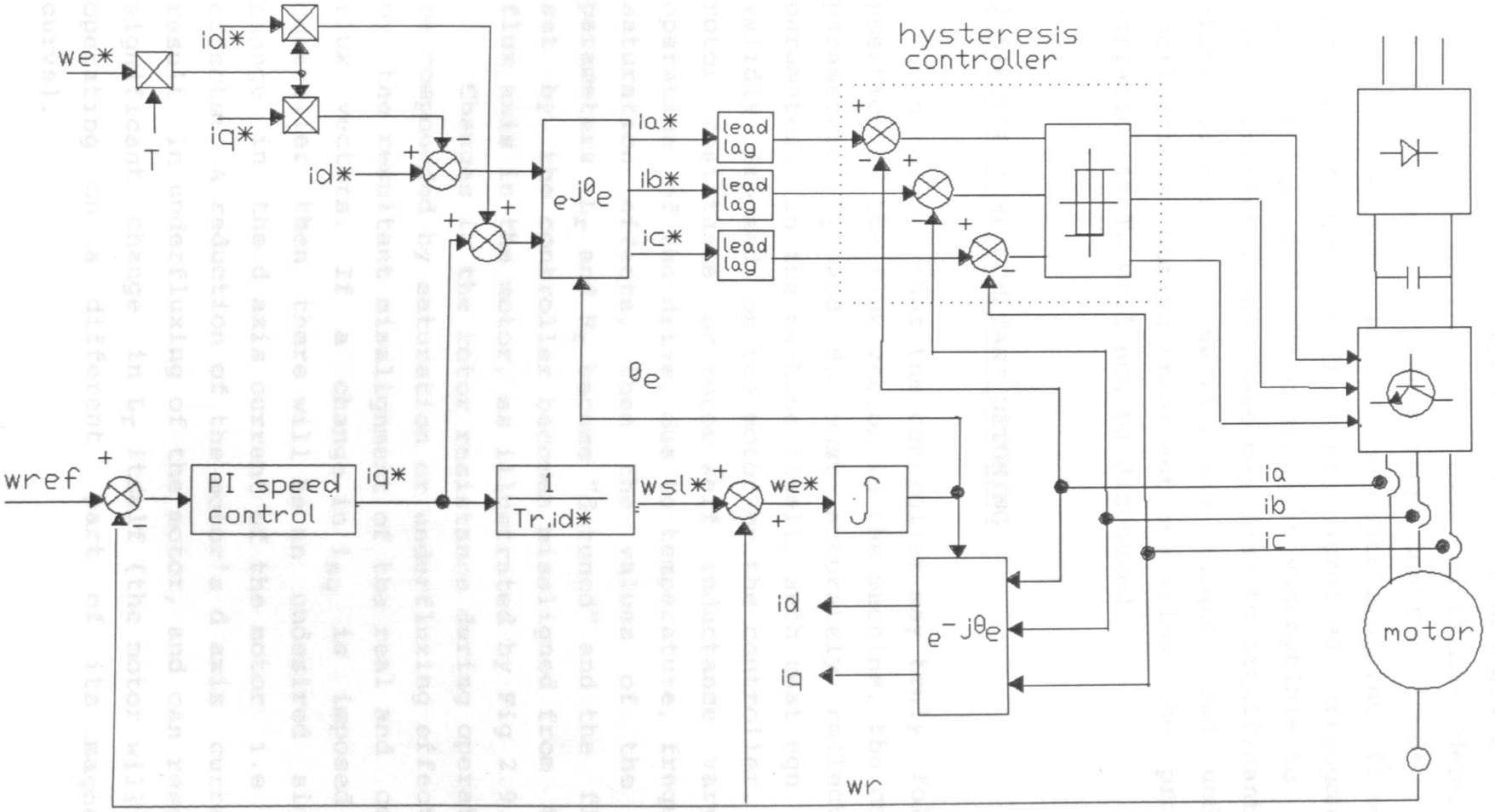
2)the inclusion of decoupling terms in the field orientated frame of reference, such that the adjusted reference signals ($'i_{sdref}$, $'i_{sqref}$) are calculated as :-

$$'i_{sdref} = i_{sdref} - w_e T i_{sqref} \qquad (2.23)$$

$$'i_{sqref} = i_{sqref} + w_e T i_{sdref} \qquad (2.24)$$

where T represents the lag of the current loops, obtained experimentally. The revised I - Type controller is shown in Fig. 2.8.

The major disadvantages of indirect control over direct control result from the need for precision control of the d and q axis currents, and accurate knowledge of the rotor time constant. This is because the flux vector angle imposed by the controller is derived directly from i_{sdref}, i_{sqref} and T_r on the assumption that these match the corresponding parameters in the machine itself. If this is not the case then the torque and flux will not be controlled independently by i_{sq} and i_{sd} respectively.



I Type Control with d-q Axis and Stator Lead Lag Compensation

Figure 2.8.

Direct control is somewhat robust to changes in the rotor time constant during operation, and is not dependent on the performance of the current control loops. This is because there is closed loop control of the flux vector and thus torque and flux are controlled orthogonally by i_{sq} and i_{sd}. Indirect control is susceptible to both of these problems, and these may lead to significant steady state errors in the flux and torque, and undesirable oscillations during transient operation. The problem of variation in T_r will now be discussed.

2.6)ROTOR TIME CONSTANT DETUNING

In order that the controller may truly follow the position of the flux vector in the machine, the controller parameters L_r and R_r must accurately reflect those parameters in the machine itself, such that eqn 2.10 is validly imposed on the motor by the controller. If the rotor resistance or rotor self inductance vary during operation of the drive, due to temperature, frequency or saturation effects, then the values of the control parameters L_r and R_r become "detuned" and the flux axis set by the controller becomes misaligned from the real flux axis in the motor, as illustrated by Fig 2.9.

Changes in the rotor resistance during operation may be compounded by saturation or underfluxing effects caused by the resultant misalignment of the real and controller flux vectors. If a change in i_{sq} is imposed by the controller then there will be an undesired significant change in the d axis current of the motor i.e coupling effects. A reduction of the motor's d axis current will result in underfluxing of the motor, and can result in a significant change in L_r itself (the motor will now be operating on a different part of its magnetisation curve).

Angular Relationship of the Current
Vectors when "Detuning" occurs.

Figure 2.9.

The effect of detuning on the torque and flux response in the induction motor has been described as follows. Garces [11] provides analytical expressions to relate the steady state error in the real d axis flux magnitude and real q axis flux magnitude (which should be zero) to δT_r - the discrepancy between the controller value for T_r and the actual machine value - and the slip frequency. Matsuo and Lipo [12] describe the torque and flux vector deviation due to variation of rotor resistance, suggesting that the percentage deviation in detuning of T_r results in a similar percentage deviation in the rotor flux vector and expected steady state torque. Sugimoto and Tamai [33] describe the effect on the transient response of the motor due to detuning, showing that the d axis flux, q axis flux and developed torque have transient terms varying with T_r for a step change in torque demand. The expressions are experimentally verified. Krishnan and Doran [16] describe two separate cases of detuning:- error in rotor resistance due to temperature effects and error in magnetising inductance caused by saturation in the machine, and provide analytical expressions to relate the parameter sensitivity to the steady state and dynamic operation of the drive system. Ho and Sen [17] describe three different control structures based on the stator flux vector, rotor flux vector and airgap flux vector, and conclude that parameter sensitivity is similar for all three methods.

The net conclusion from these papers is that an on line method for T_r identification is essential for correct implementation of an indirect vector controller, particularly for T_r deviation due to temperature and frequency effects, but preferably also to estimate changes in L_r during changes in flux levels such as field weakening. Matsuo [12] and Sugimoto [18] describe methods for T_r identification using extra sensory coils. Matsuo [12] employs negative sequence current injection and measures the resultant negative sequence voltage using

sensing coils, in order to identify R_r. Sugimoto [18] employs a Model Reference Adaptive Strategy but again requires a modified machine which employs search coils, and it seems that these coils may be better employed as a direct vector controller.

The two methods for T_r identification employed in this project were chosen because they did not require modifications to the induction motor. The first method is here termed "Reactive Power Measurement" and the second "PRBS Injection".

2.6.1)Reactive Power Measurement

Garces [11] proposes a method of comparing demanded and measured reactive powers. He relates the discrepancy to a δT_r function and provides analogue computer simulations to justify the method. Krishnan [19] and Koyama [13] describe experimental implementations of this algorithm with limited success, both in steady state and transient T_r identification. The basis of this technique is the derivation of a function which gives the necessary information on the position and magnitude of the flux vector. The function (Fo) itself is derived from the reactive power expression ([11] and [13]) and can be calculated in two ways. The first method uses the demanded values for v_{sd} and v_{sq} (assuming that these are correctly imposed on the motor) and measured values for i_{sd} and i_{sq} to calculate Fo as :-

$$Fo = v_{sd}i_{sq} - v_{sq}i_{sd} + \sigma L_s w_e(i_{sd}^2 + i_{sq}^2) \quad (2.25)$$

A corresponding function (Fo*) can be calculated from the controller reference values :-

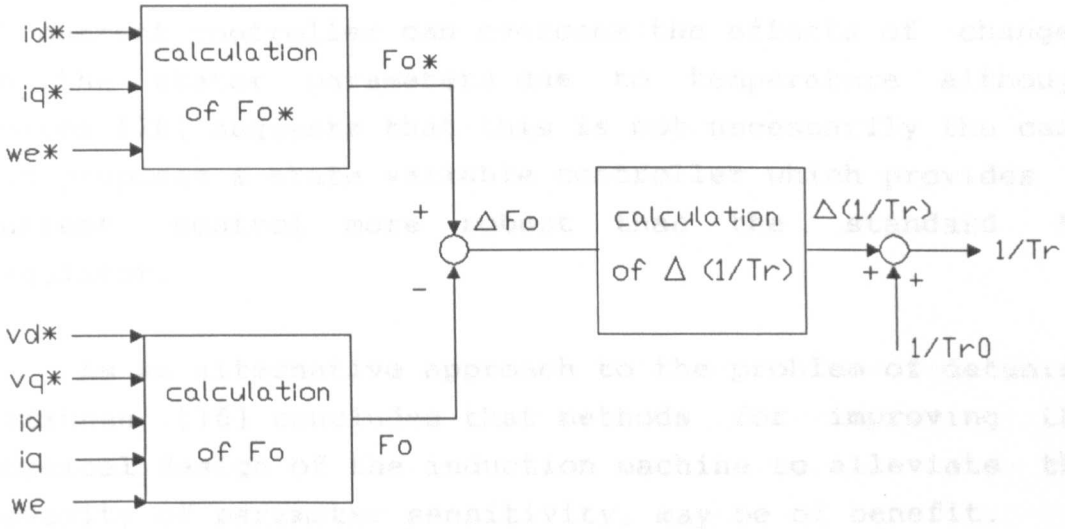
$$Fo^* = - w_e(M^2/L_r)i_{sdref}^2 \quad (2.26)$$

The calculation of δT_r is based on the error between Fo and Fo* and provides a correction term for the value of T_r used by the slip frequency controller. The block diagram for this control scheme is illustrated in Fig. 2.10. This scheme works in parallel to the speed control and will only work if the slip frequency and the stator frequency are not zero. It will identify changes in rotor resistance, but does not consider changes due to under-fluxing or over-fluxing.

2.6.2)PRBS Injection

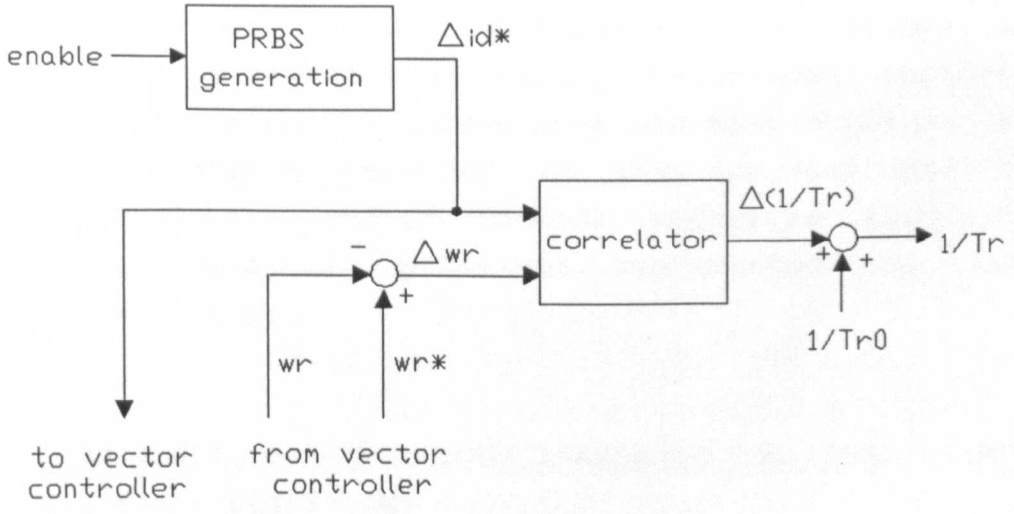
Leonhard [6] proposes the injection of a Pseudo Random Binary Signal (PRBS) onto the d axis current reference demand. The clock frequency of the disturbances is chosen to be sufficiently high to avoid direct influence on the machine torque due to deviations in i_{mr}. Indirect disturbances on the machine torque are seen as disturbances on the motor speed, and these indicate that there is a discrepancy between the demanded and actual rotor flux angle Θ_e, due to detuning. Cross-correlation of the injected and measured disturbances gives an estimate of the error function δT_r. The block diagram illustrating this identification scheme is illustrated in Fig 2.11. This method is again only useful for identification of temperature changes in T_r.

These methods require background computational ability from the controller to carry out disturbance injection and T_r update. The main advantage of these strategies however is that no extra sensory devices are required. Assuming that the motor voltage and current references are correctly imposed on the machines using V - Type or V - Type with current feedback control, then the controller reference signal can be used for identification. Krishnan [19] describes an accurate method for calculating the instantaneous motor line voltages from



Rotor Time Constant Identification using
Reactive Power Measurements

Figure 2.10.



Rotor Time Constant Identification using
PRBS Injection

Figure 2.11.

the inverter base drive signals. It is assumed that a fast PI current controller can overcome the effects of changes in the stator parameters due to temperature although Lorenz [20] suggests that this is not necessarily the case and proposes a state variable controller which provides a current control more robust than the standard PI regulator.

As an alternative approach to the problem of detuning Krishnan [16] concludes that methods for improving the physical design of the induction machine to alleviate the severity of parameter sensitivity, may be of benefit.

CHAPTER 3

PARALLELISM AND MOTOR CONTROL

3.1) INTRODUCTION

This chapter outlines the proposed implementation of the transputer network to the vector control of an induction motor. The first section outlines the range of products available for implementing digital control strategies to complex real time control applications. The second section describes the parallel nature of motor control. The next section outlines several implementations of multiprocessor controllers to vector control algorithms to demonstrate the effectiveness of this approach and illustrates the disadvantages of using conventional processors for this application. The final section introduces the transputer as a device designed specifically for parallel processing applications and compares it to parallel processing techniques employing conventional sequential processors. It then outlines the transputer network employed in this project and the breakdown of the vector control algorithm into the individual parallel processes implemented on each transputer.

3.2) MICROPROCESSOR TECHNOLOGY AVAILABLE FOR REAL TIME CONTROL APPLICATIONS

The recent development of microprocessor technology has been such that the control engineer now has a large selection of processor hardware from which to choose. The substantial increase in processor speed has made possible the digital implementation of control strategies such as

optimal control and adaptive control for induction motors, which were once of only theoretical value. The advent of Very Large Scale Integration (VLSI) means that processor design is no longer a restriction and dedicated processors for drive control can now be manufactured.

The range of processor hardware to handle complex real time control problems can be said to fall into the following categories:-

1) Single conventional processor: In its baseline form this constitutes the memory and arithmetic unit of the control system. A high interface chip count together with an external coprocessor or analogue processing is usually necessary if fast real time processing is required.
2) Microcontroller: Control orientated processor in which common interface components such as Analogue-to-Digital converters, Digital-to-Analogue converters, timers, interrupt controllers and pulse width modulated outputs are carried on board chip thus reducing off processor chip count.
3) Application Specific Integrated Circuits (ASICS): An extension of (2) which minimises off processor chip count and on-chip silicon for a specific application.
4) Digital Signal Processor: A high-speed development of (1) utilising high clock rates and reduced instruction set on-chip architecture for fast multiply-shift-add operations.
5) Concurrent or Parallel Architectures: Either a number of conventional processors or specific parallel processors are used in an effort to obtain increased speed and/or flexibility of control processing.

For real time control problems of low to medium complexity, microcontrollers and ASICS are principally exploited for direct commercial application. For the "modern", state variable based control problems, with matrix manipulation based strategies, the DSP chips may be

thought to be very attractive with their pipe-lined architecture. Parallel architectures, and particularly the Transputer still however remain largely unassessed for real time control. Superficially they offer an increase in speed and hence an increase in the degree of processing within the control system sample time. It is equally intuitive that two or more processes operating in parallel may continuously require each other's data so that the processing may degenerate into "all talk and no work". This is a significant factor in the assessment of the suitability of parallism for a motor drive control.

3.3)THE PARALLEL STRUCTURE OF MOTOR CONTROL

A general motor drive tasking schemat is shown in Fig. 3.1. The low level consists of signal input, conditioning, control processing and actuator output for the power converter. This level corresponds to the conventional "interrupt" processing that must be serviced for the drive function. For example in DC drives this equates to speed, armature current and field current measurement, speed and current control, and then the production of the firing pulses for use with say a 3 phase bridge converter. A switched reluctance drive requires speed and current measurement, control calculation and then suitable energisation of the required phases. In a basic slip controlled induction motor drive, again speed measurement is required, calculation of the slip frequency and voltage amplitude is then performed, and then production of the transistor base drives for say a PWM inverter.

The next level covers the drive's intelligence, supervision and memory. Intelligence is used to describe on-going background calculations, the principle examples being parameter estimation procedures for a partly or wholly undefined drive system. The supervision function

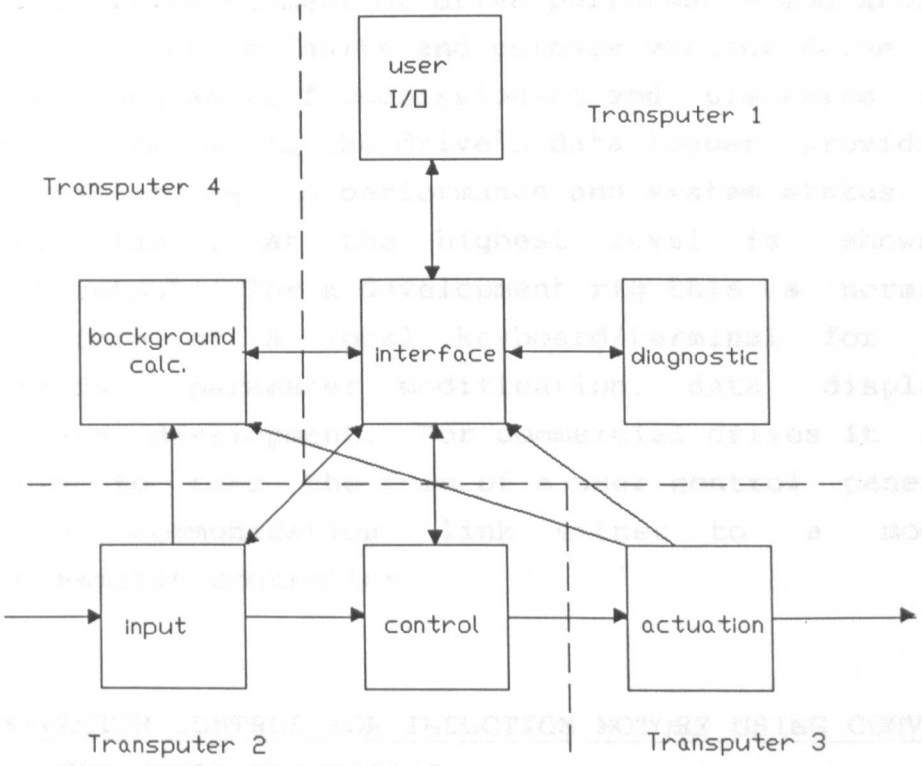


Figure 3.1. Tasking Schemat for Motor Control
(Transputer Mapping for Vector Control)

covers the assessment of drive performance and arbitration with respect to faults and perhaps various drive control modes such as self commissioning and piecewise control. System memory is the drive's data logger providing the information as to performance and system status upon a drive fault. At the highest level is shown user input/output. For a development rig this is normally of the form of a local keyboard/terminal for on-line commands, parameter modification, data display and software development. For commercial drives it is more likely to take the form of a user control panel or a remote communication link either to a modem or programmable controller.

3.4)VECTOR CONTROL FOR INDUCTION MOTORS USING CONVENTIONAL SEQUENTIAL PROCESSORS

The development of vector control for induction motors has utilised parallel processing techniques which use networks comprised of "conventional" sequential processors. Gabriel [8] describes a multiprocessor "direct" vector controller for a current controlled voltage source inverter which uses two Intel 8085 microprocessors. One performs the speed, torque and flux control algorithms and input /output routines. The second performs the flux detection (using sensing coils mounted within the machine) and current control with co-ordinate transformations. Harashima [9] describes the implementation of two "indirect" control structures based on three microprocessors :- two i-8031 chips are assigned to the 3-2 and 2-3 phase transformations and an i-8086 performs vector and speed control. For both of these structures the signal actuation for the power converter are not derived from the microprocessors. Kubo [10] describes "indirect" vector control using four 16bit 8086 microprocessors for speed control and slip frequency

calculation, current control and vector calculation, current component and speed detection, and pwm wave form generation. Other examples of multiprocessor vector control can be found in [17] and [6].

These implementations have proved the effectiveness of applying parallel processing architectures to vector control of induction motors. It is obvious that the major consideration when applying parallel processing techniques to a control algorithm is the breakdown of that algorithm into subroutines (processes) which can run concurrently (at the same time). It is also apparent that the actual processor time for each of these processes will differ, resulting in some "redundancy" of the individual processors. It is necessary to strike a balance therefore between the reduction of this process redundancy, and obtaining a low process time per sample of information. A further consideration in this respect arises from the amount of inter-process communication required by the algorithm, as experience has shown that this is the principle factor in determining "processor performance". In an attempt to lend a quantitative measure to parallel processor performance for motor control the following parameters are defined for each process within the system [46]:

t_c: the time taken to transfer one byte of data from one parallel process to another.

t_s: the communication period whose inverse is the frequency at which inter-process communication takes place

t_t: the total process communication time within t_s

t_p: the process time for a given parallel process, including communications.

The times t_s and t_p are application specific. Two ratios are defined: $\alpha = t_t/t_s$ which is a measure of computational overhead incurred in making an n processor system/subsystem work to achieve the required

system/subsystem performance, and ß = t_p/t_s which is a measure of the processor utilisation. Expressed as percentages, α should be low whilst ß should be high.

The major hardware design problem to be overcome by the previously described systems has been the organisation of inter - process communication. Perhaps the most basic conventional approach to inter process communications using conventional microprocessors consists of a processor with access to local memory, the bus being connected to a network bus via a bi-directional buffer or crossbar switch. Communication involves a second processor holding the processor of the receiving system, enabling the buffer, writing to the local memory, disabling the buffer, and releasing the held processor which can then read the local memory. If controlled in software this gives a worst case t_c between 2µs and 5µs. A minimum t_c for a conventional bussed system is achieved with dual port RAM in which communication proceeds within a successive read/write cycle by the send and receive processors respectively. This can be as little as 200ns for the latest processors. Leaving aside the high cost of the dual port RAM it can be shown that parallel bussed architectures suffer from the following disadvantages:-

1) arbitration logic establishing a priority for sending and receiving processes must be implemented in hardware, thereby increasing design complexity and chip count. Arbitration logic must also be present when a processor has access to two or more dual port RAMS.
2) the hardware complexity reduces the flexibility with regard to system expansion.
3) multiprocessor bussing imposes severe board layout problems, high tracking densities, multilayering and corresponding high board costs.
4) any alleviation of the hardware and tracking complexities result in hierarchical systems involving master-slave architectures eg. Bose [3] describes a

"mailbox" approach to inter - process communication and a further alternative in the form of a "radial" approach to the multiprocessor network, whereby data transfer is achieved via a "supervisor" processor. This reduces the options for task assignments.

5) software development is cumbersome, the user generally responsible for code to download programs to the appropriate processor.

3.5) TRANSPUTER IMPLEMENTATION OF A PARALLEL PROCESSING NETWORK FOR VECTOR CONTROL

3.5.1) The Transputer

The Transputer is available as either a 16 bit (T212/T222) or 32 bit (T414/T425) microcomputer and is specifically designed for parallel processing. The device itself contains a medium speed processor with an integer word multiply time of 2.5µs, 2K Bytes of RAM, a memory interface to access external local memory, two on board timers, four bi-directional serial links for communication with other transputers or transputer look-alike peripherals, and an event pin for use with external interrupts. Data transfer using the links takes place at a nominal 10M bits/second, which results in a transfer time, as measured for the T212 Transputer, of $t_c \approx 5\mu s$ per byte. This is slow and must be considered as a significant factor in the design and assessment of transputer implemented algorithms. A full description of the transputer is given in Appendix D.

A transputer based parallel processing network suffers from none of the disadvantages associated with parallel bussed architectures. The serial link arrangement results in a true grid modular structure in which inter-processor links are by the standard four wire connection. The hardware structure itself is implicitly non-

hierarchical. Communication priorities and the parallel and hierarchical planning are designated to user defined software and system configuration using respectively, the transputer's high level programming language, OCCAM, and the Transputer Development System(TDS). It is worth emphasising that these two system utilities are as instrumental in producing a flexible parallel processing system as the transputer hardware itself. The OCCAM language was developed specifically for parallel processing and is based on modular process modelling in which a process is defined as an independent and self-sufficient software routine. Several processes can be defined to run concurrently with inter process communication being achieved using defined communication channels. In addition to the usual arithmetic, boolean, conditional and repeat statements, OCCAM also allows the designation of priority to particular processes. The TDS provides the environment for editing, compiling, and configuring the processes to run on the transputer network. Following the separate compilation of the individual processes, a high level editor is invoked to assign a particular process and communication links to each transputer or peripheral in the network. A full description of the OCCAM language and the TDS is given in Appendix D.

The above is illustrative of the benefits of the transputer system and its supporting utilities over a parallel architecture using conventional processors. The OCCAM structure is conducive to a systematic and disciplined planning of parallel algorithms with the resulting low system development time. The hardware is remarkably user-transparent, easily expanded and, considering the complexity of the devices, has a low board cost. Furthermore 16 bit, 32 bit and floating point (T800) transputers are directly interconnectable, and slave processors (such as DSP's which may be more suitable for intensive vector computation or modern control than a

transputer) can be interfaced to the transputer network. The fact that different computer architectures may be combined under transputer control illustrates the powerful modularity of the system. The disadvantages are mainly the high cost per chip, and the long byte communication time t_c. It has been noted however that the value of t_c of less than 5μs has been reported for T800 processors operating at 20Mbits/s communication rate.

3.5.2)The Transputer Parallel Processing Network

The parallel implementation strategies for the vector control schemes described in chapter 2 are derived from the tasking philosophy of Fig. 3.1. The philosophy rests on the assumption that since the transputer inter-process communication time is comparable, if not longer than the arithmetic operation times, mathematical operations such as background calculations, control processing and co-ordinate transformations should not be parallelized. The parallelism operates upon the strategic functions within the overall controller, all of which operate on the data pertaining to each sample instant. Delays incurred through any pipelining of sample data through successive strategic operations (eqn. between the control and actuation processes) can if necessary be accounted for by the z-transform delay operator. The chief consequence of the parallelization employed is that the "low level" processes execute sequential procedures: there are no parallel constructs within a process. All statements and procedures within a process are timed to execute within the system sample time t_s. This sample time is obtained from one transputer acting as "synchronisation master", and is communicated to other transputers via communication protocol. It thus follows that the communication statements themselves must be sequentially constructed: no work can be done whilst the transputer is waiting to communicate. In a sense this is intuitive: a process which

takes longer than t_s will result in loss of synchronisation, whilst a process which takes less than t_s does not need to utilise the communication wait time. The only instance when a communication is not sequentially constructed is when an input is totally asynchronous, ie. user input from the host to the supervising transputer. Here the transputer does not know when a user input will occur. The user input statement is merely "paralleled" with a timer process using an OCCAM ALT construction (see Appendix D.): if no input communication occurs within a set time, the input is ignored and the rest of the process continues. The time spent waiting is obviously a component making up the time t_s of the supervisor.

The breakdown of the vector control algorithm into parallel processes running on the individual transputers (T1 - T4) is also illustrated in Fig 3.1. The transputer network employed is described in Appendix D. The overriding consideration is that the communication time t_s must be equal to the sample time corresponding to the fastest closed loop dynamics of the drive itself. The fastest drive dynamics invariably relate to the motor drive's current or torque feedback loops which for power drives is generally not less than 2ms. It was found that the calculation routine for the actuation signal generation, ie. pwm calculation or bang - bang control required a significant process time, and this was thus solely assigned to a T212 processor (T3) , and dictated the system sample time t_s (ie. acted as synchronisation master). For this project t_s was chosen to be 250µs, primarily as this was the minimum time required to execute the actuation calculations for the bang - bang controller on T3. It was then found that the signal acquisition, speed control, vector control calculations and current control could be implemented on a single T212 transputer (T2) running in parallel. The control calculations would be initiated by communication with T3 and would be completed before the next communication with T3 to ensure

synchronisation of the processes. The timing and synchronisation are discussed in detail in chapter 5. The supervisory routine for the drive was implemented on the T414 (T1) so that the 32 bit range of the device could be exploited for signal analysis, and the 2M Byte memory array used for data capture. Connected directly to this is the host machine which provides user interaction, and access to the hard disk for permanent data storage, graphics and plot utilities. The third T212 processor (T4) is also connected directly to the T414 transputer and provides the drive's background computational ability for parameter identification.

CHAPTER 4

THE TRANSPUTER NETWORK - INVERTER INTERFACE BOARDS

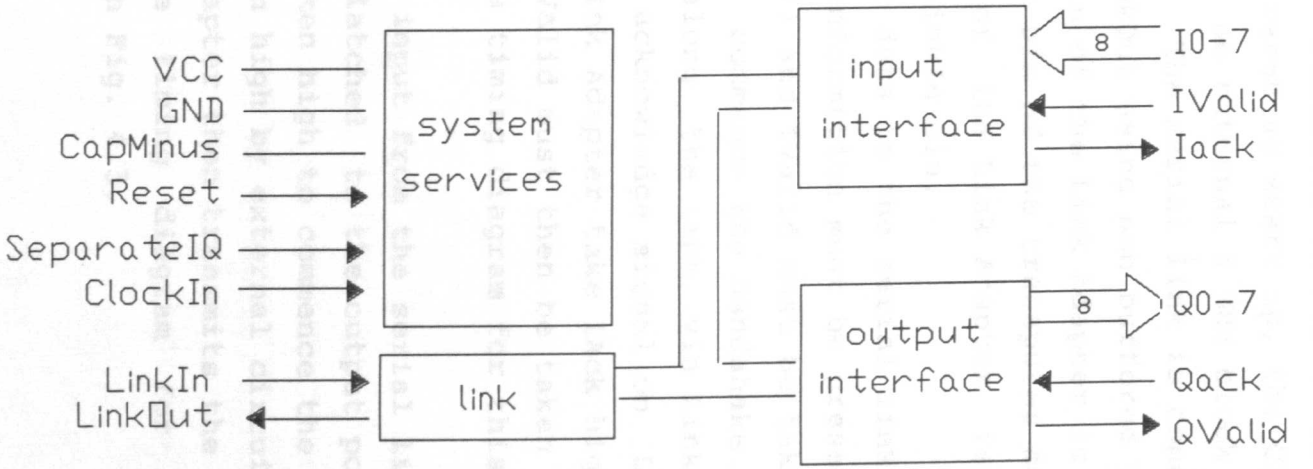
4.1) INTRODUCTION

This chapter will outline the design and operation of the interface circuits used for motor speed and current measurement, and for base drive signal generation for the inverter. The philosophy behind the design of external circuits which have to interface with the transputer network is that they should "appear" to the network as parallel processors, which communicate to the network via standard transputer serial communication links.

4.2) THE C011 LINK ADAPTER

The communication interface for the peripheral devices on the external interface boards used in this project has been the INMOS IMS C011 Link Adapter [21], a device which acts a transputer "look alike" peripheral when connected to a transputer via a link. The C011 can be configured to operate in two modes. In Mode 1 it has two byte wide parallel ports, one for input, one for output. Serial information transmitted from the transputer is converted to latched parallel output, and conversely latched parallel input can be transmitted to the transputer in correct serial form. In Mode 2 it provides an interface between the serial communication link and a microprocessor bus via a bi-directional byte wide port. All of the Link Adapters employed in this project have been used in Mode 1.

The schematic diagram for the C011 configured for Mode 1 is illustrated in Fig. 4.1.



INMOS IMS C011 Link Adapter Block Diagram

Figure 4.1.

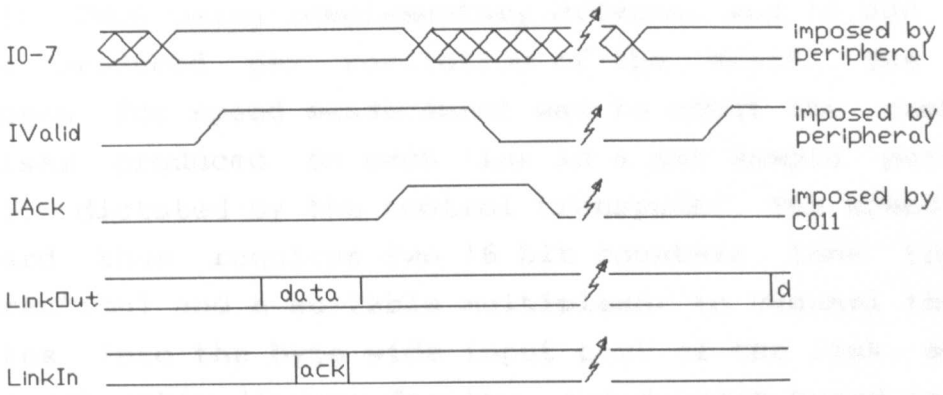
The control logic consists of the SeparateIQ pin, connected to ground to select Mode 1 with a link communication speed of 10 Mbits/sec, the Reset pin, connected to a momentary action switch to allow the Link Adapter to be reset at start up, the ClockIn pin which is connected to an external 5 MHz clock circuit, and the CapMinus pin. The serial link is connected directly to LinkIn and LinkOut using non buffered twisted pair cables. The LinkIn pin of the Link Adapter is directly connected to the LinkOut pin of the transputer data source, and the LinkOut pin of the Link Adapter is connected to the transputer's LinkIn pin.

To output data to the serial link from the interface board, the information must be presented to input port (pins I0 - I7) and IValid must be taken high by external circuitry to commence the handshake. The data is then transmitted along the link, via LinkOut, and the link provides an acknowledge signal on LinkIn. This signal makes the Link Adapter take IAck high for a prescribed period, and IValid must then be taken low to complete the handshake. The timing diagram for this procedure is shown in Fig. 4.2.

Data is input from the serial link via LinkIn, and immediately latched to the output port (pins Q0 - Q7). QValid is taken high to commence the handshake and Qack must be taken high by external circuitry to acknowledge. The Link Adapter then transmits the acknowledgement on LinkOut. The timing diagram for this procedure is illustrated in Fig. 4.3.

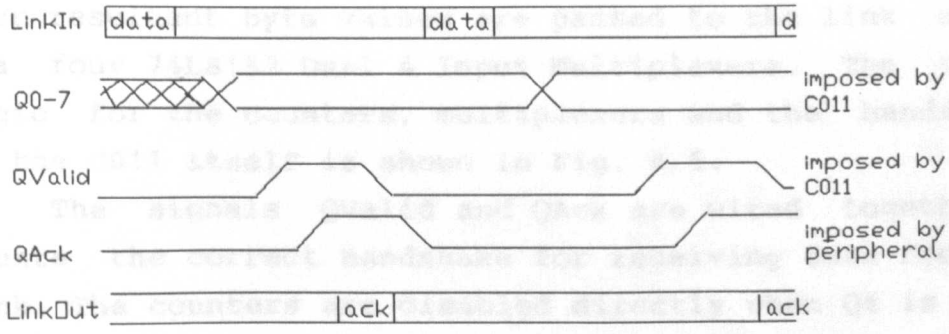
4.3)THE SPEED INPUT BOARD

The motor speed signal is derived from an Incremental Optical Encoder mounted directly on the shaft of the induction motor. The general specifications for this device are given in Appendix A. The encoder provides a



Timing Diagram for Data Input to the Link Adapter

Figure 4.2.



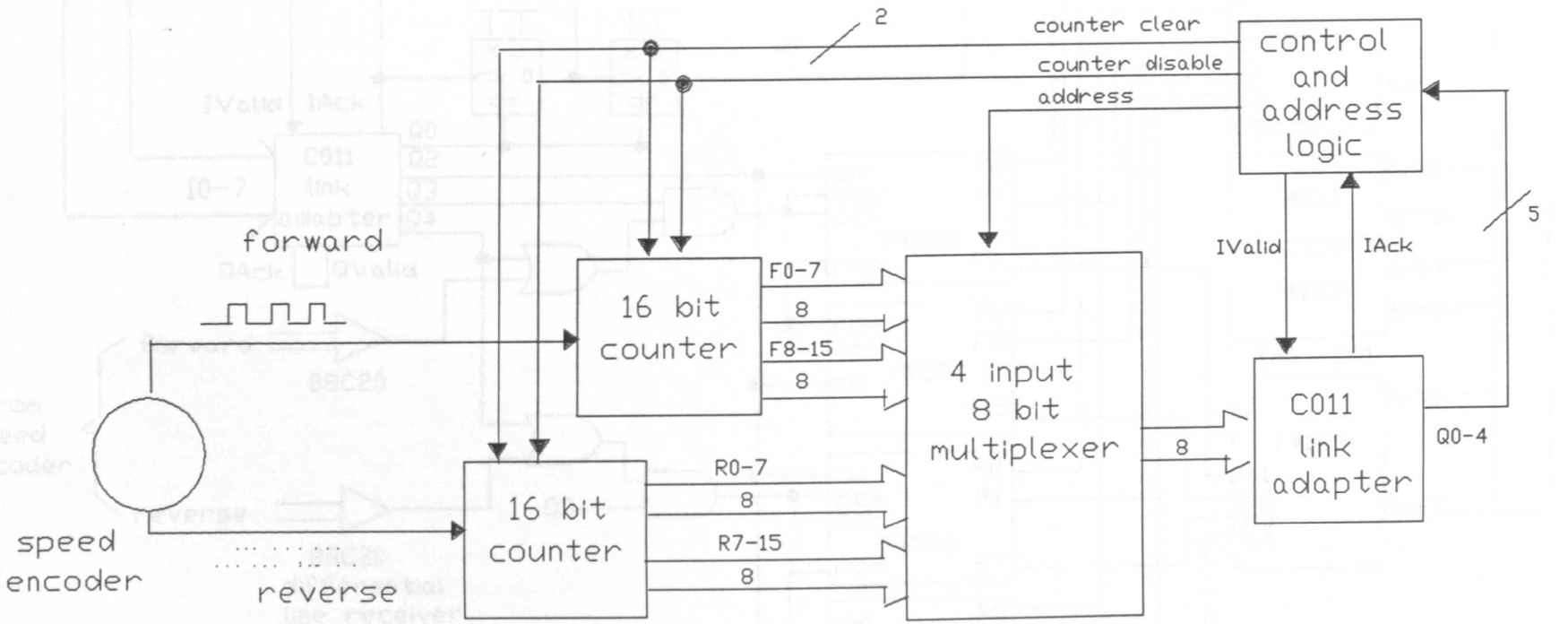
Timing Diagram for Data Output From the Link Adapter

Figure 4.3.

pulse train on one of two lines dependent on the direction of rotation of the shaft. The form of these signals are 5 Volt CMOS using complementary drivers, and 10,000 pulses are produced per revolution of the shaft. The method chosen for speed measurement was to count the number of pulses produced on each line in a set sample period (2 ms), dictated by the control transputer. The speed input board thus requires two 16 bit counters (one for each direction) and a suitable multiplexer to channel the four bytes into the byte wide input port of the link adapter. The schematic diagram for the speed input board is given in Fig 4.4.

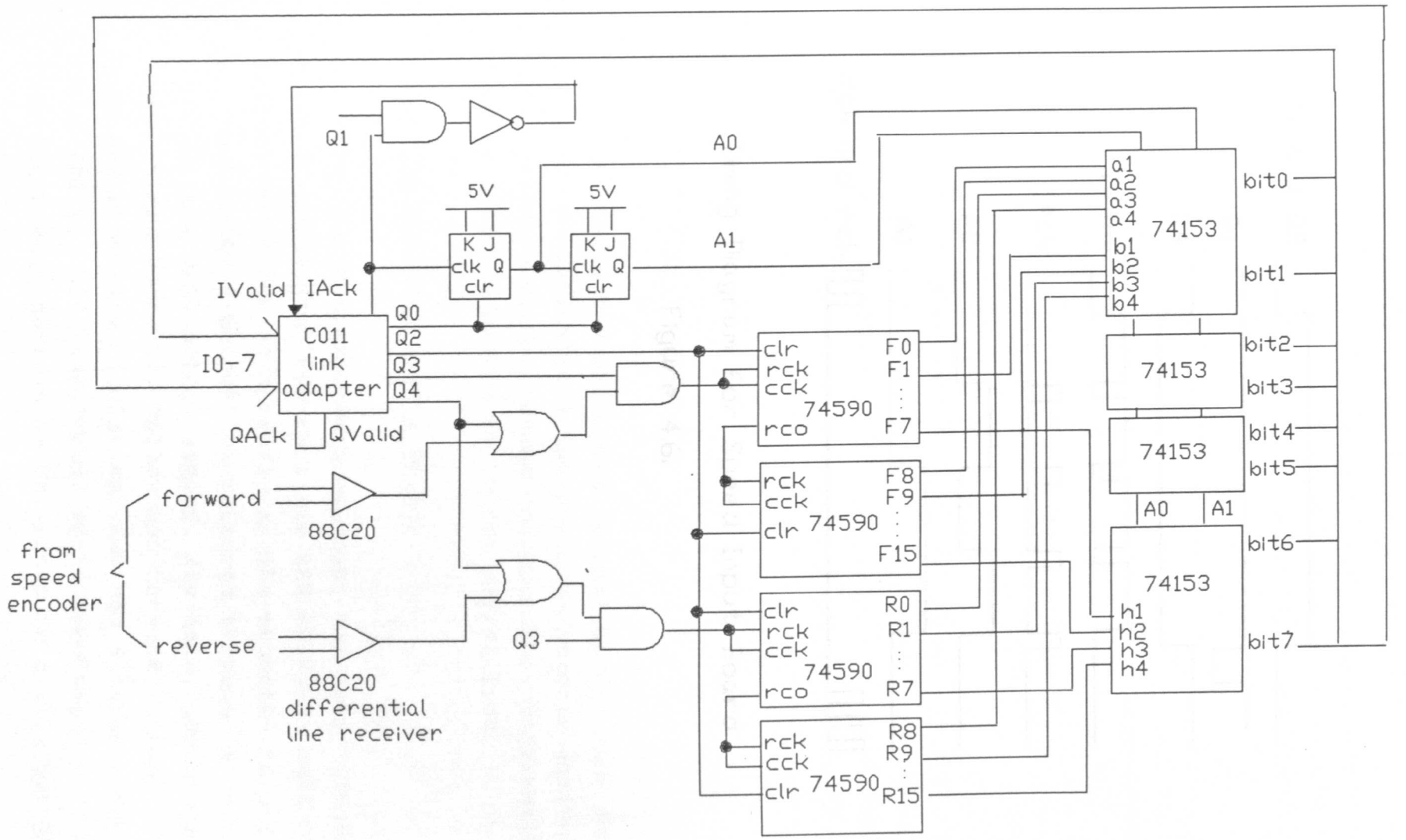
The pulse train signals are input to the board via DS88C20 CMOS Compatible Differential Line Receivers. Logic is provided for counter clear pulses to be superimposed at start up and the pulse train is then fed directly to the counter. The 16 bit counter is made up of two 74LS591 8-Bit Binary Counters, specifically chosen as they have an 8-bit output register, and can be easily cascaded. The four resultant byte values are passed to the link adapter via four 74LS153 Dual 4 Input Multiplexers. The control logic for the counters, multiplexers and the handshaking of the C011 itself is shown in Fig. 4.5.

The signals QValid and QAck are wired together to create the correct handshake for receiving data from the link. The counters are disabled directly when Q4 is taken low by the controlling transputer at the start of the counter read cycle, and the inverse of this signal, Q0 is used to enable IValid for the output of data to the link. The address lines (select 0 and select 1) of the multiplexers are changed by JK Flip-Flops triggered by the rising edge of IAck. Also IValid is taken low when the IAck signal goes high, resulting in the Link Adapter being "disabled" during the changeover of the multiplexers. When the four bytes have been read by the transputer it outputs two further control bytes to clear the counters, restart the counters and reset the JK Flip-Flops. The timing



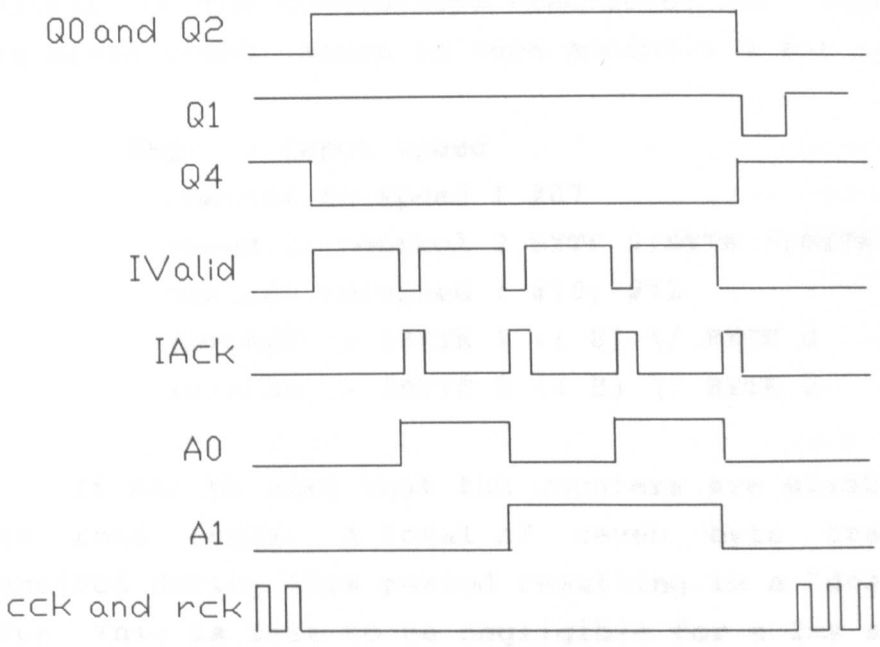
Block Diagram for the Speed Input Board

Figure 4.4.



Circuit Diagram for Speed Interface Board

Figure 4.5.



Timing Diagram for Speed Input Board

Figure 4.6.

diagrams for the control signals are shown in Fig. 4.6.

The occam procedure executed every speed sample instant on the controlling transputer for inputting the two speed count values is (see Appendix D for syntax):-

```
SEQ  -- input speed
  control.to.speed ! #07
  speed.to.control ? BYTE 0;BYTE 1;BYTE 2;BYTE 3
  control.to.speed ! #10; #12
  forward := (BYTE 1 << 8) \/ BYTE 0
  reverse := (BYTE 3 << 8) \/ BYTE 2
```

It can be seen that the counters are disabled during the read cycle. A total of seven byte transfers are required during this period resulting in a "dead time" of 35µs. This is felt to be negligible for a 2ms sample time (1.7%).

4.4)THE CURRENT INPUT BOARD

The Current Input Board is made up of two separate circuits. One conditions the analogue signals derived from the transducers. The second converts the information to digital form and passes it to the serial link.

4.4.1)The Analogue Signal Board

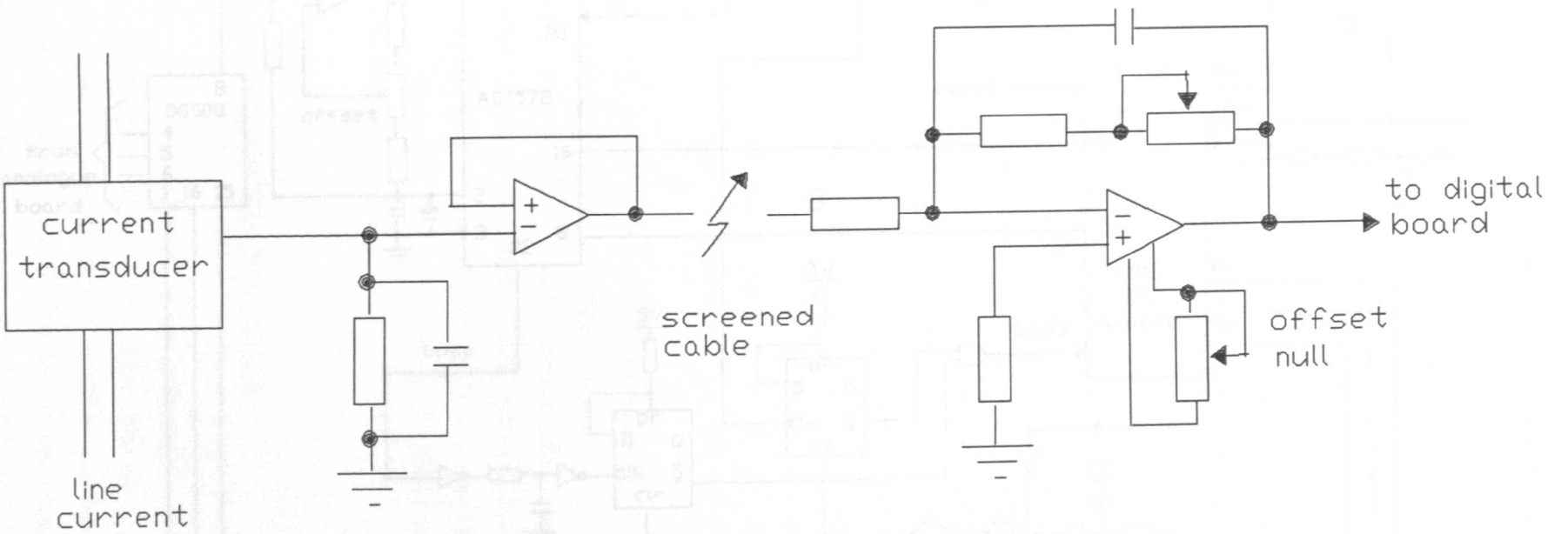
The three transducers employed for this project to measure motor line currents are LEM LT-80-P Hall Effect devices described in Appendix A. The magnetic flux created by the primary current is balanced through a secondary coil with a Hall effect sensor. Its main advantages are that it can correctly follow any current signal, offers complete electrical isolation, and has a large measurement range (80 A). The transformer ratio used was 1 : 1000, and the secondary current is converted to a voltage in the

range +/- 2.5 V using a 47kΩ resistor. The signal is then buffered using a simple voltage follower. These devices are mounted near the inverter. The buffered signals are transmitted to filter board mounted within the control box using screened cable to reduce noise pickup. Each signal then passes through a standard low pass filter circuit (cut-off frequency 2 kHz). Gain and offset adjustment for each filter are provided to correct for any discrepancy between the transducers and sensing resistors. The schematic for this circuit is illustrated in Fig. 4.7.

4.4.2)The Analogue to Digital Conversion Board

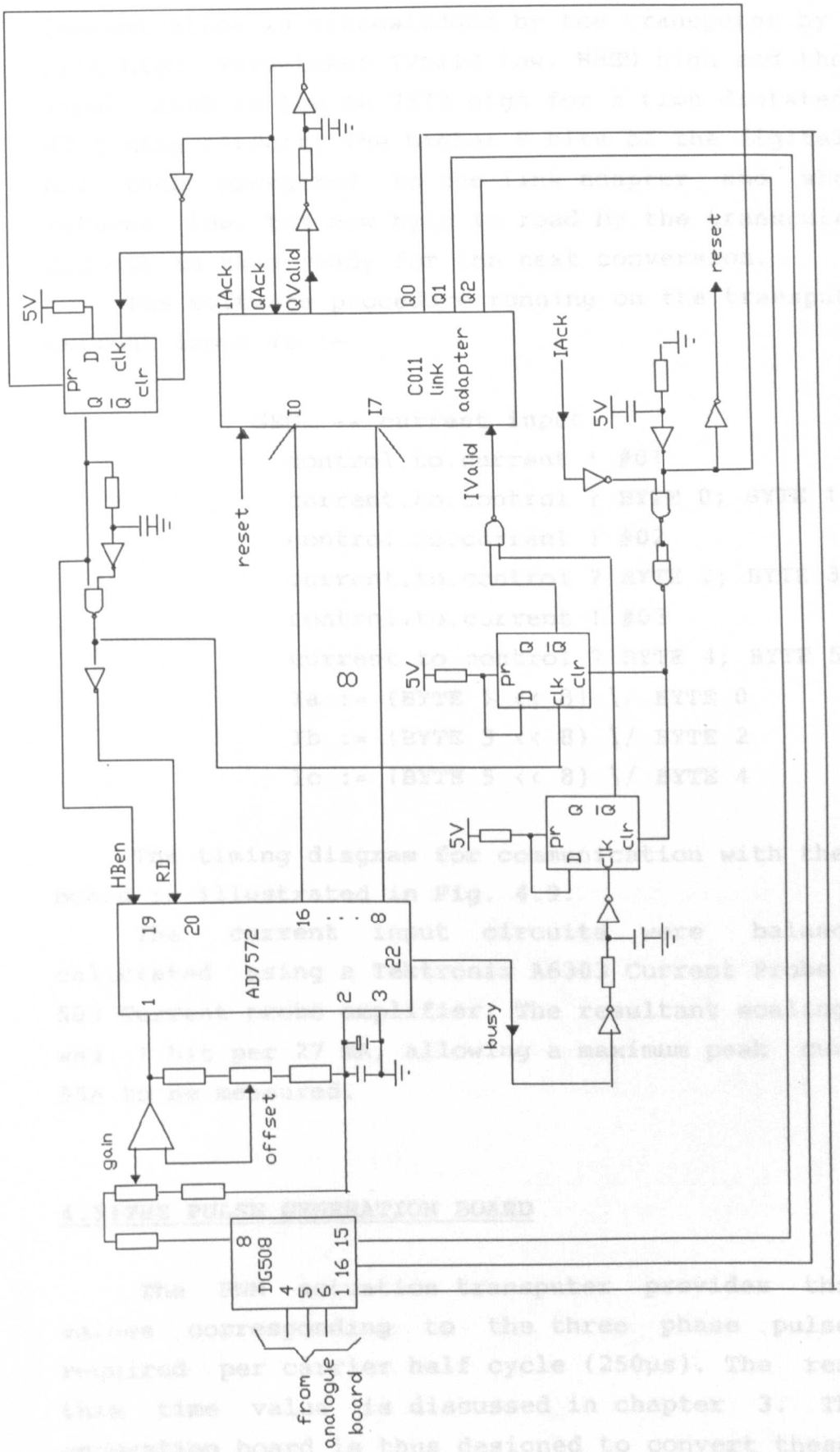
This circuit was developed as a multi-channel analogue to digital card for interfacing to a transputer, and provides access to up to 8 independently addressable analogue signals. The circuit diagram for this system is illustrated in Fig. 4.8.

The 3 analogue current signals are fed into a DG508 Analogue Multiplexer, whose address lines are fed directly from Q0-Q2 of the Link Adapter. The analogue-to-digital converter is an AD 7572 12 bit device with a conversion time of 5μs. The input requirement to this device is a 0-5V signal and the corresponding output is in the range #0000 - #0FFF. Therefore the buffer circuit is included to give bipolar offset and gain adjust for the +/-2.5V input signal. Conversion is initiated by the controlling transputer by outputting the address byte for the multiplexer to the Link Adapter. The signal QValid takes the control input to the AD 7572 (HBEN) low and starts the conversion process on the multiplexed analogue signal. The BUSY output signal of the device is taken low for the period of conversion (t_{con}). On completion of the conversion the lower 8 bits of the digital value obtained are presented to I0-I7 of the link adapter, and the signal BUSY goes high, which takes IValid high and initiates communication to the transputer of this byte.



Circuit Diagram for Analogue Signal Board for Current Measurement

Figure 4.7.



Circuit Diagram for Digital Signal for Current Measurement

Figure 4.8.

Communication is acknowledged by the transputer by taking IAck high. This takes IValid low, HBEN high and the signal input READ to the AD 7572 high for a time dictated by an RC timing circuit. The higher 8 bits of the digital value are then presented to the link adapter and when READ returns low, the new byte is read by the transputer. The circuit is then ready for the next conversion.

The software procedure running on the transputer for current input is :-

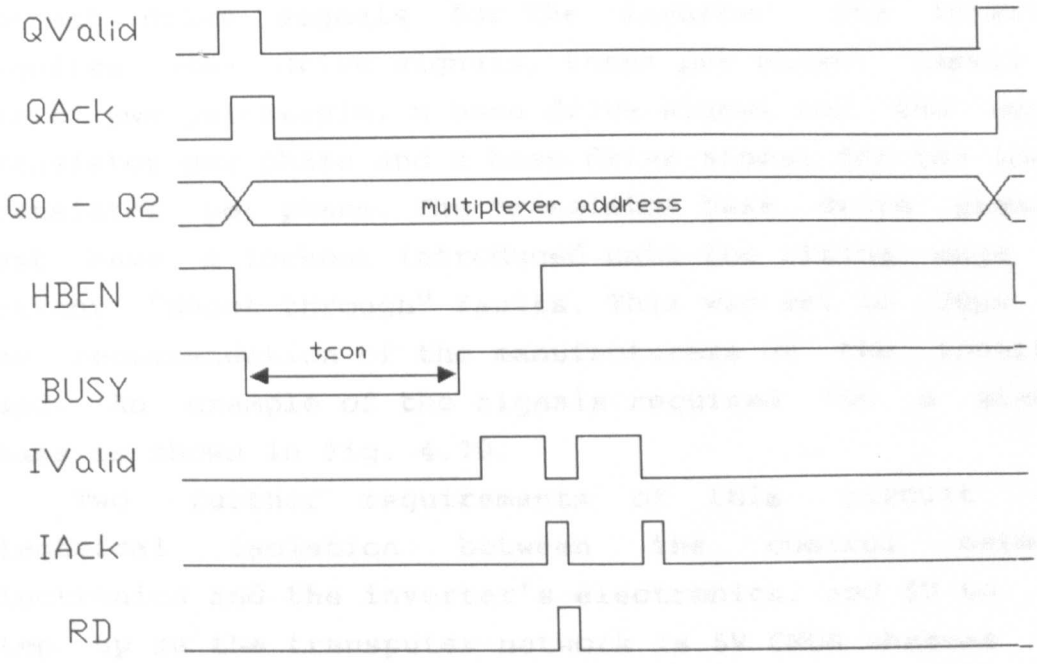
```
SEQ  -- current input
  control.to.current ! #01
  current.to.control ? BYTE 0; BYTE 1
  control.to.current ! #02
  current.to.control ? BYTE 2; BYTE 3
  control.to.current ! #03
  current.to.control ? BYTE 4; BYTE 5
  Ia := (BYTE 1 << 8) \/ BYTE 0
  Ib := (BYTE 3 << 8) \/ BYTE 2
  Ic := (BYTE 5 << 8) \/ BYTE 4
```

The timing diagram for communication with the current board is illustrated in Fig. 4.9.

The current input circuits were balanced and calibrated using a Tektronix A6303 Current Probe and AM 503 Current probe amplifier. The resultant scaling factor was 1 bit per 27 mA, allowing a maximum peak current of 55A to be measured.

4.5)THE PULSE GENERATION BOARD

The PWM actuation transputer provides three byte values corresponding to the three phase pulse widths required per carrier half cycle (250μs). The reason for this time value is discussed in chapter 3. The pulse generation board is thus designed to convert these values



Signal Timing Diagram for the
Current Input Board

Figure 4.9.

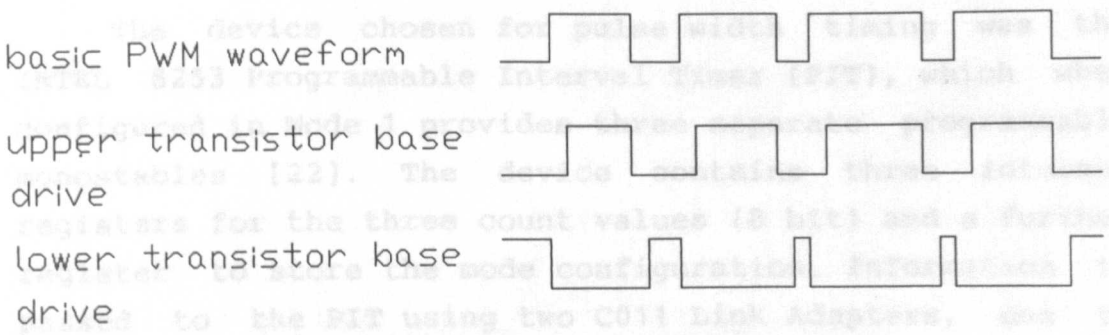


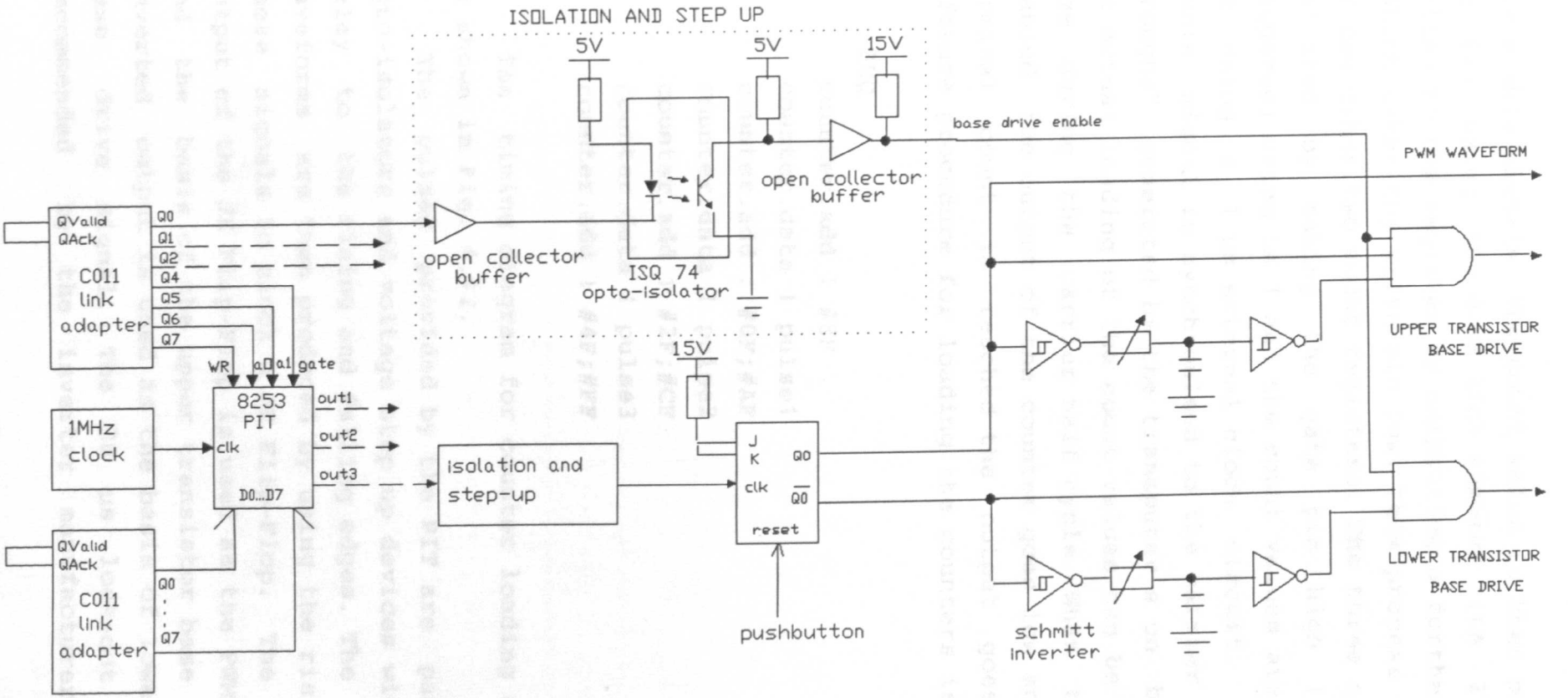
Figure 4.10. Inverter Drive Signal Requirements

to into three pulse trains and then convert these into the correct drive signals for the inverter. The inverter requires nine drive signals, three per phase: namely a basic pwm pulsetrain, a base drive signal for the upper transistor per phase and a base drive signal for the lower transistor per phase. The transistor base drive signals must have a lockout introduced onto the rising edge to prevent "shoot through" faults. This was set to 20μs on the recommendation of the manufacturers of the inverter used. An example of the signals required for a single phase is shown in Fig. 4.10.

Two further requirements of this circuit are electrical isolation between the control network electronics and the inverter's electronics, and 5V to 15V step up as the transputer network is 5V CMOS whereas the input signals to the inverter need to be 15V CMOS compatible. The protection circuitry of the inverter itself comes between the pulse generation board output and the transistor base drive units themselves. This means that the inverter's protection circuits (overcurrent, overload, overvoltage, undervoltage, overtemperature, and external trips) could be utilised.

The circuit diagram for the pulse generation board is shown in Fig. 4.11.

The device chosen for pulse width timing was the INTEL 8253 Programmable Interval Timer (PIT), which when configured in Mode 1 provides three separate programmable monostables [22]. The device contains three internal registers for the three count values (8 bit) and a further register to store the mode configuration. Information is passed to the PIT using two C011 Link Adapters, one to provide the control and address signals, and the other to provide the configuration data and count values. When a count register is to be loaded with a new value, the transputer outputs a control byte to the address link adapter (LA1) which correctly addresses the required register using A0 and A1, and disables the read, write and



Circuit Diagram for the Pulse Generation Board

Figure 4.11.

gate enable signals. The count value is then presented to the PIT using the data link adapter (LA 2). This is written to the register by outputting a further byte to LA 1 which takes the write pin low. This process is repeated for the other two count registers. The three counters are initiated by taking the gate pin high (rising edge triggered) using LA 1 and the count values are then timed out using a 1 µs external clock circuit. The counter enable signal is synchronised to the carrier "peaks" and "troughs" generated by the transputer's on board timer, but actual loading of the count values can be done at any time during the carrier half cycle. When the gate is enabled the output of the counter goes low and when the terminal count is reached the output goes high. The software procedure for loading the counters is :-

```
SEQ
  counter.add ! #8F
  counter.data ! pulse1
  counter.add ! #0F;#AF
  counter.data ! pulse2
  counter.add ! #2F;#CF
  counter.data ! pulse3
  counter.add ! #4F;#FF
```

The timing diagram for counter loading and enabling is shown in Fig. 4.12.

The pulses provided by the PIT are passed through opto-isolators and voltage step up devices with negligible delay to the rising and falling edges. The correct PWM waveforms are then produced by using the rising edge of these signals to clock a JK Flip-Flop. The non-inverted output of the JK Flip-Flop is used as the PWM drive signal and the basis of the upper transistor base signal. The inverted output is used as the basis of lower transistor base drive signal. The 20 µs lock-out delay (as recommended by the inverter manufacturers) is then

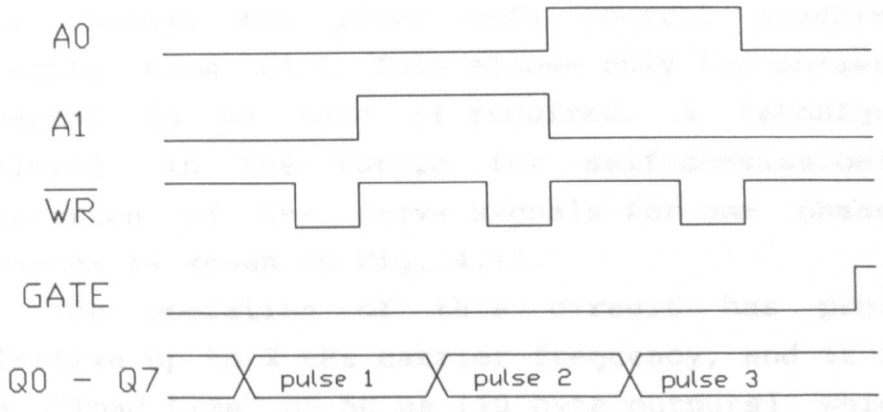


Figure 4.12. Signal Timing Diagram for the Pulse Generation Board

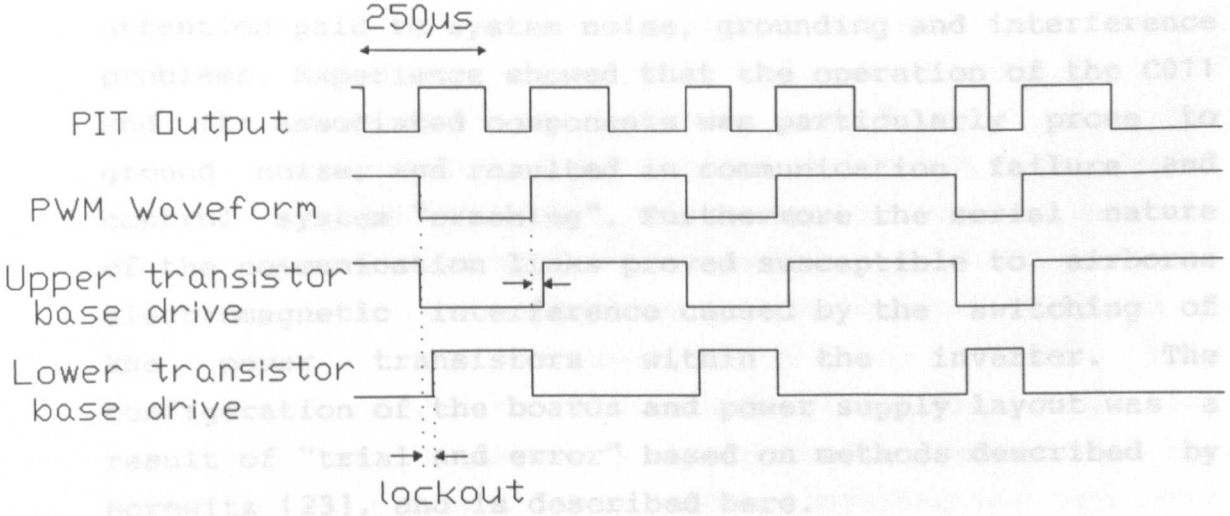


Figure 4.13. Pulse Generation Board Output Waveforms

introduced onto the rising edge of the transistor drive signals using a simple RC variable delay network. Before being passed to the inverter the transistor drive signals pass through AND gates with control signals derived directly from LA 1. This allows only two phases of the inverter to be used if required, a technique to be employed in the future for self-commissioning. The generation of the drive signals for one phase of the inverter is shown in Fig. 4.13.

The operation of this circuit has proved very effective up to 2 kHz carrier frequency, and it is mainly the "load time" of 50 µs (10 byte outputs) which proves restrictive.

4.6) INTERFACE BOARD DESIGN AND CONTROLLER LAYOUT

It is worth mentioning the considerations made when designing the interface board layout and the overall attention paid to system noise, grounding and interference problems. Experience showed that the operation of the C011 and its associated components was particularly prone to ground noise, and resulted in communication failure and control system "crashing". Furthermore the serial nature of the communication links proved susceptible to airborne electromagnetic interference caused by the switching of the power transistors within the inverter. The configuration of the boards and power supply layout was a result of "trial and error" based on methods described by Horowitz [23], and is described here.

The interface boards were laid out on double sided PCBs, with tracks routed in order to minimise route length and capacitive and inductive coupling between signals. 74HC series components were used where TTL compatibility was required, but 40 series CMOS used where possible to increase ground noise immunity. Liberal use of ground planes and decoupling capacitors were also used in order

to increase ground noise immunity and common mode rejection.

The B004 board was mounted within the IBM PC itself and derived its 5V power supply from it. The power supply for the IBM PC was isolated from the mains earth. The interface boards were mounted within an earthed aluminium box to provide a Faraday shield against airborne carried noise. A single 5V power supply was used for the B006 board, current input board digital circuitry, speed input board and speed encoder, and the non isolated pulse generation board circuitry, and this was mounted within the box. The ground point of this supply acted as the star point for the grounds of all the boards used, and it was at this point that the transputer network was earthed. Further power supplies for the pulse generation board (5V and 15V isolated) and the analogue current input board (+/- 15V) were mounted within the box and all used mains filters. Thick wires were employed to connect power supply grounds to the relevant boards. This configuration reduced the susceptibility of the control circuitry to ground noise carried on the mains earth and ground loops.

The main sources of airborne interference were the signal leads connecting the current and speed transducers to the interface boards, and the cable connecting the pulse generation board to the inverter itself. The current transducers were mounted close to the inverter and used a separate +/- 15V power supply (with mains filter). This supply also powered the buffer amplifiers. The cable used for the current signals was single core screened, with the screen used to common the grounds at sending and receiving ends. The speed encoder signals were derived from differential line drivers within the encoder itself, which provided sufficient airborne noise immunity. The pulse train signals for the inverter were screened with the screen grounded at the control circuit end.

The resultant control circuit worked effectively. However, it was noticed that when high acceleration currents were used by the controller, the whole microprocessor system would "crash" on an intermittant basis. The nature of this problem was found to be "Transputer Deadlock" - a problem associated specifically with parallel processing.

The term Deadlock describes a situation whereby a process is waiting for a communication from another parallel process before it can procede, and that communication does not arrive. This scenario is normally due to incorrect implementation of the parallel algorithms, but in this case the cause is associated with the physical operation of the transputer's serial links themselves. The high acceleration currents cause increased airborne and ground noise, which can interfere with the data transmission from the link adapters to the control transputer. If the acknowledge data on the serial link is corrupterd during a communication, then the recieving process will not complete the communication. It cannot therefore proceed to its next instruction and this causes the process to stop. If the control processor stops, it cannot then provide the communications required to the overseer and actuation processors, and these processes will also stop. The deadlock is thus propogated throughout the whole of the control network, and the system must then be shut down and rebooted.

It is thus essential that attention be paid to the problems of noise reduction in a transputer based control system in order to ensure the correct operation of the communication links.

CHAPTER 5

TRANSPUTER IMPLEMENTATION OF VECTOR CONTROL

5.1) INTRODUCTION

This chapter describes the occam implementation of the vector control algorithms described in chapter 2. The actuation procedures are described initially so that the output requirements from the control processors can be precisely defined. The supervisory routines running on the host and the B004 motherboard are described next as they are common to all of the vector control algorithms employed. Sections three, four and five describe the control processes employed and how they interface with the actuation transputer and the transducer boards. Finally the implementation of the T_r identification strategies on a fourth transputer are described, noting their "background" nature.

5.2) ACTUATION SIGNAL GENERATION

The control strategies employed demanded that two separate actuation routines be devised for providing the required drive signals to the inverter. The first method, commonly termed Pulse Width Modulation (PWM) imposes motor voltages, whereas the second method, here termed "Bang-Bang" control, imposes motor currents. It was decided that the routines should calculate the required pulsewidths for each phase of the inverter and then output the values to a custom built interface board which would then convert them to real time pulses with suitable "lock out" protection. The operation of the board is described in chapter 4, and requires only byte wide output on two separate serial

links from the actuation transputer.

5.2.1)PULSE WIDTH MODULATION

5.2.1.1)The PWM Algorithm

The principle of sine weighted pulse width modulation utilises the comparison of a triangular "carrier wave" with a sinusoidal "modulation wave" in order to generate the pulses. The most popular techniques employ a fixed amplitude carrier wave, and a modulation wave whose amplitude, phase and frequency are controlled. The frequency of the carrierwave can be held constant (asynchronous PWM) or varied such that the carrierwave is always in synchronism with the modulation wave (synchronous PWM). Digital generation of PWM waveforms is typically based on one of these analogue techniques [24],[25],[26],[14],[27] and although the main advantage of synchronous PWM over asynchronous PWM is that unwanted harmonics are reduced at higher speeds, asynchronous techniques are easier to implement. For this reason a fixed frequency carrier wave was employed in this project.

The calculation strategy employed in this project is directly analogous to the traditional analogue method for pulse width modulation, and is known as "asynchronous symmetrical regular sampling". A fixed frequency, fixed amplitude triangular carrier wave is compared with a sinusoidal modulation wave, whose frequency, amplitude and phase are control inputs. The ratio of the modulation wave amplitude to the carrierwave amplitude is called the "modulation index" (m). Calculations based on the instantaneous values of the carrier and modulation waves at each sample point provide the pwm waveform. The basic principle behind this strategy is shown in Fig. 5.1. and compared to the analogue derived output.

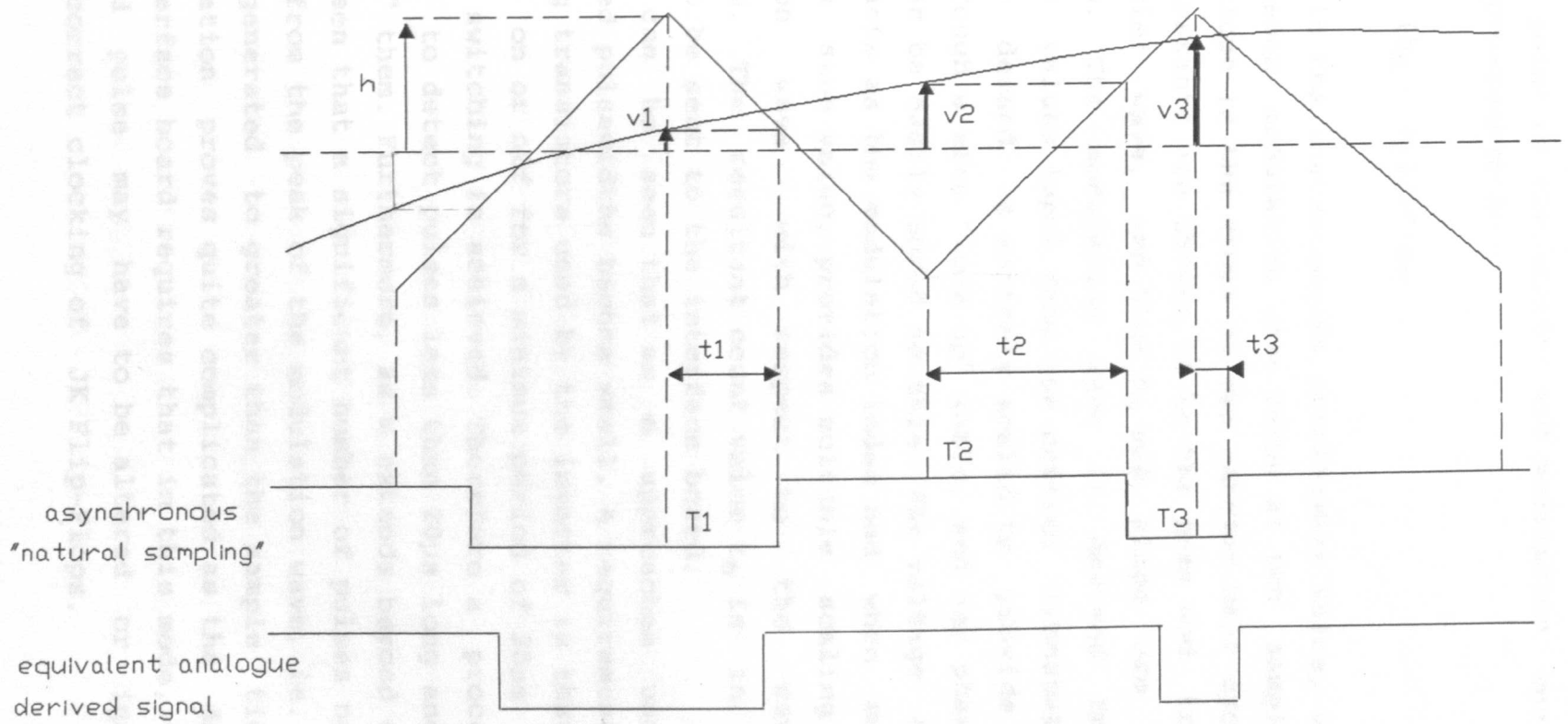


Figure 5.1. Pulse Width Calculation Strategy.

At each sample instant T_n the time until the next crossing point of the carrier and modulation waves (t_n) can be approximated by :-

$$t_n = (h_n - v_n)/\text{slope} \qquad (5.1)$$

where h_n is the instantaneous carrierwave value, v_n is the instantaneous modulation wave value at the sample point T_n, and slope is the slope of the carrier half cycle. The sample instants are chosen to be the peak and trough of the carrier wave, and thus h_n and slope are software constants. The modulation wave is derived from the reference values input from the control transputer. The frequency demand is suitably scaled to provide a step angle through a sine "look-up" table, and any phase change demand can be easily added to this. The voltage amplitude demand acts as the modulation index and when multiplied with the sine value, provides suitable scaling of the modulation wave with respect to the carrierwave amplitude. The resultant count value t_n is in μs, and ready to be sent to the interface board.

It can be seen that as m approaches unity, the calculated pulsewidths become small. A requirement of the switching transistors used by the inverter is that they be switched on or off for a minimum period of 20μs so that correct switching is achieved. Therefore a procedure is required to detect pulses less than 20μs long and ignore or "drop" them. Furthermore, as m extends beyond unity it can be seen that a significant number of pulses need to be dropped from the peak of the modulation wave, ie. set the pulses generated to greater than the sample time. This consideration proves quite complicated as the design of the interface board requires that in this mode, the next generated pulse may have to be altered or ignored to provide correct clocking of JK Flip-Flops.

5.2.1.2)OCCAM Implementation of the PWM Algorithm

The occam pseudo-code (see Appendix D) for the PWM routine (PWM.TST) is shown here. The numbers shown in brackets next to a particular process represent the time taken to execute that process in microseconds.

```
SEQ
  ... setup
  ... startup
  PRI PAR

    SEQ       -- PWM generation
      WHILE TRUE
        SEQ
          ... initiate low to high half cycle     (2)
          ... read control variables             (30)
          ... enable counters                     (5)
          ... access sine look-up table          (10)
          ... calculate modulation voltages      (12)
          ... calculate pulsewidths              (20)
          ... load pulsewidths to counters       (45)
          ... time out half cycle                 (1)
          ... initiate high to low half cycle     (2)
          ... pass flux angle data to control    (20)
          ... enable counters                     (5)
          ... access sine look-up table          (10)
          ... calculate modulation voltages      (12)
          ... calculate pulsewidths              (20)
          ... load pulsewidths to counters       (45)
          ... time out half cycle                 (1)

    SEQ       -- dummy
      ... do nothing
```

The setup process defines the variables and constants employed in the program, and also defines the procedure "buffer" which is used at startup to read in the values for the look-up table and assign them to memory addresses. The start-up procedure executes buffer, assigns the initial values to the variables, and programs the interface board to run in the correct mode by outputting the required control and mode information to the interface links. The initial count values are then passed to the interface board.

The PWM routine itself is actually a single sequential process, but is defined in software as a high priority process running in parallel with a low priority process called "dummy". The reason for this is that the process requires access to the transputer's high priority timer, and this is only achieved by defining it as a high priority process. Dummy does not actually do anything. The PWM routine is governed by a WHILE TRUE construct which means that the process will be repeated indefinitely.

The low to high carrier half cycle is initiated by reading the value of the transputer clock to the variable "now". The control variables are then read in from the control transputer from the link "control.to.pwm", and the design of the algorithms running on the other transputers dictates that this information will be available as soon as the pwm transputer is ready to receive it, ie. the control transputer will be waiting for this communication. The counter start signal is passed to the interface board precisely 35µs after the start of the cycle by employing the "TIME ? AFTER (clock value)", which provides a wait until the transputer clock value specified is reached. This initiates the timing out of the count values loaded in the previous half cycle. The look-up table pointers are then incremented using the step angle derived from the frequency input. The look-up table itself consists of 10,000 integer values (0 - 100) representing 0 - 180 degrees of a sinewave. The same look-up table is accessed

for 180 - 360 degrees and a toggle is set to -1. Two pointers are used at 90 degrees phase shift to access sine and cosine information for a particular flux angle. The three demodulation voltages are calculated directly from the d and q axis voltage demands and flux angle information using the transformations given in Appendix C.

The parameter "h" is calculated as the difference between the carrier and modulation values. If this value is negative then the pulsedropping is required and the output must remain high for the whole of the carrier half cycle (positive pulsedropping). The routine checks the previous pulsewidth to see if a dummy pulse must be generated to correctly clock the interface board. Similarly if the value of h is greater than 20,000, the output must remain low for the whole of the half cycle (negative pulsedropping). If pulsedropping is not required, the pulsewidth is calculated and checked to ensure that it is not too long or too short (minimum pulsewidth requirement of the transistors). Again the previous pulse must be checked and this one altered if necessary to maintain correct clocking of the interface board.

The calculated pulsewidths are loaded onto the interface using byte output instructions to the serial links. The transputer then waits until the clock value has reached (now + 247) and then initiates the high to low carrier half cycle. This results in a process time for the sample of 250μs. This was chosen as the lockout time and minimum pulse time for the inverter are 20μs and thus will form only a small percentage of the PWM sample time. The second cycle is again initiated by reading the transputer clock value, and then outputs the flux angle sine and cosine values to the control transputer. The rest of the routine is similar to the first half cycle, except that the carrier value and slope are inverted, and no control variables are input. At the end of this half cycle the routine repeats itself.

The resultant system provides a 2 kHz PWM waveform, with a 250μs delay between input of the control variables and enabling of the count values.

5.2.2)THE "BANG-BANG" CURRENT CONTROLLER

5.2.2.1)The Bang-Bang Algorithm

The actuation transputer employed in this mode provides a fast "localised" current control for the inverter itself, by comparing the reference and measured line currents at a sample instant and switching the upper or lower transistor of the line according to the result, for the whole of the sample period. For this mode the the current measurement needs to be performed by the actuation transputer itself. The inverter drive signals are produced using the same pulse generation board as for PWM generation. The operation of the current detection board is described in chapter 4, and for the purposes of program development can be considered as presenting the instantaneous current measurement as a 16 bit word on the serial link "current.to.control", immediately after an address byte has been output to the link "control.to.current". This algorithm is implemented with a sample time of 250μs, and again can be considered as a carrier with two distinct half cycles, although in this case the carrier has no influence on the calculation strategy.

5.2.2.2)OCCAM Implementation of the Bang-Bang Algorithm

The occam pseudo-code for the bang-bang controller (CCON.TST) is given here.

```
SEQ
  ... setup
  ... startup
  PRI PAR

    SEQ          -- pulse generation
      WHILE TRUE
        SEQ
          ... initiate first half cycle                (2)
          ... read control variables                  (30)
          ... access sine look-up table               (10)
          ... calc modulated reference currents       (12)
          ... lead lag compensation                   (23)
          ... measure line currents                   (43)
          ... compare meas and ref currents
              and calculate pulses                     (7)
          ... demodulate measured currents            (27)
          ... load and enable counters                (50)
          ... time out half cycle                      (1)
          ... initiate second half cycle               (2)
          ... pass measured currents to control       (20)
          ... access sine look-up table               (10)
          ... calc modulated reference currents       (12)
          ... lead lag compensation                   (23)
          ... measure line currents                   (43)
          ... compare meas and ref currents
              and calculate pulses                     (7)
          ... demodulate measured currents            (27)
          ... load and enable counters                (50)
          ... time out half cycle                      (1)

    SEQ          -- dummy
      ... do nothing
```

The structure of this program is similar to that of the PWM program. The setup and startup procedures define the variables, constants and initial values, load the sine look-up table, and program the pulse generation board. The calculation strategy is implemented as a high priority parallel process in order to access the high priority timer and is executed in an infinite loop. The low priority process "dummy" does nothing.

The first carrier half cycle is initiated by reading the current value of the transputer clock to the variable "now", and then the reference values for frequency, i_d and i_q are read immediately from the control transputer. The flux angle pointers are incremented and the flux angle sine and cosine values accessed from the look-up table. The three instantaneous reference values for the line currents are calculated from i_d, i_q, $\sin\Theta_e$ and $\cos\Theta_e$ using the transformation given in Appendix C. If required, Lead-Lag compensation is introduce onto these reference values using the binomial transformation outlined in Appendix E.

The three instantaneous values of line current are measured directly from the current input board by sending a control byte to address the relevant transducer, and then reading the measured value. The measured and (compensated) reference values for the instantaneous line currents are then compared for each phase and the result used to determine whether the upper or lower transistor of each inverter leg should be switched on for the next half cycle. These switching states are then compared with the previous switching states for each leg, to determine whether a "short" (10µs) pulsewidth is required to clock the interface JK Flip-Flops, or a "long" (255µs) pulse is required so that the Flip-Flops are not clocked during the half cycle. Modulation of the measured line currents for monitoring purposes is then carried out using the transformations given in Appendix C. The interface board counters are then loaded using the same routine as for the

PWM strategy. A TIME ? AFTER (now + 210) wait statement is then employed to ensure that the pulse initiation is synchronised to the same point of every carrier half cycle, and the counters enable by outputting the correct control byte to the interface board. The half cycle is completed by a further wait statement to time out the 250μs sample period. This value is chosen as the minimum time required for the actuation calculations. The second half carrier cycle is identical to the first, except that the measured values of i_d and i_q are output to the control transputer, and no reference values are input. The modulation of the measured currents is redundant in this half cycle. The resultant system gives a 2 kHz sample time.

5.3)SUPERVISORY ROUTINES

The user interface to the vector control network is provided by supervisory routines running on the B004 motherboard and the IBM PC itself. The specification for these routines were that they were to provide :-

1)keyboard and screen drivers for direct motor speed control

2)an initialisation routine to allow the user to change the control parameters at startup

3)an ability to change the value of rotor time constant used by the controller whilst the motor is running

4)access to instantaneous variables used by the controller

5)an ability to capture data over a prescribed period and transfer this data to a disk file

6)access to graphics and plot routines to allow visual data analysis whilst the motor is running

7)an ability to call up background T_r identification routines when required for assessment

The basis of the software design was to put the onus of the supervisory work onto the B004 motherboard and use the host only as a "buffer" routine. Provision for writing to disk files must be provided within the host routine, and it must also be capable of accessing graphics and plot routines without disturbing the operation of the transputer network.

5.3.1)The OCCAM Implementation of the IBM PC Supervisor

This program is written in occam as a sequential procedure, (SETUP.EXE) and compiled and linked to the host's processor using the TDS development system. The occam pseudo-code for this process is given here.

```
SEQ
  ... setup
  ... startup
  WHILE TRUE
    SEQ
      IF
        choice = 'y'
          ... read and display current measurements
        choice = 'c'
          ... change speed
        choice = 'r'
          ... take 600 consecutive measurements
        choice = 'f'
          ... file measurements
        choice = 't'
          ... change rotor time constant
        choice = 'i'
          SEQ
            ... select type of id strategy
            ... execute id strategy
            ... file results
```

```
        choice = 'q'
            ... quit program
        ... read new choice
```

The setup procedure consists of defining variables, constants and procedures used by the program. The procedures defined are "write.word.to.link" and "read.word.from.link", which provide 32bit word communication from the host to the B004 motherboard via the IBM PC input/output port - transputer serial link interface, "display" which converts an integer value to a string and sends it to the screen, "readparameters" which reads a string from the keyboard and converts it to an integer value, and "openfile" and "closefile", which provide access to hard disk files.

The start up routine invokes the procedure load.table, which opens the disk file "result18" and passes its contents to the transputer network to be loaded into memory on the actuation transputer as the sine look-up table. It assigns initial values to the variables employed, sends the initial values of the control parameters to the screen (namely L_s,R_s,σL_s,T_r,i_{sdref} and the torque current limit value), and asks the user to modify these values if required. When these values have been corrected the host then passes them to the B004 for distribution to the correct routines. A set of instructions for using the host interface is then displayed.

The on line supervisor consists of a sequential program which monitors input commands from the keyboard and acts accordingly on them. The command message for inputting a new speed is displayed on the screen and the user input converted to integer form and passed to the B004 via the serial link. The variable "choice" is set to 'y' and the program then passes to an infinite loop which executes a process on the basis of the command variable choice using a simple "IF" statement. It then reads in

the new command variable from the keyboard. If the command variable is :-

'y' the process passes this value to the B004 transputer and then reads in the single set of five control and measured variables used by the control transputer at that particular instant, and displays them on the screen.

'c' the supervisor then asks the user to input a new speed within a prescribed range. It then passes the control value and new speed demand to the B004. The speed transient is initiated and 600 consecutive data samples are stored on the B004. The last set of captured data is then passed to the supervisor and this is displayed on the screen.

'r' this control value is passed to the motherboard and 600 consecutive steady state data samples are stored on the B004.

'f' this value is sent to the motherboard and the 600 stored samples are read into five memory arrays on the host processor. The supervisor then asks the user to input a file name for the data and this file is opened using the predefined procedure. The array information is then written to this hard disk file and the file closed on completion.

't' the supervisor asks the user to input a new value for the control value T_r (rotor time constant) within a prescribed range. The supervisor then passes the command value and the new control variable to the B004 for incorporation into the control strategy.

'i' the supervisor asks the user which type of identification strategy is to be employed, ie. reactive power (v) or PRBS injection (p). It passes these values to the motherboard and initiates a rotor time constant identification routine. This routine will run for a period of 600 samples, updating the controller rotor time constant value or i_{sd} reference value depending on which mode is chosen, and storing

five calculation variables on the motherboard every sample. The 600 sample sets are then passed to the supervisor via the serial link and stored in the host's memory. The supervisor then asks the user if this data should be written to a hard disk file, and if so to input the filename. The file is then opened, written and closed, and the process finished by displaying two final results from the T_r identification routine on the screen. Note that this is available for the V type with current feedback controller only.

'q' which terminates the execution of this program and returns the host to its operating system.

Once a command variable has been executed the supervisor asks the user to input a new command variable and the procedure repeats itself.

When the supervisory program is terminated the transputer network continues to run but with no user interface. The plot and graphics routines can then be called up for visual analysis of the transient and steady state data. User interface can be restored by calling up the supervisor routine and initial setup and startup processes are by-passed.

5.3.2)The OCCAM Implementation of the B004 Supervisor

This routine (OVS.TST) makes use of the ability of a single transputer to execute parallel processes on a time slicing basis. Two procedures are defined, one high and one low priority process. The high priority process (buffer) provides synchronised communication with the control transputer whereas the low priority process (overseer) interfaces between the buffer and the host and T_r identification processes as a "background" routine, preventing the long interface and calculation processes from extending the sample time of the control system. The

occam pseudo code for this program is given here.

```
SEQ    -- B004 supervisory routine
  ... setup
  ... startup
  PRI PAR

    SEQ   --  buffer
      WHILE TRUE
        SEQ
          ... pass control values to controller
          ... read measurements from controller
          ... read clock
          ALT
            overseer.to.buffer ? fe
              SEQ
                ... read control vars from overseer
                ... write measurements to overseer
            TIME.?.AFTER (now + 1800)
              SKIP

    SEQ   -- overseer
      WHILE TRUE
        SEQ
          ... input command variable from host
          IF
            choice = 'c'
              ... input speed reference from host
              ... pass new speed reference to buffer
              ... store 600 samples from buffer
              ... pass last sample set to host
            choice = 'r'
              ... store 600 samples from buffer
            choice = 'y'
              ... read one set of samples from buffer
                  and pass to host
```

```
choice = 'f'
  ... pass 600 data samples to host
choice = 't'
  ... input new Tr from host
  ... pass new Tr to buffer
choice = 'i'
  SEQ
    ... input ident mode from host
    IF
      mode = 'v'
        ... reactive power identification
      mode = 'p'
        ... PRBS identification
    ... pass identification data to host
```

The setup process defines variables, constants, internal channels and the procedures used by the program. The three procedures used are "read16" and "write16", which provide correct serial link interface between the 32bit T414 motherboard transputer and the 16bit host and T212 control transputers, and "load.pwm" which simply reads the values for the sine look-up table from the host and passes them to the control transputer. The start up procedure executes "load.pwm", reads the control parameters from the host and passes them to the control transputer, and sets up the initial values for the variables used.

The program is configured to run in parallel mode using the PRI PAR statement which governs the two processes "buffer" and "overseer". "Buffer" has its own local variables defined within the process and is executed in an infinite loop. The control values for speed, flux current and rotor time constant are passed directly to the control transputer via the serial link. These values will be read by the control routine when it is ready, and it will pass the set of five sampled values of control and measured parameters to the buffer immediately after. The

current value of the transputer clock is then read to the variable "now". The interface between the user and the control network is created by the ALT construct, using the input from the internal channel "overseer.to.buffer" and input from the transputer clock as its operands. If the overseer routine is ready to communicate with the network it will output the speed demand to the buffer on "overseer.to.buffer". The buffer will then read in the flux current and rotor time constant demands from overseer, and pass the last set of five sampled control and measured parameters read from the control transputer to the overseer routine using, the channel "buffer.to.overseer". The overseer has a period of 1800μs in which to initiate this communication from the time that the transputer clock was read. If this communication does not occur the TIME ? AFTER (now + 1800) input to the ALT construct dictates that this communication is ignored for this cycle, and the buffer procedure returns to its initial mode in readiness for communication with the control transputer. The principle behind the time-slicing technique employed by the transputer ensures that the buffer process will always be executed on demand and as such the overseer can be treated as a "background" routine.

The overseer routine acts as a software buffer between the host or T_r identification routine and the control transputer. The routine is governed by a WHILE TRUE statement and is thus executed as an infinite loop. The routine starts by reading the command value "choice" from the supervisor, and selects a process depending on the value of this parameter. These values are :-

'c' the new speed demand is read from the supervisor. This is then passed to the buffer along with the current flux current and rotor time constant demands, and the previous set of sampled data are read into this process from buffer and put into a two dimensional array. The array pointer is

incremented and this process repeated 600 times to capture data over the transient period.

'r' this command initiates a capture of data over 600 consecutive samples and so a steady state "window" is stored.

'y' then only a single sample set is captured and this is passed to the host for screen display.

'f' the process invoked passes the whole of the two dimensional array previously captured to host for writing to a hard disk file. In this case there is no interaction between buffer and overseer.

't' the new value of rotor time constant is read from the supervisor. This is then passed to the buffer along with the current speed and flux current demands, and the single set of variables passed from the buffer is ignored.

'i' the supervisor becomes an interface between the control transputer and the transputer executing the background calculation routines. Its mode of action depends on the identification routine selected. For the reactive power process the supervisor writes the demand variables for speed, flux current and rotor time constant to, and reads the control and measured data from buffer as before. It then passes the measured data to the T_r identification transputer which performs the identification routine. It then reads in the modified demand values of T_r from the identification transputer and this is passed to the buffer on the next iteration.

This process is executed over a period of 600 samples, and relevant parameters are stored in the two dimensional array as before. The PRBS process updates i_d every sample according to the value passed from the identification transputer, and passes the measured value of speed to it. After 600 iterations the identification transputer performs its correlation routine and passes the results to the

supervisor for storage. The modified value of T_r is then passed to the buffer. When the identification is complete the array is passed to the host for storage in a hard disk file, and the final value of two relevant parameters are passed to the host for screen display. Note that this is available for the V type with current feedback controller only.

On completion of any of these processes the routine returns to the start to wait for the new command value from the host.

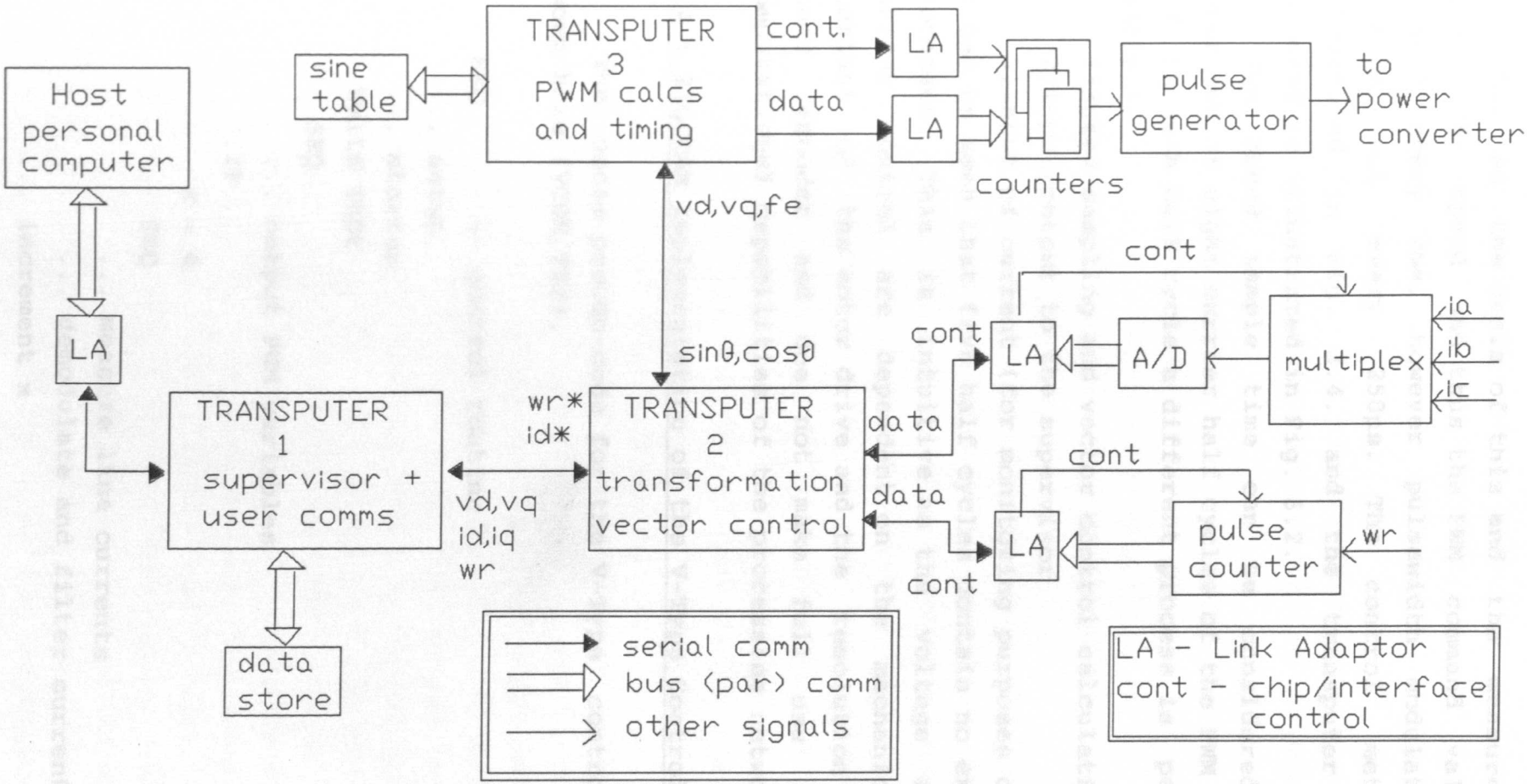
5.4)THE CONTROL TRANSPUTER ROUTINES

The three control algorithms employed, namely V Type, V-Type with Current Feedback and I-Type control, are similar in structure and implemented as a single sequential program on the control transputer. They are divided into two distinct "half cycles" with each half cycle being initiated by communication with the actuation transputer.

5.4.1)V-Type Control

5.4.1.1)The V-Type Control Algorithm

For V-Type Control the control transputer uses one bi-directional serial link for communication with the supervisor, one for communication with the speed input board, one for communication with the current input board, and one for communication with the actuation transputer, programmed in this application to run as a PWM generator. The control algorithm demands that when a new speed demand is input from the supervisor routines, the controller calculates suitable reference signals for the PWM



Implementation of V Type Vector Control Strategies

Figure 5.2.

generator on the basis of this and the measured motor speed. The speed and thus the PWM command values are updated every 2ms, however pulsewidth modulation is carried out every 250µs. The control method is illustrated in Fig. 2.4. and the transputer network employed is illustrated in Fig. 5.2.

The speed sample time can be considered to be comprised of eight carrier half cycles of the PWM routine, and in each half cycle a different process is performed, namely :-

1)speed sampling and vector control calculations

2)input/output to the supervisor

3)input of current (for monitoring purposes only)

It can be seen that five half cycles contain no executable processes. This is intuitive as the voltage and thus current control are dependent on the mechanical time constant of the motor drive and the resolution of the speed encoder and doe not make full use of the computational capabilities of the processing network.

5.4.1.2)OCCAM Implementation of the V-Type Controller

The occam pseudo-code for the V-Type controller is shown here (VCON.TST).

```
SEQ         -- control routine
  ... setup
  ... startup
  WHILE TRUE
    SEQ
      ... output PWM variables                    (30)
      IF
        x = 4
          SEQ
            ... measure line currents             (43)
            ... demodulate and filter currents    (44)
      ... increment x                             (1)
```

```
            ... input flux angle data                (20)
            IF
              x = 1
                SEQ
                  ... read commands from supervisor   (30)
                  ... pass measured data to supervisor(50)
              x = 7
                SEQ
                  ... measure speed                   (30)
                  ... PI speed control                (2)
                  ... calculate PWM variables         (29)
            ... increment x                           (1)
```

The setup process is used as before to define the variables and constants used by this program and defines the procedure "load.pwm" which is used to pass the values for the sine look-up table to the PWM generator. The startup procedure executes "load.pwm", and then reads the control parameters from the supervisor. It then assigns initial values to the variables used in the program. The control routine then runs as a sequential procedure governed by a WHILE TRUE construct.

The first carrier half-cycle is initiated by communicating the control values v_d, v_q and f_e to the PWM generator. The overall programming strategy ensures that the sender is ready before the receiver and synchronises the controller to the PWM timing. The routine then uses a flag ("x") to determine the process to be executed in this half-cycle. At startup, "x" is set to 0. If the flag is '4' the three phase currents are read directly from the current input board, and converted to instantaneous values of i_d, i_q using the transformations given in Appendix C. The measured currents are then passed through a digital low pass filter, the algorithm for which is outlined in Appendix E. If the flag has another value no process is implemented. At the end of the half-cycle the flag is incremented. The resultant current measurements are

sampled almost directly in the middle of the PWM output pulses, resulting in significant reduction of the carrier frequency harmonic content. The timing of the current input is such that they are read in the middle of a speed sample period as well.

The second carrier half cycle is initiated as soon as the PWM generator is ready to transfer the flux angle sine and cosine values required for current demodulation. The value of the flag is again used to determine which process is implemented. If "x = '1'" the controller reads the speed, flux current and rotor time constant demands from the supervisor, and passes the current values of speed, demanded v_d and v_q, and measured i_d and i_q to it. If the flag is '7', the controller measures the speed from the speed input board and performs the vector control calculations. The flux current demand i_d is a control value passed from the host. The torque current demand i_q is derived from a digitally implemented PI controller acting on the speed error value, the algorithm for which is described in Appendix E. A limit is placed on this value to prevent excessive motor currents. The slip frequency demand is derived directly from equation 2.10 and suitably scaled to be in electrical Hz. This is added to the measured speed signal (also scaled in electrical Hz) to provide the frequency demand for the PWM generator. The d and q axis voltage demands for the PWM generator are calculated using equations 2.17. and 2.18., again with limits placed on these values. If the flag is equal to '3' or '5' no process is performed. The carrier half cycle ends with the flag being incremented or reset if "x = '7'".

The V-Type controller thus provides a 500Hz speed sample frequency and monitor sample frequency, and although the control parameters for the PWM generator are updated at this frequency, pulsewidths are updated at a rate of 2kHz.

5.4.2)V-Type Control with Current Feedback

5.4.2.1)The Control Algorithm

The transputer network and program structure are identical to those used by the V-Type controller. The major changes are that the d and q axis currents are sampled every carrier cycle and the reference values for the PWM generator are calculated every carrier cycle using PI controllers acting on the current error signals. The speed is still sampled at a rate of 500 Hz. The control method is illustrated if Fig 2.6.

5.4.2.2)OCCAM Implementation of V-Type Control with Current Feedback

The occam pseudo code for the control program is illustrated here (ICON.TST).

```
SEQ     -- control routine
  ... setup
  ... startup
  WHILE TRUE
    SEQ
      ... output PWM variables                    (30)
      ... measure line currents                   (43)
      ... demodulate and filter currents          (44)
      ... increment x                              (1)
      ... input flux angle data                   (20)
      IF
        x = 1
          SEQ
            ... read commands from supervisor     (30)
            ... pass measured data to supervisor(50)
```

```
x = 7
  SEQ
    ... measure speed                    (30)
    ... calculate i_sqref                (2)
IF
  softstart = TRUE
    ... use preset PWM variables
  softstart = FALSE
    SEQ
      ... PI control of currents
      and d - q axis compensation        (26)
... increment x                          (1)
```

Setup and startup are identical to that of V-Type control, however a further flag ("softstart") is set to TRUE. The transputer network is initiated before the inverter itself is enabled (by an external push-button). The purpose of the softstart flag is to prevent the closed loop current controllers from working (and saturating) when the controller is activated and before the inverter is enabled, and prevents closed loop current control until the user "says its OK" by inputting a non zero speed demand.

The first control half-cycle is initiated by outputting the command signals to the PWM generator. For each iteration of this half cycle the currents are read, demodulated and filtered as before, and a 2kHz sample rate ensues. The flag "x" is incremented at the end of the half-cycle.

The second half-cycle is initiated by reading the flux angle sine and cosine values from the PWM transputer, and then the value of "x" is used to determine the next process. If "x = '1'" the controller inputs the demand values for speed, flux current and rotor time constant from the supervisor and outputs the measured values for speed, i_d and i_q, and command values for v_d, and v_q to it. If "x = '7'" the speed is measured and the torque current

reference generated as before using a PI controller. This process also generates the PWM frequency demand. The next process is then executed for each iteration of this half-cycle. D axis current control is achieved using a PI controller acting on the error between the flux current reference and the d axis current measured in the previous half-cycle. The controller was designed using equation 2.19. and implemented using the bilinear transform outlined in Appendix E. Similarly q-axis current control is implemented based on equation 2.20. The outputs of the controllers are the d and q axis voltage demands for the PWM generator. The compensation outlined in section 2.5.2 can be included by using the demand values for i_d, i_q and frequency to modify the d and q axis voltage demands as outlined in equations 5.1 and 5.2.

$$'v_{dref} = v_{dref} - i_{dref}w_e\sigma L_s \qquad (5.2)$$

$$'v_{q}ref = v_{qref} + i_{dref}w_e L_s \qquad (5.3)$$

Note that the current controllers are disabled until the user inputs an initial non zero speed demand. When this occurs the softstart flag is set to FALSE and the current controllers enabled. The voltage demands used prior to this are "safe" preset values that prevent current surge when the inverter is enabled. The final instruction of this half cycle increments or resets the flag "x" accordingly.

5.4.3)I-Type Control

5.4.3.1)The I-Type Control Algorithm

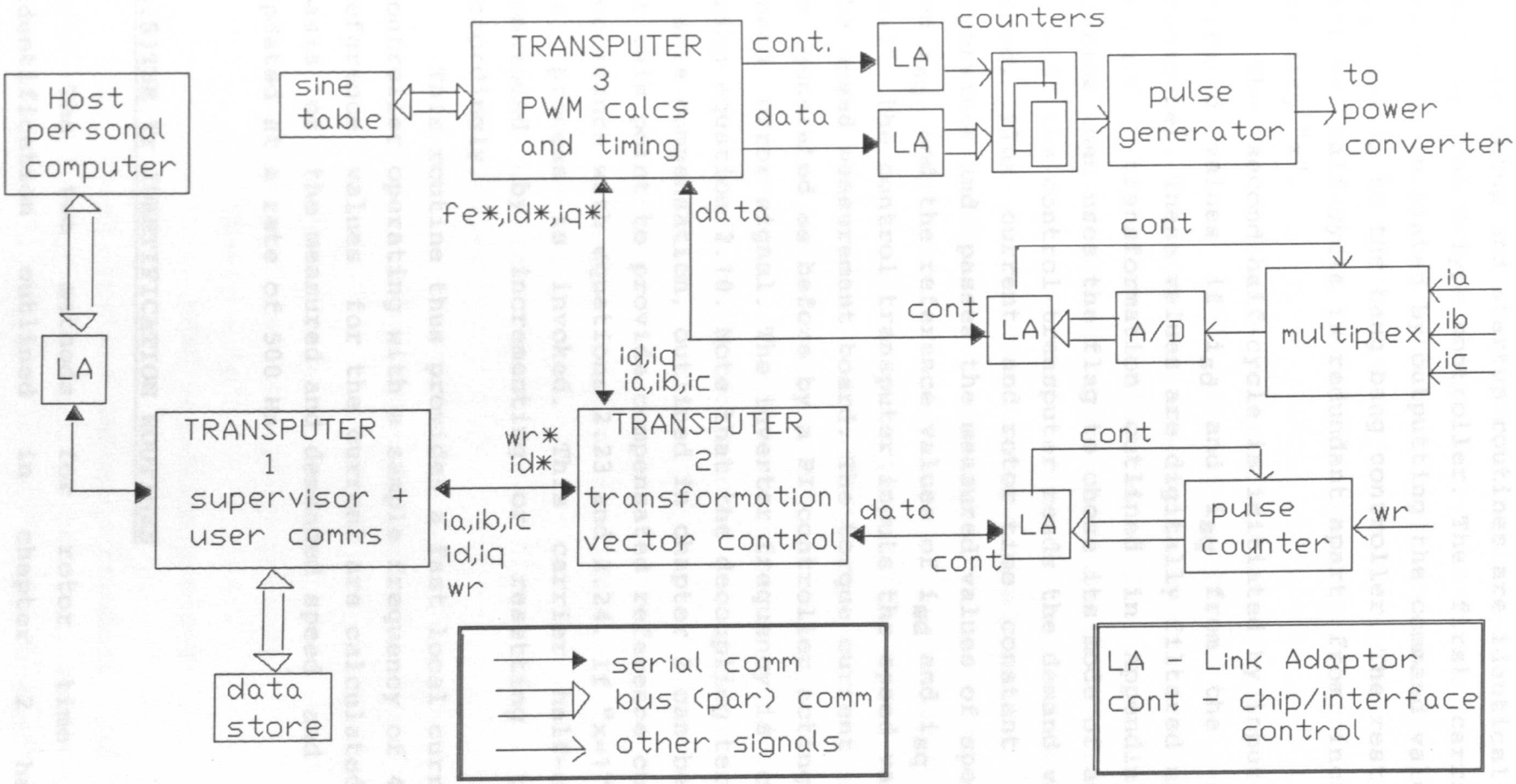
The hardware layout for this algorithm differs slightly from that used for the voltage controlled modes in that the current detection board is now connected to the actuation transputer, which performs the bang-bang

local current controller. The control algorithm is similar in structure to that of V-Type control, and in this case only two of the eight carrier half cycle which make up a 2ms speed sample time, actually do any operations. The control method is illustrated Fig. 2.8. and the transputer network implemented for this control strategy is shown in Fig. 5.3.

5.4.3.2)OCCAM Implementation ofI-Type Control

The occam pseudo-code for this program is shown here (SCON.TST).

```
SEQ         -- control routine
  ... setup
  ... startup
  WHILE TRUE
    SEQ
      ... output ref currents and frequency      (30)
      ... increment x                             (1)
      ... input measured currents                (20)
      ... filter currents                        (17)
      IF
        x = 3
          SEQ
            ... read commands from supervisor    (30)
            ... pass measured data to supervisor(50)
        x = 7
          SEQ
            ... measure speed                    (30)
            ... calculate i_sqref, f_e
                and d - q axis compensation      (27)
      ... increment x                             (1)
```



Implementation of I Type Vector Control Strategy

Figure 5.3.

The setup and startup routines are identical to those used by the V-Type controller. The first carrier half cycle is initiated by outputting the command values i_{sd}, i_{sq} and f_e to the bang bang controller. The rest of the carrier half-cycle is redundant apart from incrementing the flag "x".

The second half-cycle is initiated by inputting the measured values if i_{sd} and i_{sq} from the actuation transputer. These values are digitally filtered using the bilinear transformation outlined in Appendix E. The routine then uses the flag to chose its mode of action. If "x = 3" the control transputer reads the demand values for speed, flux current and rotor time constant from the supervisor and passes the measured values of speed, i_{sd}, and i_{sq}, and the reference values of i_{sd} and i_{sq} to it. If "x=7" the control transputer inputs the speed value from the speed measurement board. The torque current reference is generated as before by a PI controller acting on the speed error signal. The inverter frequency is calculated using equation 2.10. Note that the decoupling terms for d-q axis compensation, outlined in chapter 2 can be included at this point to provide compensated reference currents in accordance with equations 2.23 and 2.24. If "x=1" or "x=5" no process is invoked. This carrier half-cycle is concluded by incrementing or resetting the flag accordingly.

This routine thus provides a fast local current loop controller operating with a sample frequency of 4 kHz. The reference values for the current are calculated on the basis of the measured and demanded speed and are thus updated at a rate of 500 Hz.

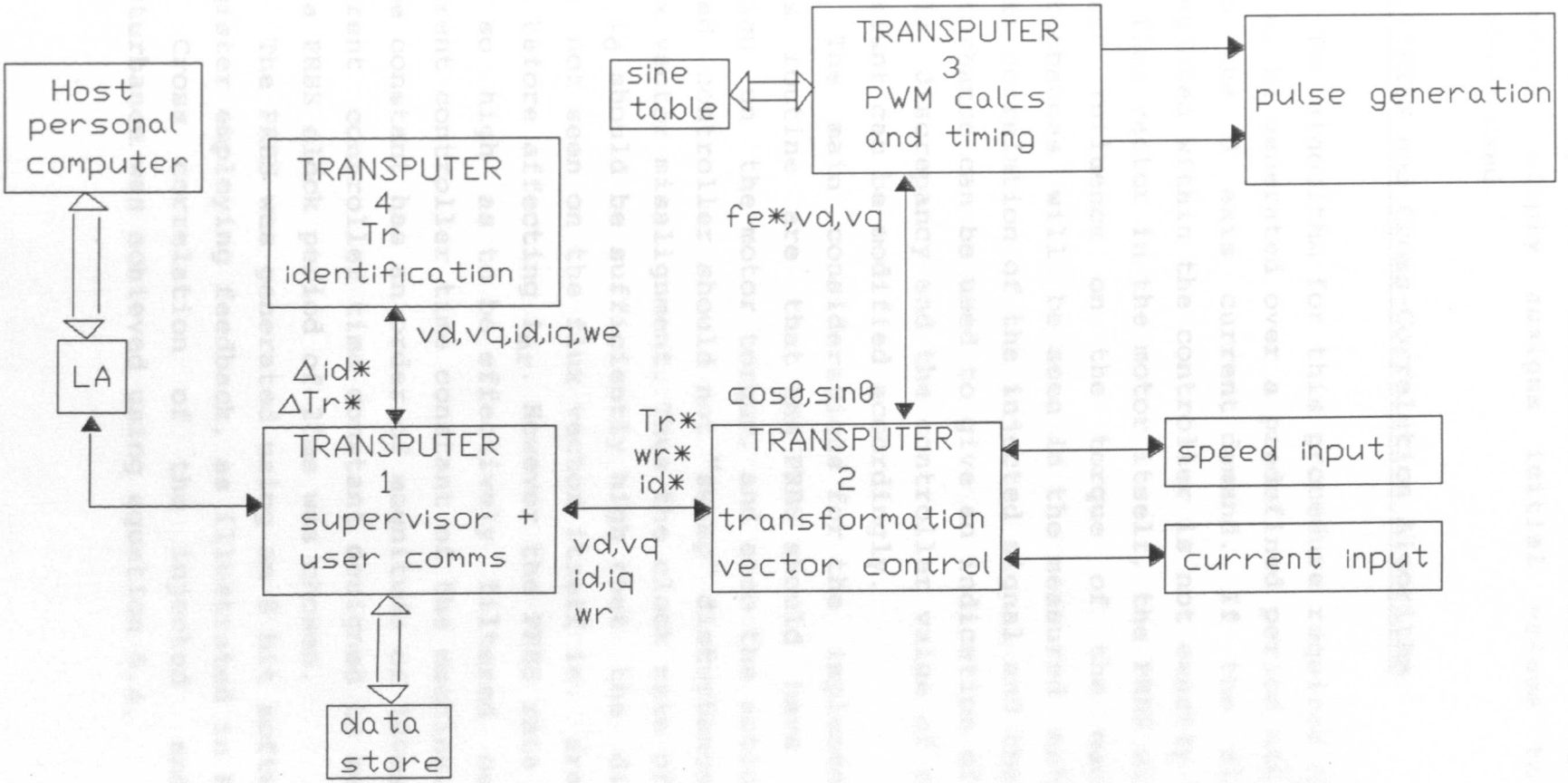
5.5)THE Tr IDENTIFICATION ROUTINES

The two methods for rotor time constant identification outlined in chapter 2 have been

incorporated into the vector control algorithm using a further T212 transputer connected to the B004 supervisor via a bi-directional serial link. The extension to the transputer network is illustrated in Fig 5.4. It must be noted here that the identification algorithms were only employed on the vector control using V-Type control with current feedback, as this was the only method which gave both reasonable estimates of the d and q axis motor voltages, and provided accurate current control. The routines were only implemented on the cage motor. The strategies are computationally intensive, demanding the use of real number routines. However it was found that the 2ms sample time dictated by the speed control loop was sufficient to allow both strategies to be performed on the sampled data. Both strategies are implemented using a single routine running on the identification transputer. The identification is triggered by the user and a command variable is used to decide which strategy to use. The pseudo code for the overall routine is given here, and the pseudo code for the two processes are given later in the chapter.

```
SEQ    -- identification routine
  ... setup
  ... startup
  WHILE TRUE
    SEQ
      ... input trigger and choice
      IF
        choice = 'v'
          ... reactive power process
        choice = 'p'
          ... PRBS injection process
```

The setup routine defines variables and constants used by the program, as well as the real number procedures "IntegerToReal", "RealOp", and "RealToInteger" which are



Implementation of Rotor Time Constant Identification

Figure 5.4.

required by the calculation strategy. The startup procedure simply assigns initial values to certain variables used.

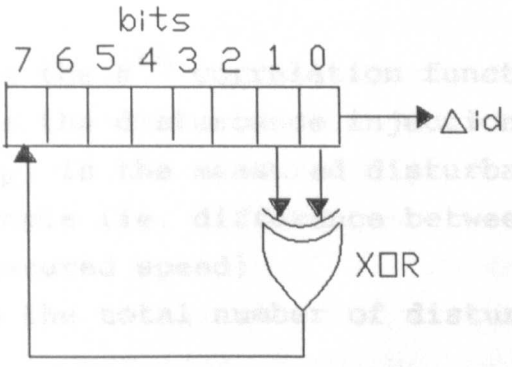
5.5.1)PRBS and Cross-Correlation Algorithm

The algorithm for this procedure requires that a PRBS signal be generated over a predefined period and injected onto the d axis current demand. If the flux vector calculated within the controller is not exactly aligned to the flux vector in the motor itself, the PRBS will have a direct influence on the torque of the machine and disturbances will be seen in the measured motor speed. Cross correlation of the injected signal and the measured disturbance can be used to give an indication of the flux angle discrepancy and the controller value of rotor time constant can be modified accordingly.

The main considerations for the implementation of this routine are that the PRBS should have no direct action on the motor torque, and also the action of the speed controller should not "swamp" disturbances due to flux vector misalignment. Thus the clock rate of the PRBS on i_d should be sufficiently high that the disturbances are not seen on the flux vector itself ie. are filtered out before affecting i_{mr}. However the PRBS rate should not be so high as to be effectively filtered out by the current controller time constant of the machine. The rotor time constant has an order of magnitude of 100ms, and the current controller time constant designed to be 5ms, and so a PRBS clock period of 20ms was chosen.

The PRBS was generated using an 8 bit software shift register employing feedback, as illustrated in Fig 5.5.

Cross correlation of the injected and measured disturbance was achieved using equation 5.4.



Shift Register Employing Feedback to Function as a PRBS Generator

Figure 5.5.

$$c_k = (1/r) \sum_{n=1}^{r} d_n . m_{(n+k)} \qquad (5.4)$$

where: c_k is the k^{th} correlation function
d_n is the disturbance injection at the n^{th} sample
$m_{(n+k)}$ is the measured disturbance at the $(n+k)^{th}$ sample (ie. difference between speed demand and measured speed)
r is the total number of disturbances injected

The amplitude and phase of the peak correlation function value can be used to determine whether T_r should be increased or reduced. This procedure must be repeated several times in order to "hone in" on the correct value of T_r.

5.5.2)PRBS OCCAM Implementation

This algorithm was implemented as a sequential process outlined by the occam pseudo-code given here. The data sampling and disturbance injection are implemented as a process which iterates for 600 cycles. The routine is completed by then completing the correlation calculations, determining the maximum correlation coefficient, and using this value to update the rotor time constant.

```
SEQ     -- PRBS injection process
  SEQ i = [0 FOR 600]
    SEQ
      ... read data from supervisor
      ... pass data to supervisor
      ... calculate disturbance
  ... calculate correlation functions
  ... sort correlation functions
  ... pass correlation functions to supervisor
  ... pass modified Tr to supervisor
```

The first process invoked is the communication of measured data from the supervisor and passing of command values for flux current and rotor time constant to the supervisor. The next process provides the PRBS signal. This is "turned off" for the first 100 samples, clocked for samples 100 -500, and then turned off again for the last 100 samples. The clock rate is 10 samples, resulting in an injection of 40 disturbances, each lasting 20ms. These disturbances provide a +/- 10% deviation to the flux current reference signal. The routine then invokes the summation calculations required by the correlation routines as the data becomes available. When the 600 iterations are completed the correlation coefficient calculations are completed, and a sorting process employed to determine the signum and amplitude of the maximum value. These are then used to determine the adjusted value of T_r which is then passed to the supervisor.

5.5.3)Reactive Power Measurement and Tr Adaption Algorithm

The algorithm for this strategy provides an adjustment for the rotor time constant every sample (2ms) on the basis of the discrepancy between the measured and demanded reactive power. The main consideration for the implementation of this algorithm is how the error function should be scaled in order to adjust T_r and whether some sort of integral control should be put onto the error function. A Proportional plus Integral controller was employed in order to remove steady state errors from this function.

5.5.4)Reactive Power OCCAM Implementation

The occam implementation of the algorithm is shown by the pseudo-code given here. It is implemented as a sequential program which iterates for 600 cycles.

```
SEQ      -- reactive power process
  SEQ i = [0 FOR 600]
    SEQ
      ... read data from supervisor
      ... pass data to supervisor
      ... calculate Fo and Fo*
      ... PI control of δFo
      ... calculate new Tr
```

Each calculation process is initiated by inputting the measured variables from the supervisory transputer, and outputting the new command variables , and certain calculation parameters to it. In this algorithm the value of T_r only is adjusted. When the values are input they are converted to real numbers and equations 2.25 and 2.26 employed to calculate the error function. This value is then used to drive a proportional plus integral controller (K = 0.5, w = 200) scaled (and limited if necessary) and the output is converted to an integer ready to be passed to the supervisor at the start of the next cycle. The adjustment to T_r is carried out by the supervisor.

IMPORTANT NOTES

(A) The following results were taken using an inverter which had an 8% imbalance on the voltage magnitude produced on each phase. The effects of this imbalance are significantly reduced when current feedback is employed, however it is an important factor when assessing the performance of the control system when running under voltage only type control (pages 112 - 116).

(B) The performance of the speed measurement technique is of importance when assessing the transient behaviour of the control systems, particularly at low speeds. The speed resolution can be described quantitatively as follows :-

At 1500 rpm - speed resolution = 0.2%

At 150 rpm - speed resolution = 2%

CHAPTER 6

EXPERIMENTAL PROCEDURES AND RESULTS

6.1) INTRODUCTION

This chapter will reproduce the experimental results obtained when the vector control networks are configured to the two motor drive systems outlined in Appendix 1. The design of the basic speed and (where applicable) current controllers is described, and simulation and test results for transient and steady-state operation of each control strategy on the wound and cage rotor motors are given. Noticeable problems with the coupling of the d-q axis current control loops are described as are the design and implementation of appropriate compensation. Signs of rotor time constant detuning are described and the implementation of the T_r identification routines is examined with suitable experimental verification. The final section discusses the transputer utilisation for the three networks implemented and provides information on the communication time and process time used by the individual devices.

6.2) THE DESIGN OF THE SPEED CONTROLLERS (ALL CONTROL STRATEGIES)

PI speed controllers were employed for both of the drive systems, to provide a speed control system with no steady-state error, and minimal overshoot during a transient. It was predicted that in most transient tests performed the output of the speed controller would be limited during most of the transient to avoid unacceptably large currents in the motor. For this reason more

elaborate controllers such as PID were felt to be unnecessary. The controllers were designed on the basis of the Laplace block diagram illustrated in Fig 6.1. This approach is based on the following assumptions :-

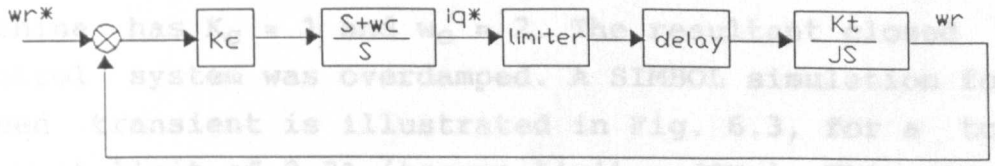
1: that a vector controlled AC motor is equivalent to a separately excited DC machine, with a constant field current in this application.

2: the inner current control loop's response time is negligible compared to the speed response time, and thus the torque is imposed immediately on the machine.

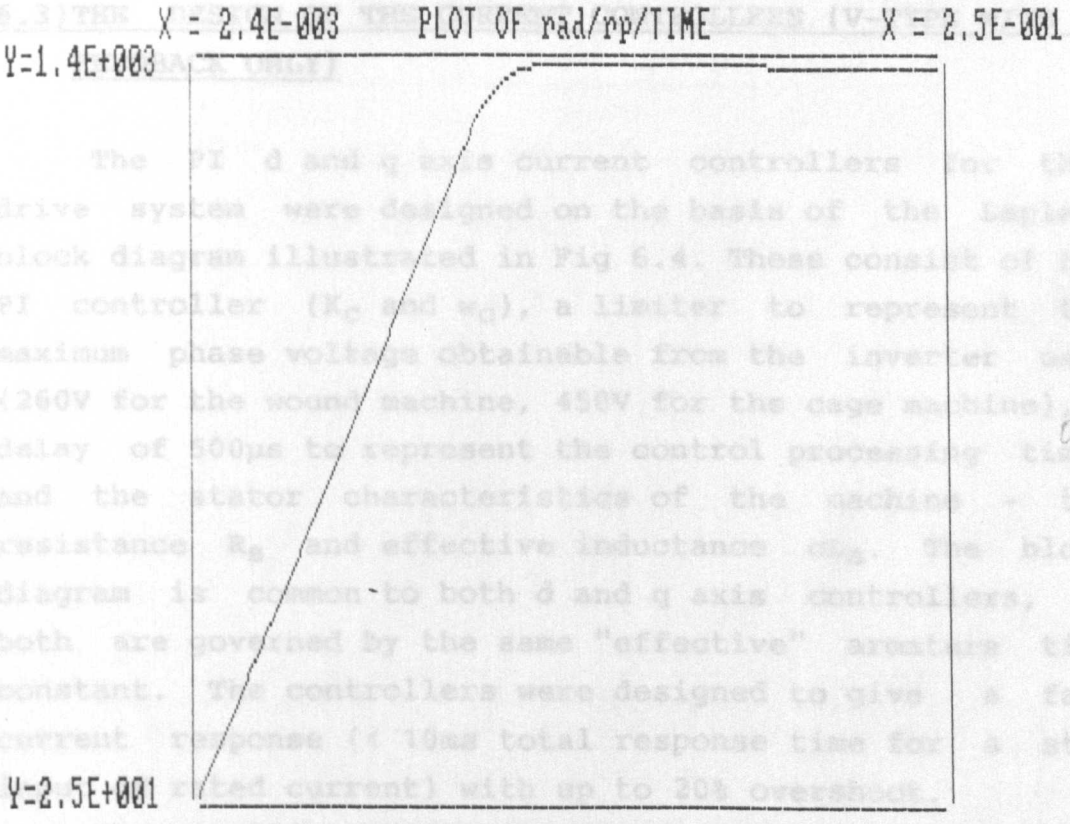
3: the motor load is purely inertial.

The SIMBOL CAD package [28], was used for the design of the controllers using Root Locus techniques, and allowed provision for sample delays, and limiters. This meant that a "mixed" system of s-plane and z-plane transforms could be used to more accurately model the drive. The main restriction of this package was that the saturation of integral elements of the controllers (or anti-wind-up) could not be included. The block diagram consists of the PI controller with gain K_c and angular frequency w_c, a limiter for the torque current reference, a delay of 500μs to represent the processing time of the controller, and the mechanical characteristics of the machine - the torque constant K_t and the inertia, J.

The resultant PI controller for the cage rotor machine had $K_c = 1$ and $w_c = 10$. The resultant closed loop control system was slightly underdamped. A simulation speed transient for a 4.5A torque current limit (torque limit = 34Nm) is shown in Fig. 6.2. This is the motor's rated current. It can be seen from this plot that the acceleration rate for rated torque is 14000rpm/s, which is to be expected considering the torque constant of the machine ($7.58NmA^{-1}$ for a 2.2A magnetising current) and the inertia ($0.023kgm^2$).



Laplace Block Diagram for the Design
of the Speed Controller
Figure 6.1.



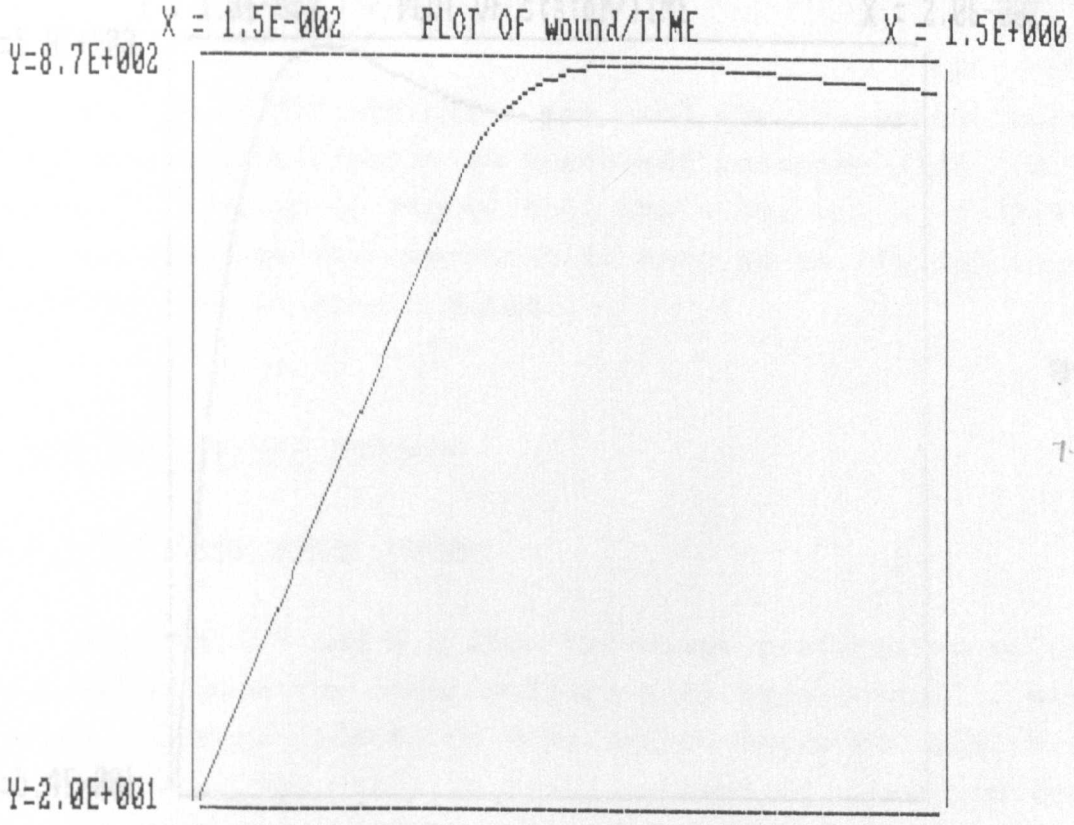
SYMBOL Simulation of a Speed Transient
(Cage Motor)
Figure 6.2.

The resultant PI controller for the wound rotor machine has K_c = 1 and w_c = 2. The resultant closed loop control system was overdamped. A SIMBOL simulation for a speed transient is illustrated in Fig. 6.3, for a torque current limit of 8.3A (torque limit = 45Nm). This was the rated current for the machine. The rated torque acceleration (with $K_t = 5.6NmA^{-1}$ for a 2.9A magnetising current, and $J = 0.32kgm^2$) as seen from the plot is 1333rpm/s. This is considerably less than that of the cage machine because the inertia of the wound machine is far greater.

6.3)THE DESIGN OF THE CURRENT CONTROLLERS (V-TYPE WITH I FEEDBACK ONLY)

The PI d and q axis current controllers for this drive system were designed on the basis of the Laplace block diagram illustrated in Fig 6.4. These consist of the PI controller (K_c and w_c), a limiter to represent the maximum phase voltage obtainable from the inverter used (260V for the wound machine, 450V for the cage machine), a delay of 500μs to represent the control processing time, and the stator characteristics of the machine - the resistance R_s and effective inductance σL_s. The block diagram is common to both d and q axis controllers, as both are governed by the same "effective" armature time constant. The controllers were designed to give a fast current response (< 10ms total response time for a step input of rated current) with up to 20% overshoot.

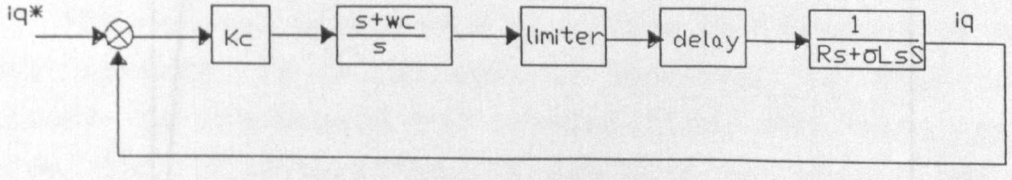
The resultant PI d and q axis current controllers for the cage rotor machine had K_c = 20 and w_c = 500 (R_s = 5.4Ω, σL_s = 0.026H). The resultant closed loop control system gave a simulated transient response with 7% overshoot, as illustrated in Figure 6.5. for rated current. The total response time (time to reach steady state conditions) was around 9ms.



SYMBOL simulation of a Speed Transient

(Wound Rotor Motor)

Figure 6.3.



Laplace Block Diagram for the Design

of the Current Controller

Figure 6.4.

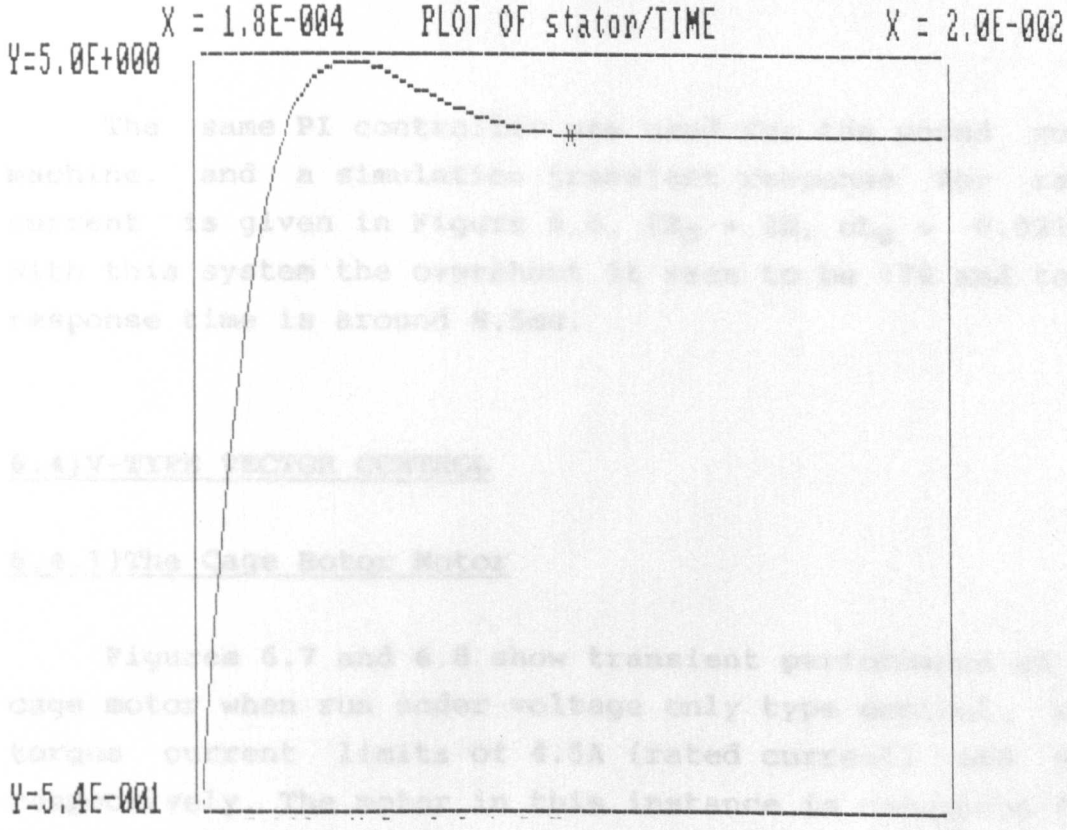


Figure 6.5. SIMBOL Simulation of a Current Transient (cage machine)

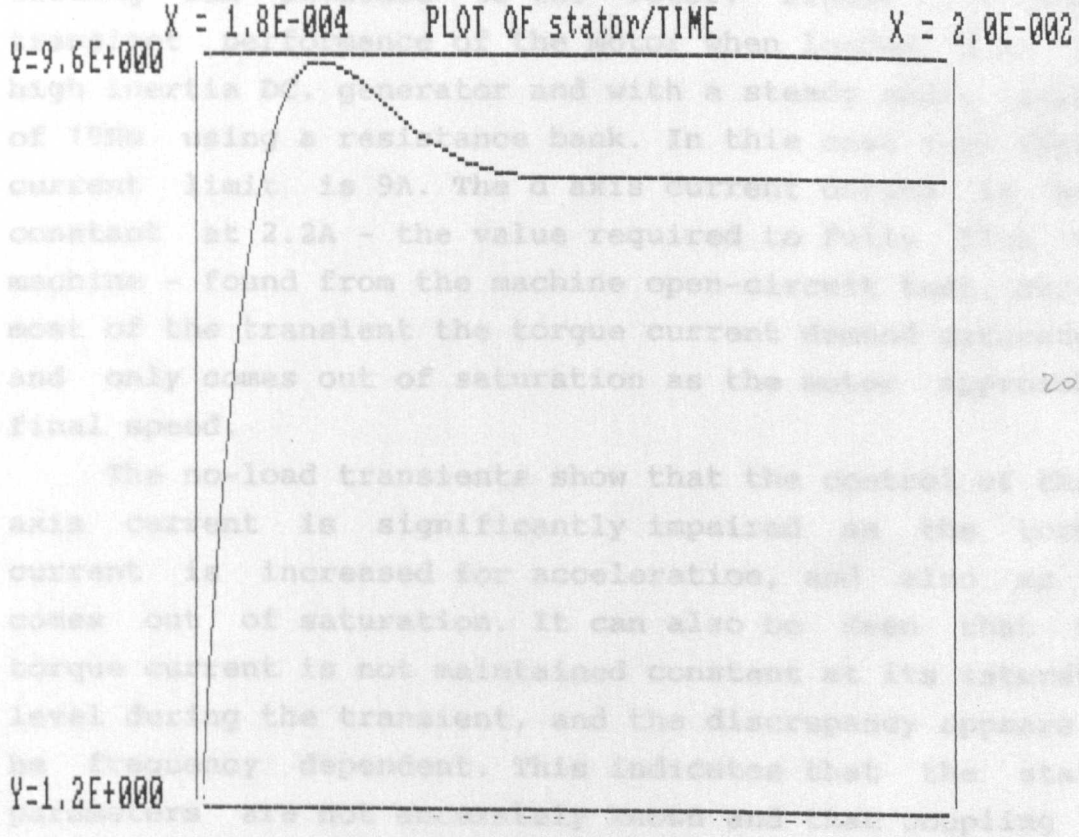


Figure 6.6. SIMBOL Simulation of a Current Transient (wound machine)

The same PI controller was used for the wound rotor machine, and a simulation transient response for rated current is given in Figure 6.6. (R_S = 2Ω, σL_S = 0.021H). With this system the overshoot it seen to be 17% and total response time is around 8.5ms.

6.4)V-TYPE VECTOR CONTROL

6.4.1)The Cage Rotor Motor

Figures 6.7 and 6.8 show transient performance of the cage motor when run under voltage only type control, with torque current limits of 4.5A (rated current) and 9.0A respectively. The motor in this instance is uncoupled from the DC generator and thus the load is merely the inertia of the motor itself (low) and the light load from the cooling fan attached to the rotor. Figure 6.9 shows transient performance of the motor when loaded with the high inertia DC. generator and with a steady state torque of 19Nm using a resistance bank. In this case the torque current limit is 9A. The d axis current demand is held constant at 2.2A - the value required to fully flux the machine - found from the machine open-circuit test. During most of the transient the torque current demand saturates, and only comes out of saturation as the motor approaches final speed.

The no-load transients show that the control of the d axis current is significantly impaired as the torque current is increased for acceleration, and also as it comes out of saturation. It can also be seen that the torque current is not maintained constant at its saturated level during the transient, and the discrepancy appears to be frequency dependent. This indicates that the stator parameters are not accurately known and that coupling is occurring between the controller d and q axis current demands and those actually imposed on the motor. As the

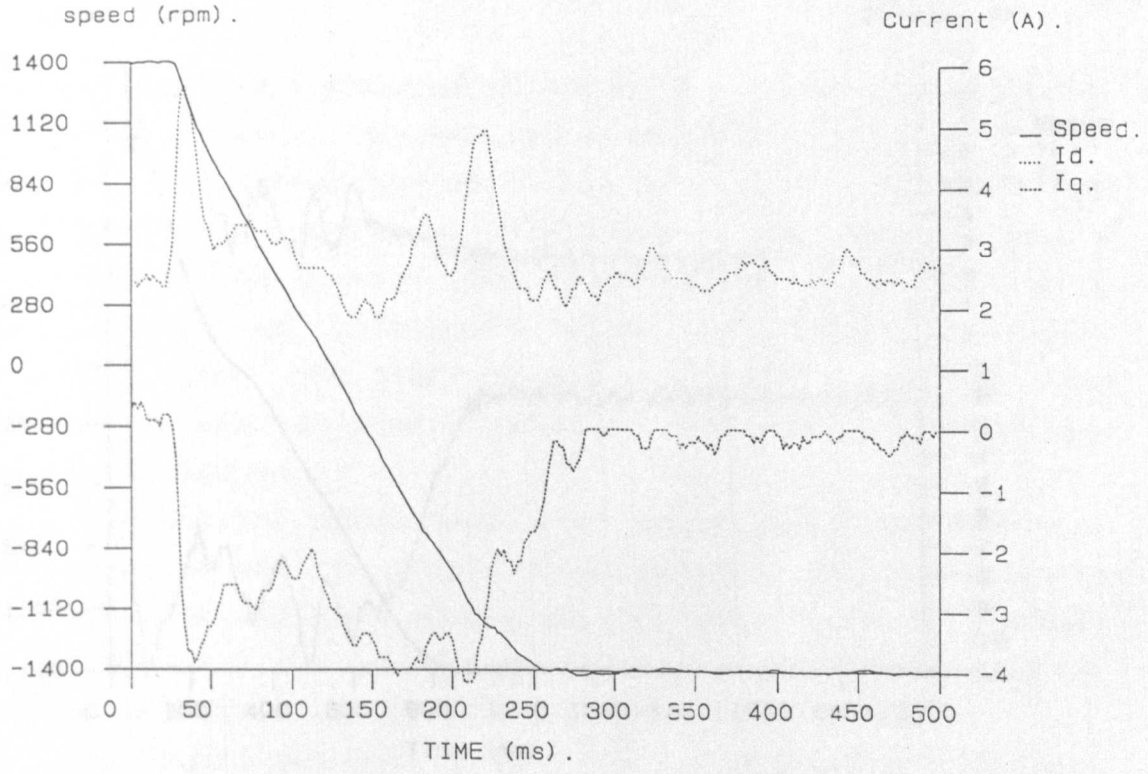


Figure 6.7. Speed Transient for the Unloaded Cage Motor with a 4.5A Torque Current Limit with V Type Control

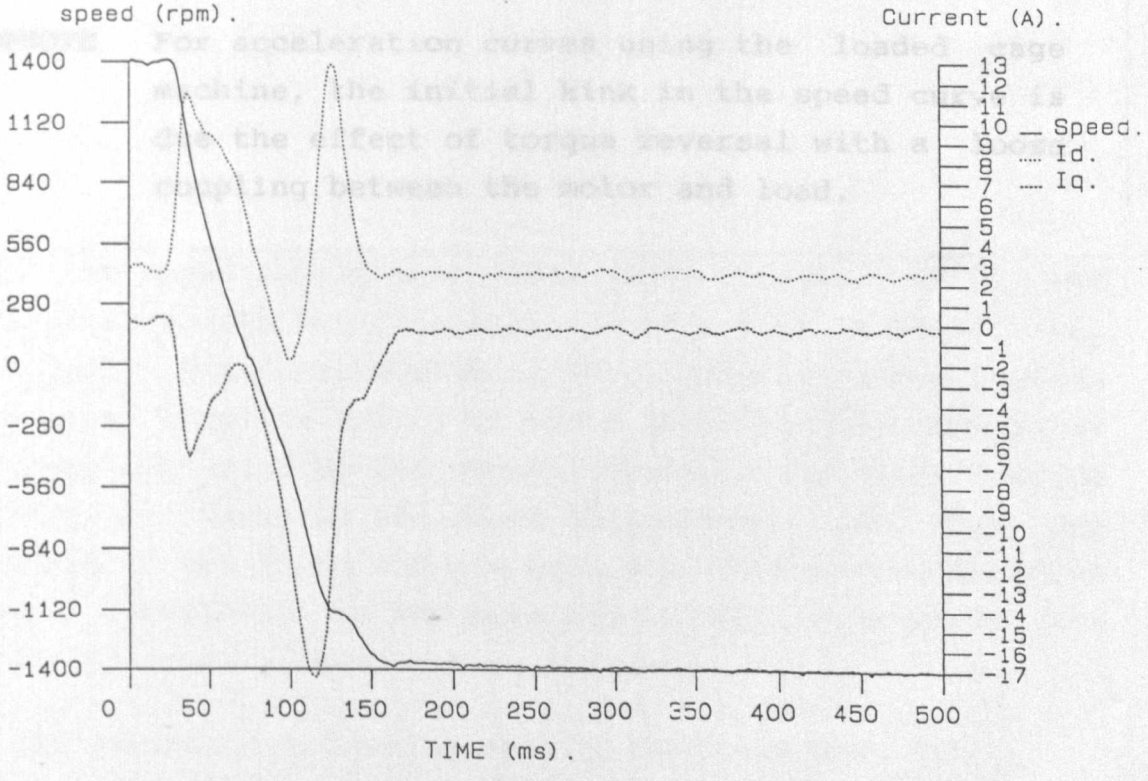


Figure 6.8. Speed Transient for the Unloaded Cage Motor with a 9A Torque Current Limit with V Type Control

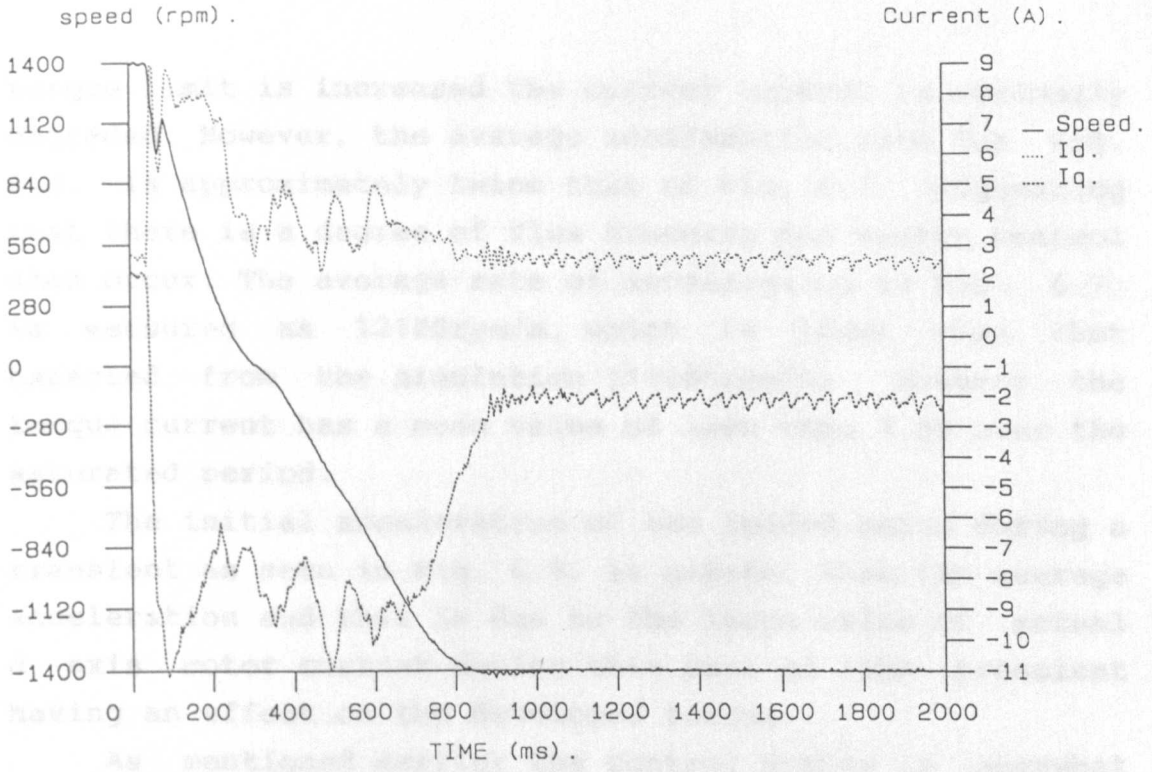


Figure 6.9. Speed Transient for the loaded Cage Motor with a 9A Torque Current Limit with V Type Control

FOOTNOTE For acceleration curves using the loaded cage machine, the initial kink in the speed curve is due the effect of torque reversal with a loose coupling between the motor and load.

torque limit is increased the current control is seriously degraded. However, the average acceleration rate for Fig. 6.8. is approximately twice that of Fig. 6.7. suggesting that there is a degree of flux tracking and vector control does occur. The average rate of acceleration in Fig. 6.7. is measured as 12100rpm/s, which is lower than that expected from the simulation (14000rpm/s). However the torque current has a mean value of less than 4.5A over the saturated period.

The initial acceleration of the loaded motor during a transient as seen in Fig. 6.9. is greater than the average acceleration and this is due to the large value of actual d axis motor current during this part of the transient having an effect on the developed torque.

As mentioned earlier the control method is somewhat crude as it relies heavily on an accurate knowledge of the stator parameters as well as the rotor time constant, which may change with frequency due to "skin depth" effects as well as temperature. It also assumes an accurate calibration of the PWM generator which is not correct during deceleration as the kinetic energy recovered from the rotor initially charges the link capacitor of the inverter to 3.4% above its nominal value (580V). It also has a "current control" sample rate equal to the speed sample rate (2ms) which is rather slow, and comparable with the effective armature time constant ($\sigma L_s \approx$ 5ms). A further source of inaccuracy at lower speeds results from the method of speed sampling. The number of pulses generated by the encoder during a set clock period (2ms) is taken as the speed measurement, and thus the effect of the "loss" of a single bit of the count on a low speed signal will be far more significant an error on the overall reading than at high speeds.

See important notes (A) and (B) opposite page 106

6.4.2)The Wound Rotor Induction Motor

Figure 6.10 illustrates a typical speed transient for the wound rotor motor running unloaded. The field current is maintained constant at 2.9A. Again this value was derived from the open-circuit machine test as the value to fully flux the machine. Rated torque current in this case is 8.3A. There is no precise control of the d and q axis currents during the transient and the oscillations in both currents can be seen as causing torque oscillations during the transient, which are represented as kinks on the acceleration curve. It has proved impossible to get a significantly higher acceleration current into the motor using this type of control.

The average acceleration rate as measured from Fig. 6.10 is 1300rpm/s which compares favourably with the simulation for rated current (1333 rpm/s).

This method of vector control may be crude but it is effective as a means of obtaining a good dynamic performance from the induction motor drive using a speed sensor only. The main problem arises from the peak currents generated during acceleration, which are not consistent with the controller values and may exceed the rating of the inverter used.

6.5)I-TYPE VECTOR CONTROL

6.5.1)The Cage Rotor Motor

Initial experimentation with the bang-bang controller was to ascertain whether lead-lag and d-q axis compensation was required for the current controllers. The results of Figures 6.11, 6.12, and 6.13 are taken for the

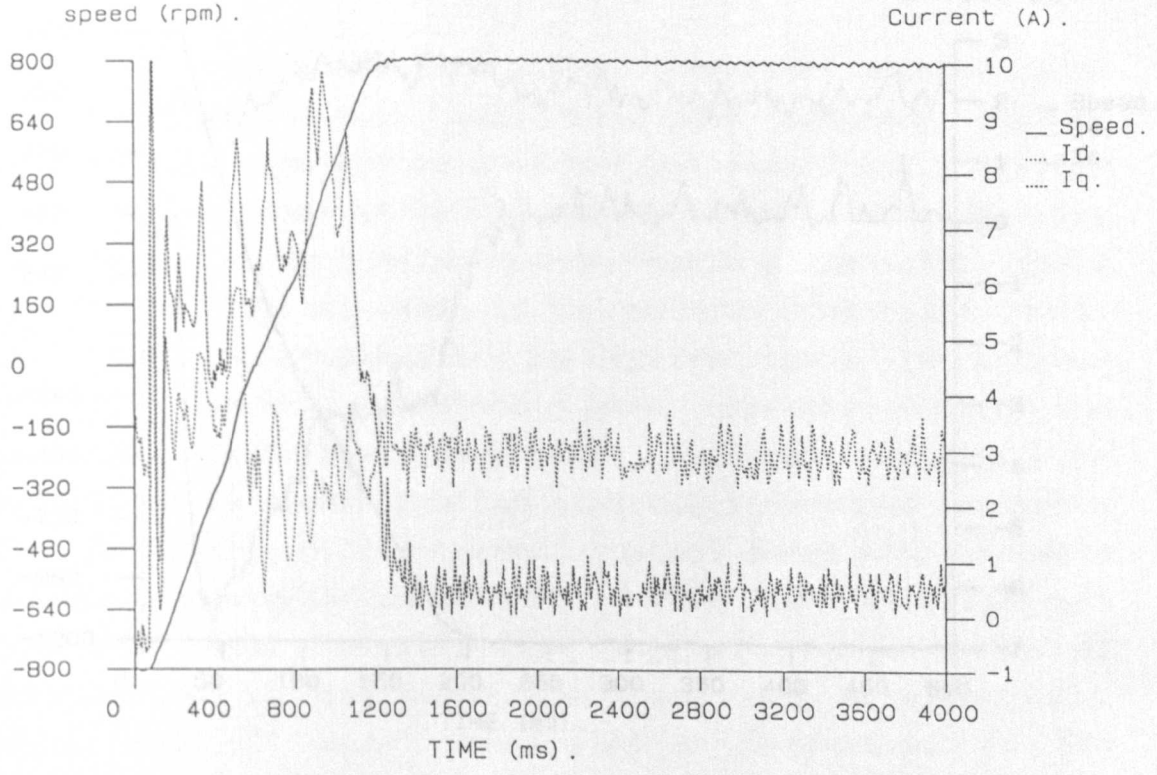


Figure 6.10 Speed Transient for the Unloaded Wound Motor with a 8.3A Torque Current Limit with V Type Control

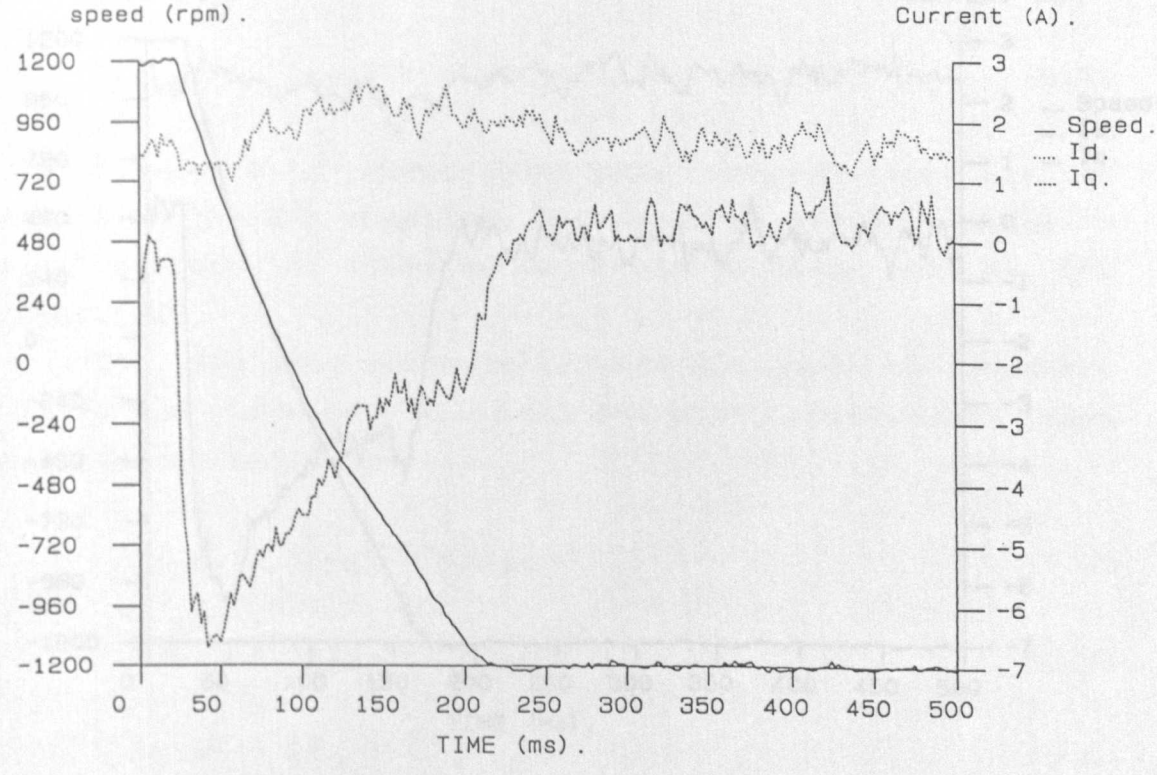


Figure 6.11 Speed transient for the Unloaded Cage Motor - I Type Control with no Compensation

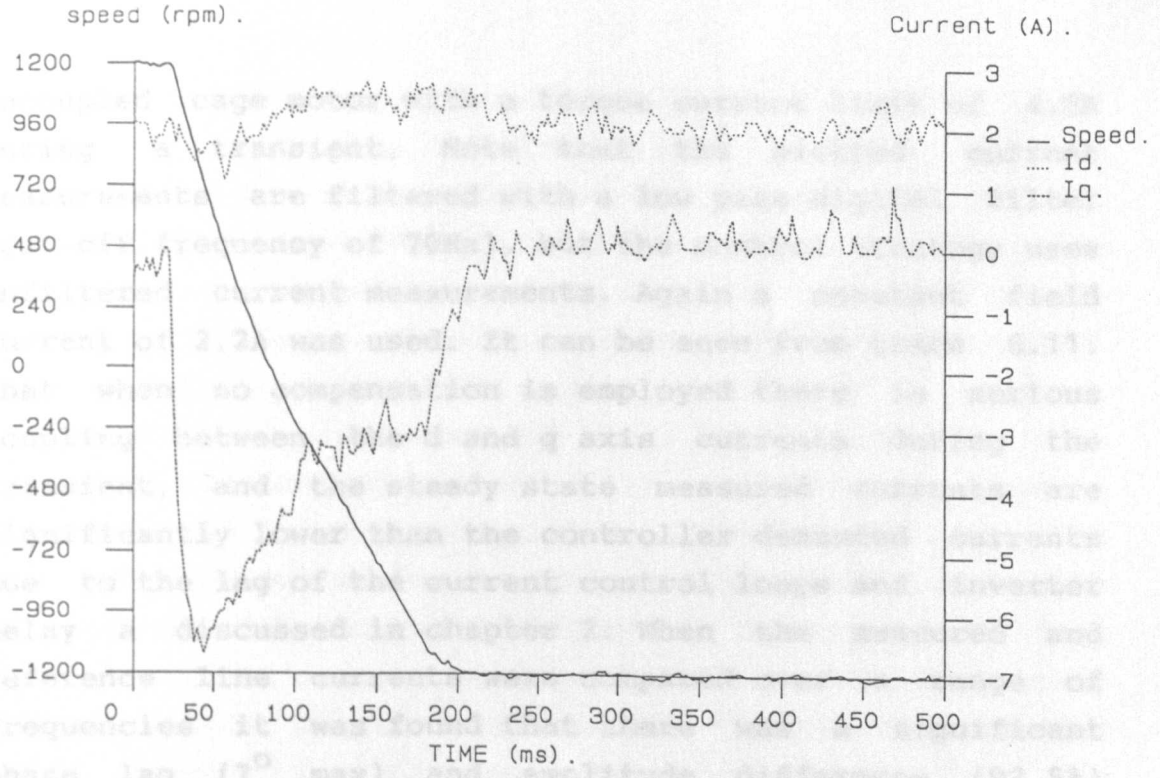


Figure 6.12 Speed transient for the Unloaded Cage Motor - I Type Control with Lead-Lag Compensation

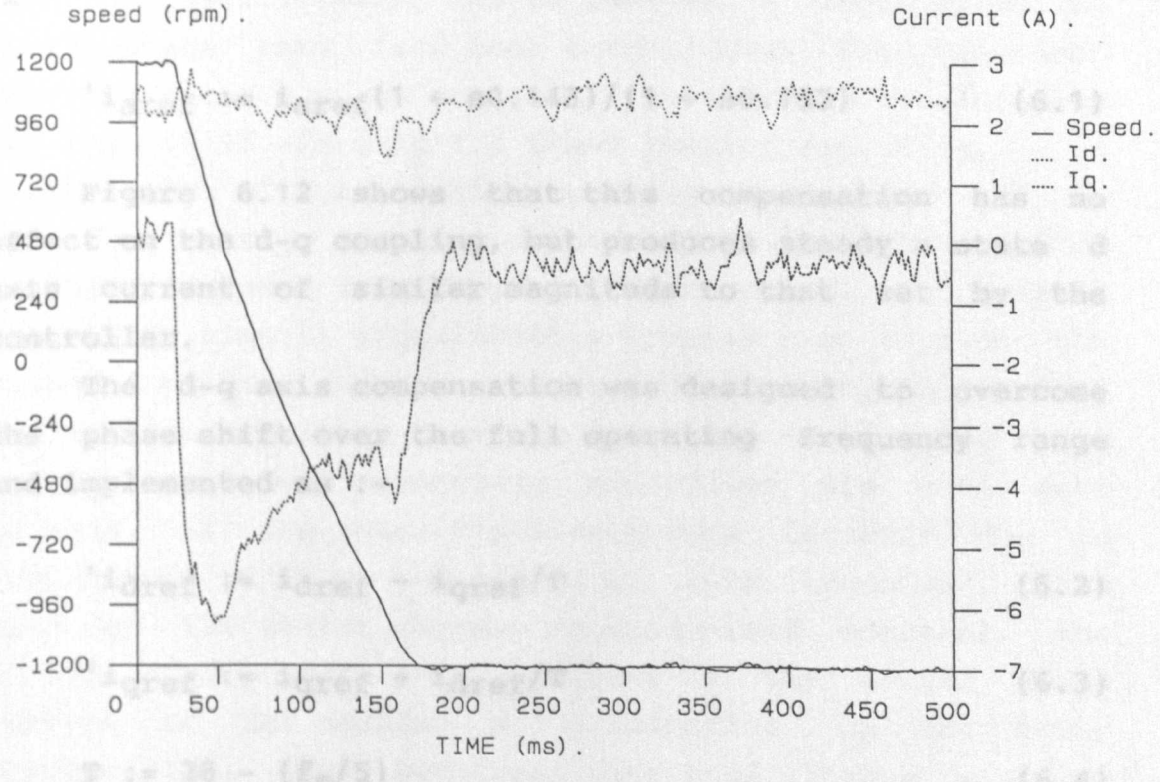


Figure 6.13 Speed transient for the Unloaded Cage Motor - I Type Control with Lead-Lag and D-Q Axis Compensation

uncoupled cage motor with a torque current limit of 4.5A during a transient. Note that the plotted current measurements are filtered with a low pass digital filter (cut-off frequency of 70Hz), but the control strategy uses unfiltered current measurements. Again a constant field current of 2.2A was used. It can be seen from trace 6.11. that when no compensation is employed there is serious coupling between the d and q axis currents during the transient, and the steady state measured currents are significantly lower than the controller demanded currents due to the lag of the current control loops and inverter delay a discussed in chapter 2. When the measured and reference line currents were compared over a range of frequencies it was found that there was a significant phase lag (7^{o} max) and amplitude difference (92.5%) between them. The lead-lag compensators were designed to overcome the amplitude discrepancy and gave a gain of 1.08 to frequencies above 1Hz. The resultant compensation was :-

$$'i_{aref} := i_{aref}(1 + s0.143)/(1 + s0.133) \qquad (6.1)$$

Figure 6.12 shows that this compensation has no effect on the d-q coupling, but produces steady a state d axis current of similar magnitude to that set by the controller.

The d-q axis compensation was designed to overcome the phase shift over the full operating frequency range and implemented as :-

$$'i_{dref} := i_{dref} - i_{qref}/T \qquad (6.2)$$

$$'i_{qref} := i_{qref} + i_{dref}/T \qquad (6.3)$$

$$T := 28 - (f_e/5) \qquad (6.4)$$

It can be seen from Fig 6.13. that there is a significant reduction of the d-q axis coupling effects using this form of compensation, but the problem is not completely removed. No further improvement could be made on the q axis current control as degradation of the d axis control would occur. It must be noted that the 4kHz sample rate used for the bang-bang controller is slow.

Transient performance of the I Type controller for differing current and load conditions is shown in Figures 6.14 and 6.15 for the uncoupled motor and Figure 6.16 for the motor loaded as for V type control. The acceleration of the unloaded machine is noticeably better than V Type control as the d and q axis currents are controlled more accurately over the transient period. The average rate for rated current is 16500rpm/s which is higher than the simulation, but in this case the mean torque current over the saturated period is greater than 4.5A. The currents do not fully follow the controller values and the resultant loss of flux tracking can be seen as a degradation of acceleration rate (and thus torque) over the transient period. It can also be seen that the average acceleration of Fig. 6.15. is only 1.5 times that of Fig. 6.14. - a further sign of loss of flux tracking.

The average acceleration of Fig. 6.16. (the loaded machine) is worse than that of V Type control. The overall transient time is significantly greater than that of the unloaded machine and thus the loss of tracking and torque degradation resulting from the fact that the d and q axis currents are not precisely controlled are much more obvious. If the rotor resistance does increase due to temperature during the period of this transient, then although the stator current is maintained constant, the change will result in an increase in the actual field current of the machine, and a reduction on the rotor current [15]. The resultant generated torque will be reduced, because the speed controller has not been modified to account for the change in R_r.

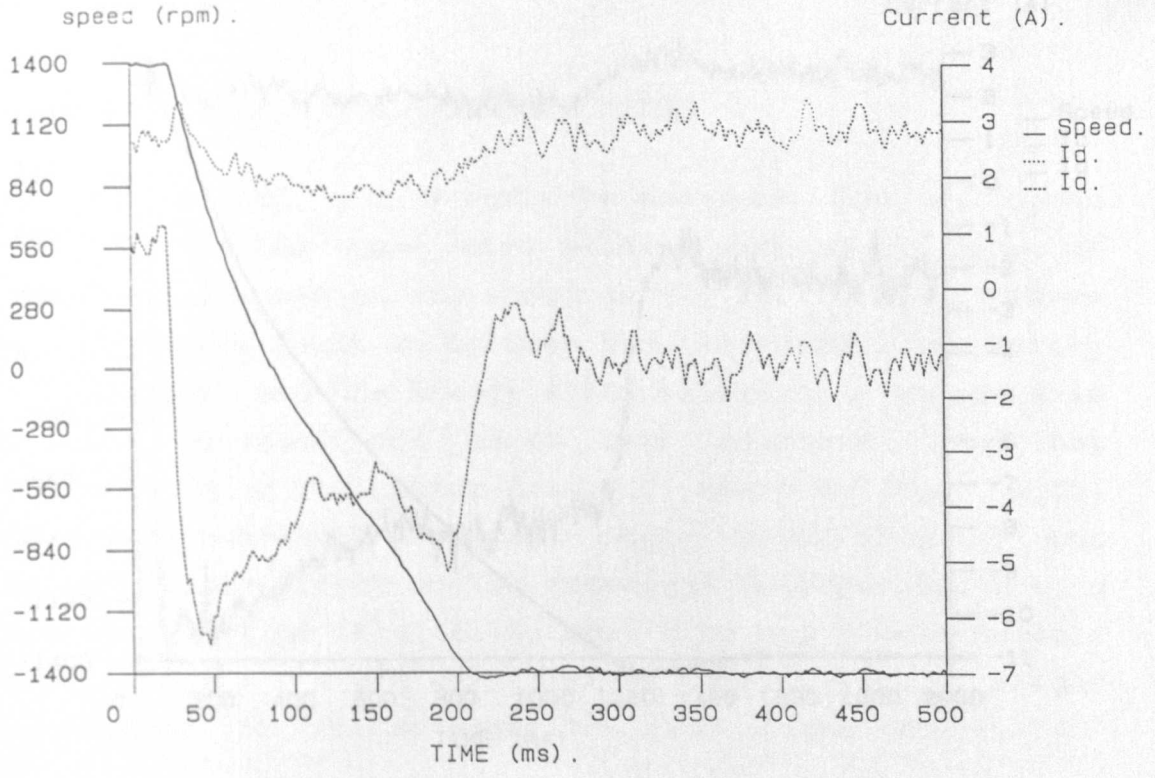


Figure 6.14 Speed Transient for the Unloaded Cage Motor with a 4.5A Torque Current Limit with I Type Control

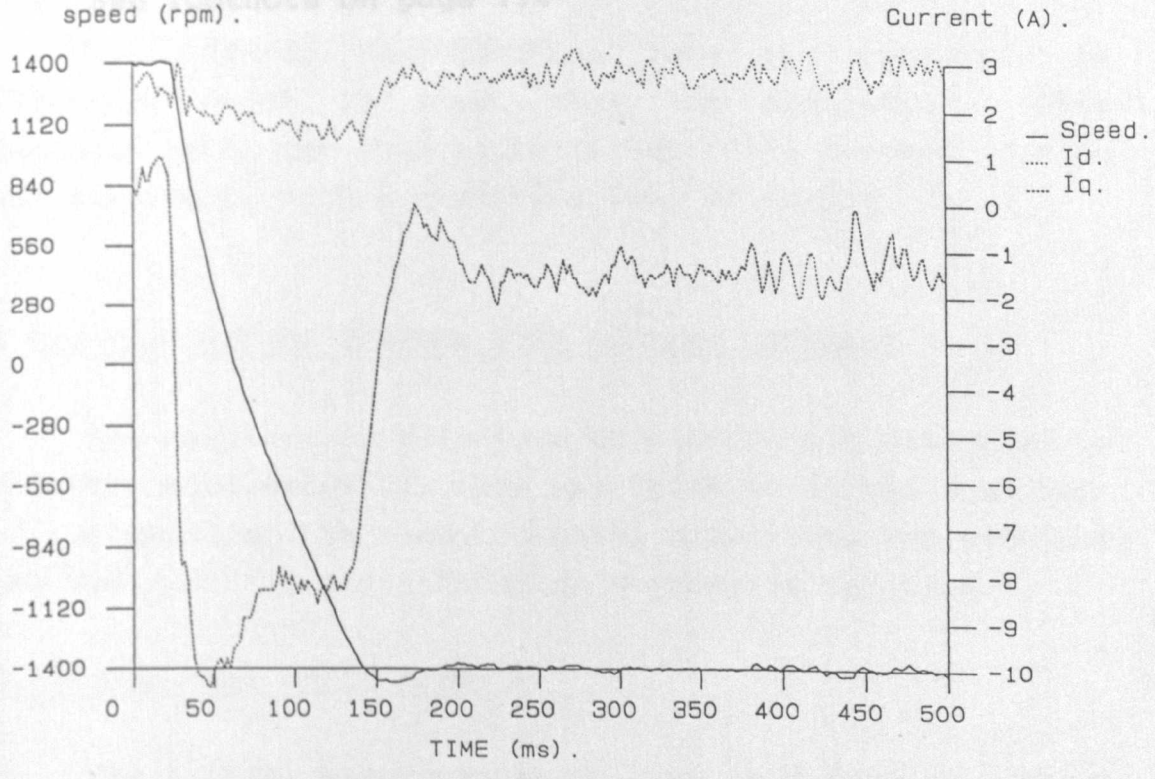


Figure 6.15 Speed Transient for the Unloaded Cage Motor with a 9A Torque Current Limit with I Type Control

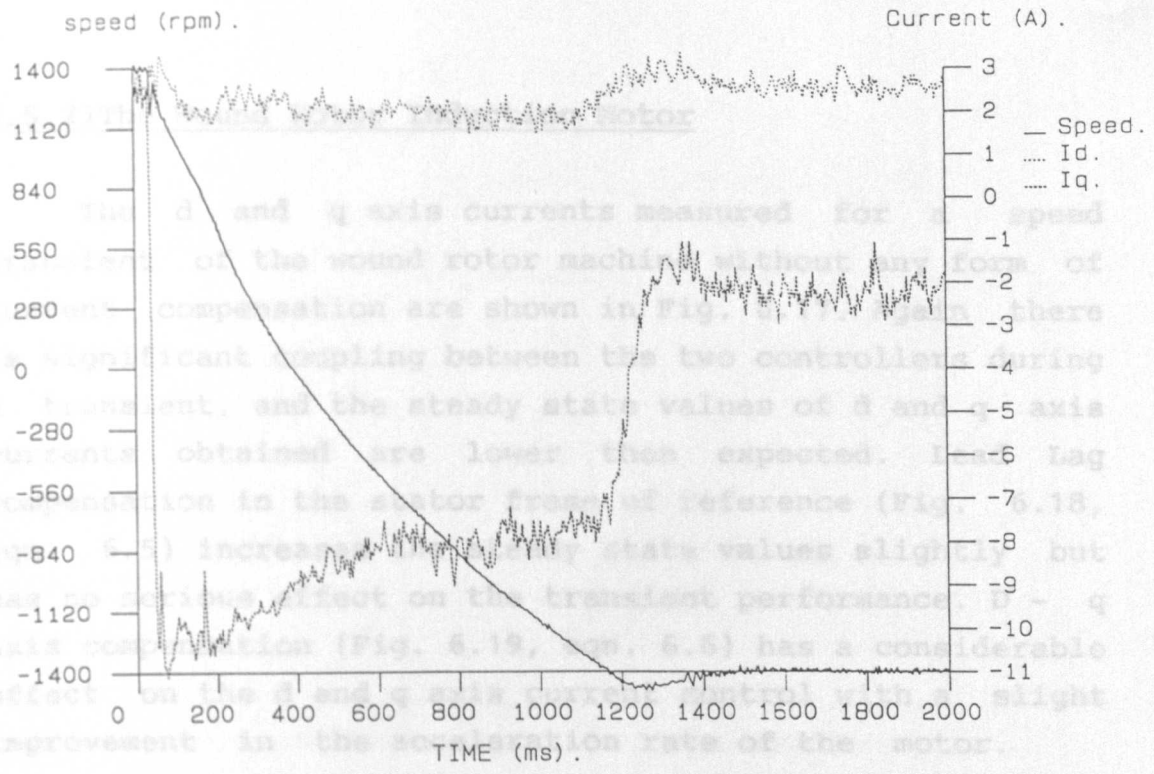


Figure 6.16 Speed Transient for the loaded Cage Motor with a 9A Torque Current Limit with I Type Control

See footnote on page 114

6.5.2)The Wound Rotor Induction Motor

The d and q axis currents measured for a speed transient of the wound rotor machine without any form of current compensation are shown in Fig. 6.17. Again there is significant coupling between the two controllers during a transient, and the steady state values of d and q axis currents obtained are lower than expected. Lead Lag compensation in the stator frame of reference (Fig. 6.18, eqn. 6.5) increases the steady state values slightly but has no serious effect on the transient performance. D - q axis compensation (Fig. 6.19, eqn. 6.6) has a considerable effect on the d and q axis current control with a slight improvement in the acceleration rate of the motor.

$$'i_{aref} := i_{aref}(1 + s0.159)/(1 + s0.138) \qquad (6.5)$$

$$T = 20 - (f_e/5) \qquad (6.6)$$

The average acceleration rate as measured is 1230rpm/s which is lower than the simulation. This suggests that the flux angle is not fully tracked during the transient, with a resulting loss of torque.

6.6)V-TYPE VECTOR CONTROL WITH CURRENT FEEDBACK

The results for this form of control are discussed in slightly more detail as this was found to be the best form of implementing the vector control algorithm, and provided the basis of the identification strategies employed.

6.6.1)The Cage Rotor Motor

The initial experimentation with this form of control was concerned with the need for employing d-q axis compensation to the output of the two current controllers.

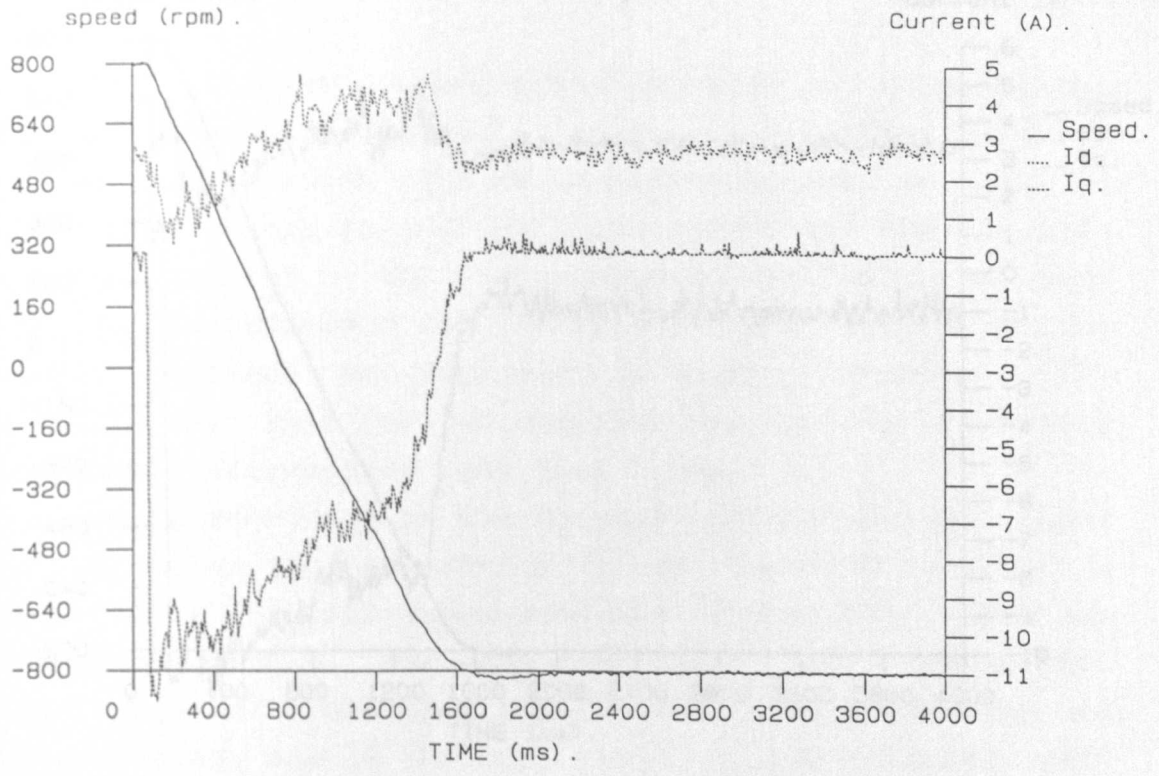


Figure 6.17 Speed transient for the Unloaded Wound Motor - I Type Control with no Compensation

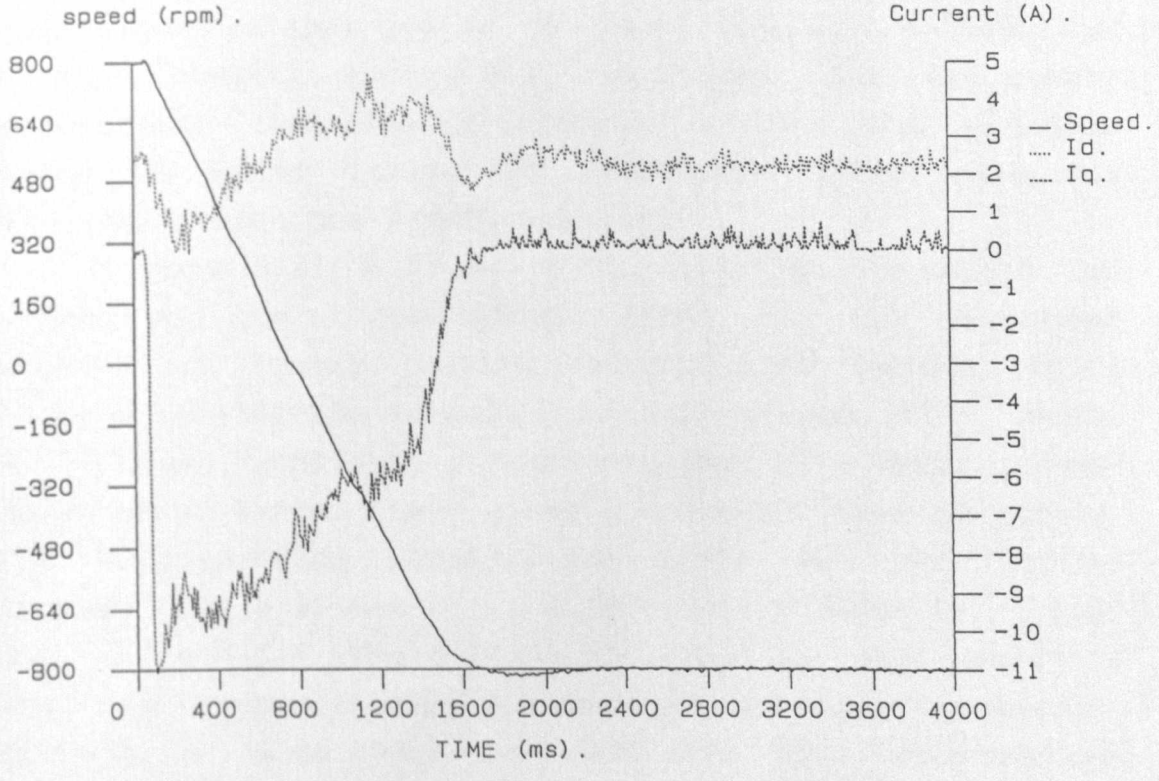


Figure 6.18 Speed transient for the Unloaded Wound Motor - I Type Control with Lead-Lag Compensation

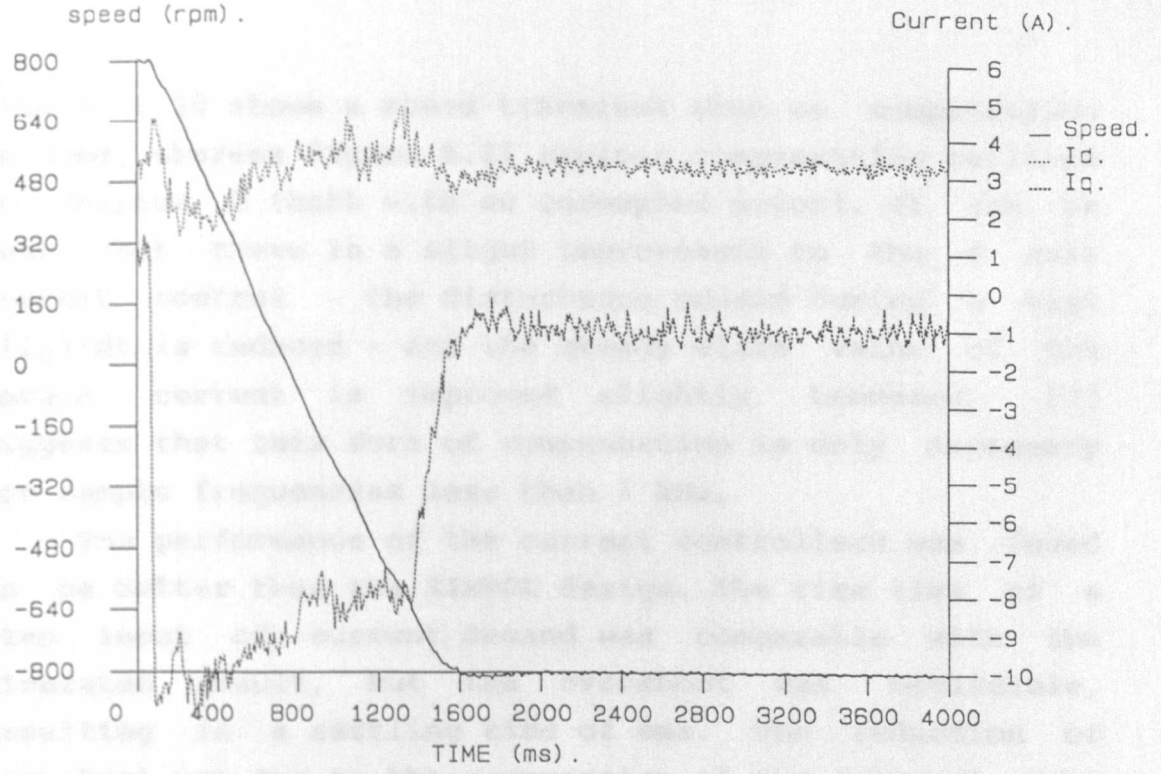


Figure 6.19 Speed transient for the Unloaded Wound Motor - I Type Control with Lead-Lag and D-Q Axis Compensation

Figure 6.20 shows a speed transient when no compensation is used, whereas Figure 6.21 employs compensation outlined in chapter 2 (both with an uncoupled motor). It can be seen that there is a slight improvement to the d axis current control - the disturbance caused during a high $d(i_q)/dt$ is reduced - and the steady state value of the torque current is improved slightly. Leonhard [7] suggests that this form of compensation is only necessary for sample frequencies less than 1 kHz.

The performance of the current controllers was found to be better than the SIMBOL design. The rise time of a step input of current demand was comparable with the simulated result, but the overshoot was negligible, resulting in a settling time of 6ms. The reduction of overshoot was due to the suppression of the integral part of the controller during controller saturation, a feature which could not be included on the SIMBOL design.

Figures 6.21 and 6.22 (uncoupled motor), and 6.23 (motor loaded as before) illustrate the transient performance of this system for comparison with V Type and I Type control. Torque and flux current are accurately controlled. The rate of acceleration for the unloaded motor at rated torque is 14000rpm/s which compares favourably with the SIMBOL simulation.

Figures 6.21, 6.22 and 6.24 illustrate the effect of increasing the torque current limit for the uncoupled machine. A torque current limit of 4.5A results in a 1400rpm to -1400rpm average acceleration time of ≈ 200ms. A 9A limit results in a transient time of ≈ 100ms. Three times rated torque limit gives a transient time of ≈ 80ms. The corresponding acceleration times for the loaded machine are ≈ 1600ms (Fig. 6.25 4.5A), ≈ 800ms (Fig 6.26 9A) and ≈ 620ms (Fig 6.27 13.5A). This is good evidence that the vector control strategy is working effectively. However it also shows that when very high acceleration currents are employed some slight detuning of the rotor time constant occurs (three times rated torque current

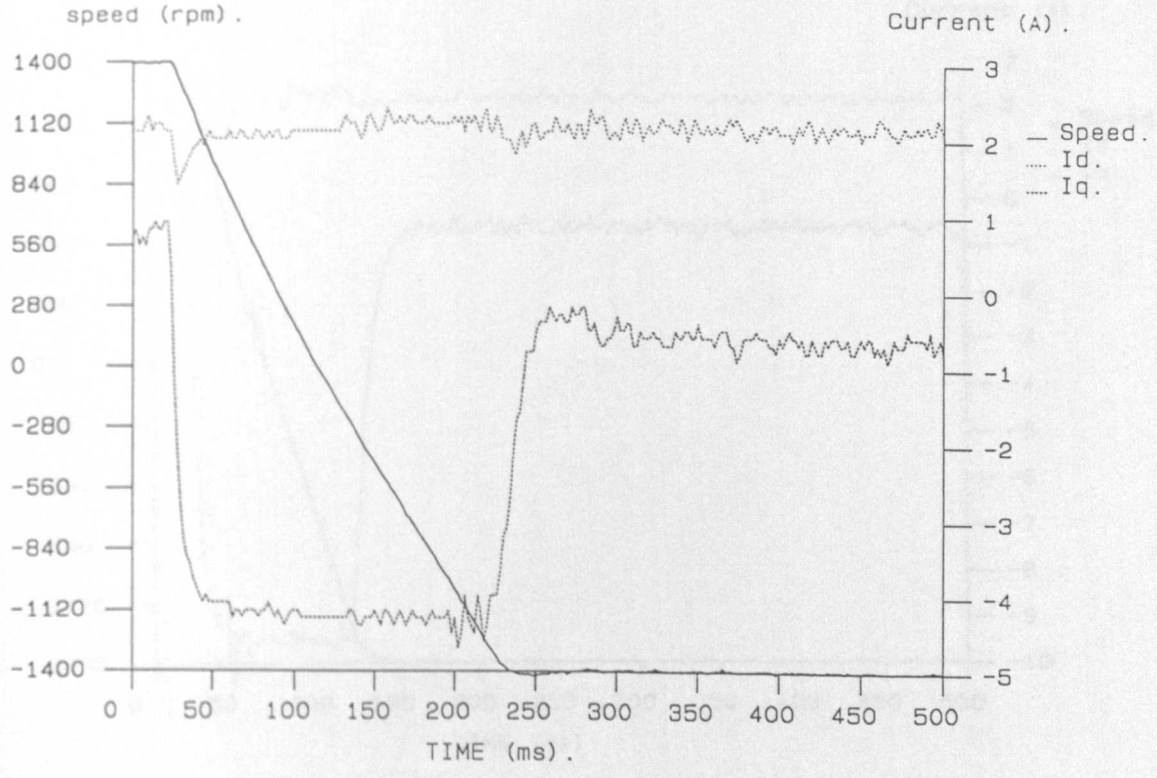


Figure 6.20 Speed transient for the Unloaded Cage Motor - VI Type Control with no Compensation

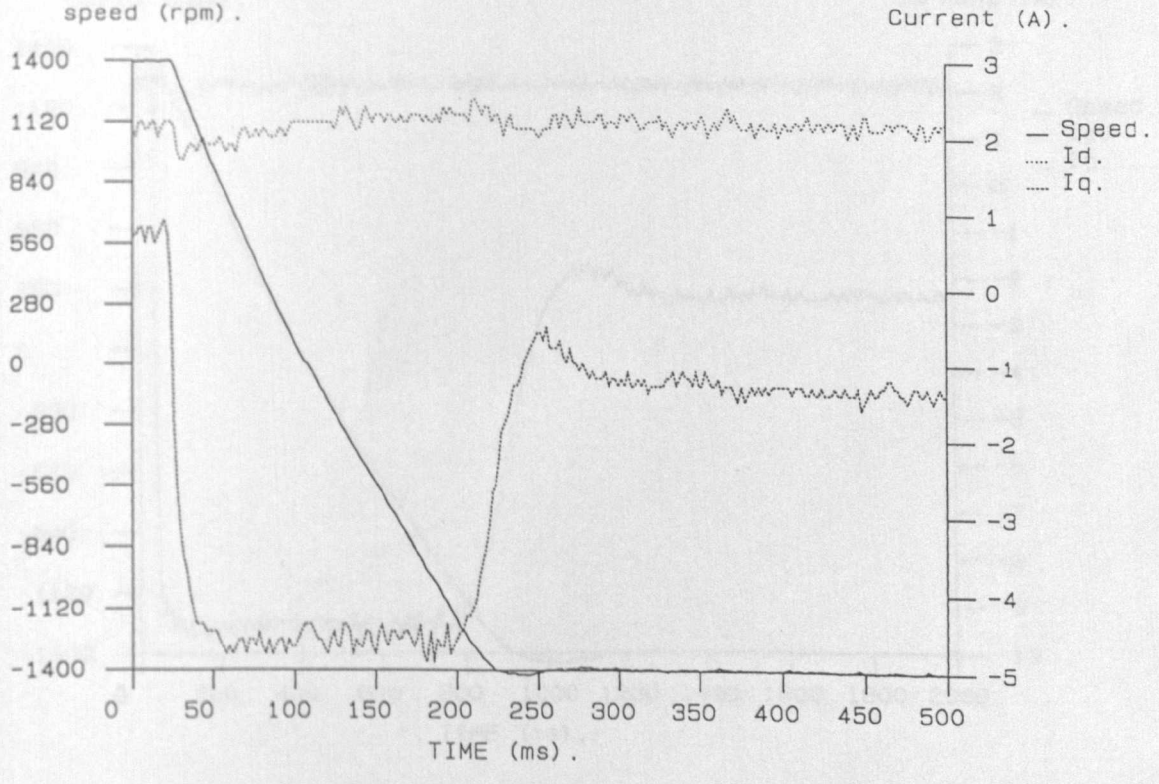


Figure 6.21 Speed transient for the Unloaded Cage Motor - VI Type Control with D-Q Axis Compensation

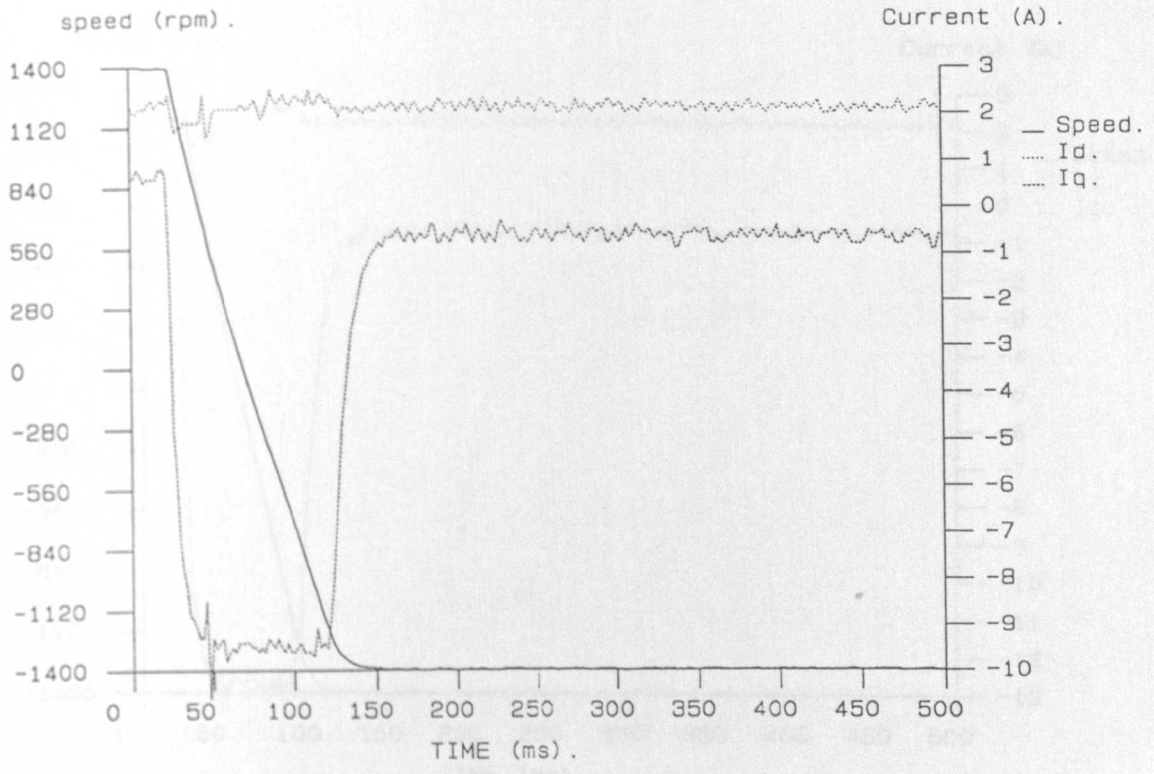


Figure 6.22 Speed Transient for the Unloaded Cage Motor with a 9A Torque Current Limit with VI Type Control

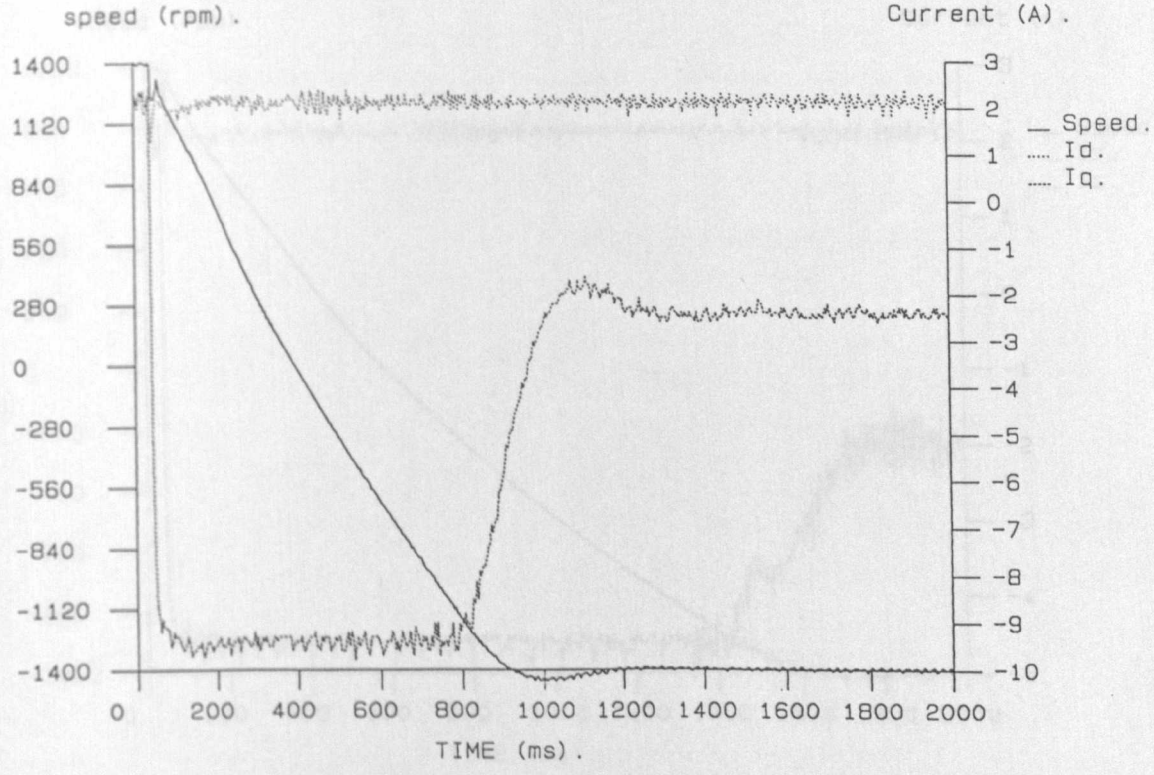


Figure 6.23 Speed Transient for the loaded Cage Motor with a 9A Torque Current Limit with VI Type Control

See footnote on page 114

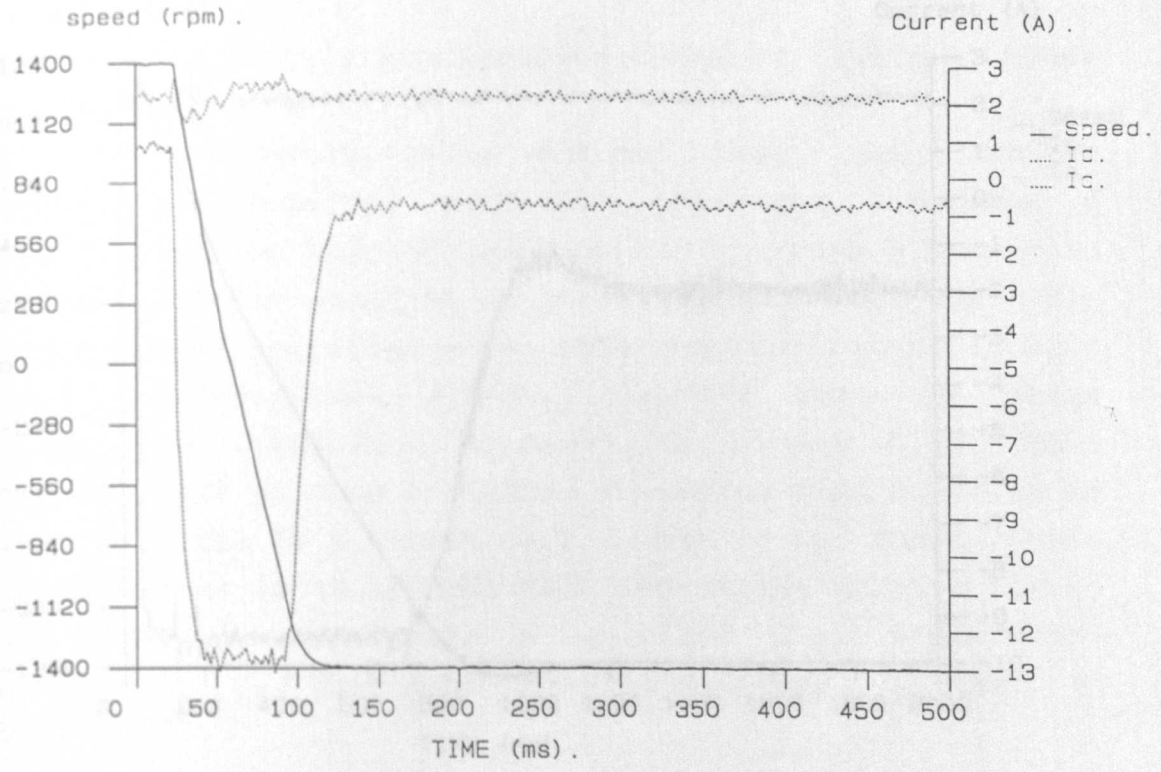


Figure 6.24 Speed Transient for the Unloaded Cage Motor with a 13.5A Torque Current Limit with VI Type Control

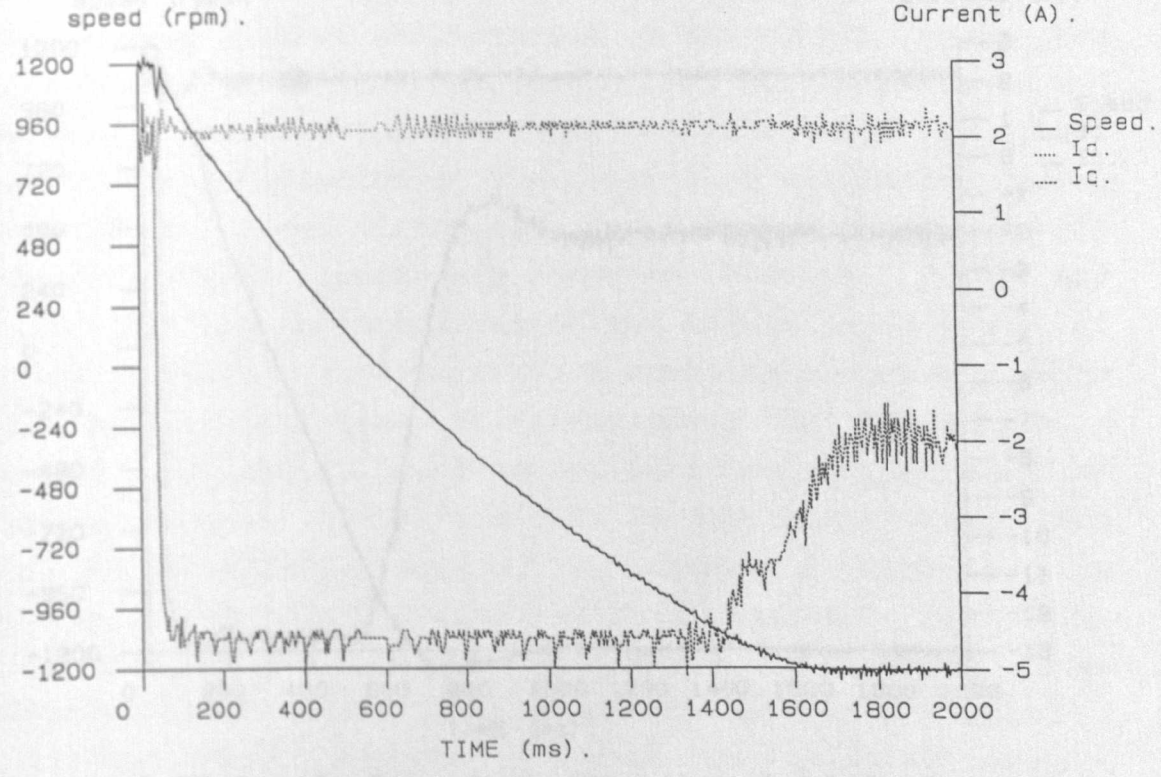


Figure 6.25 Speed Transient for the loaded Cage Motor with a 4.5A Torque Current Limit with VI Type Control

See footnote on page 114

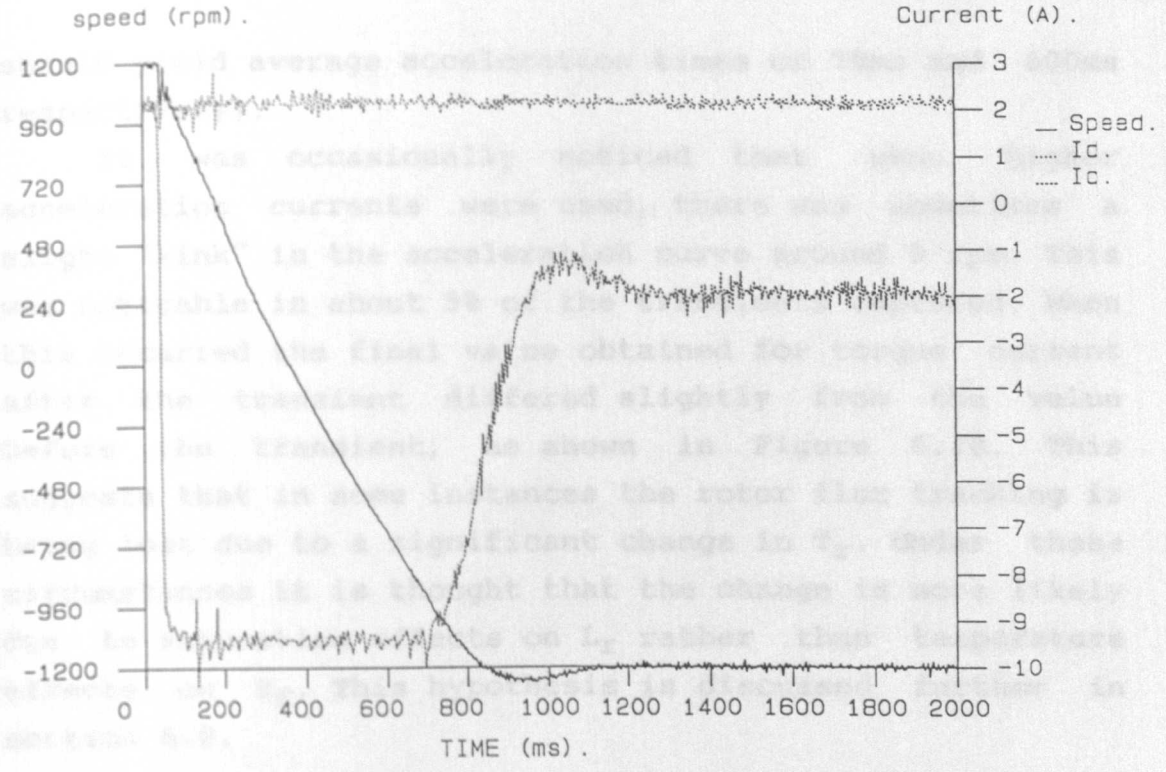


Figure 6.26 Speed Transient for the loaded Cage Motor with a 9A Torque Current Limit with VI Type Control

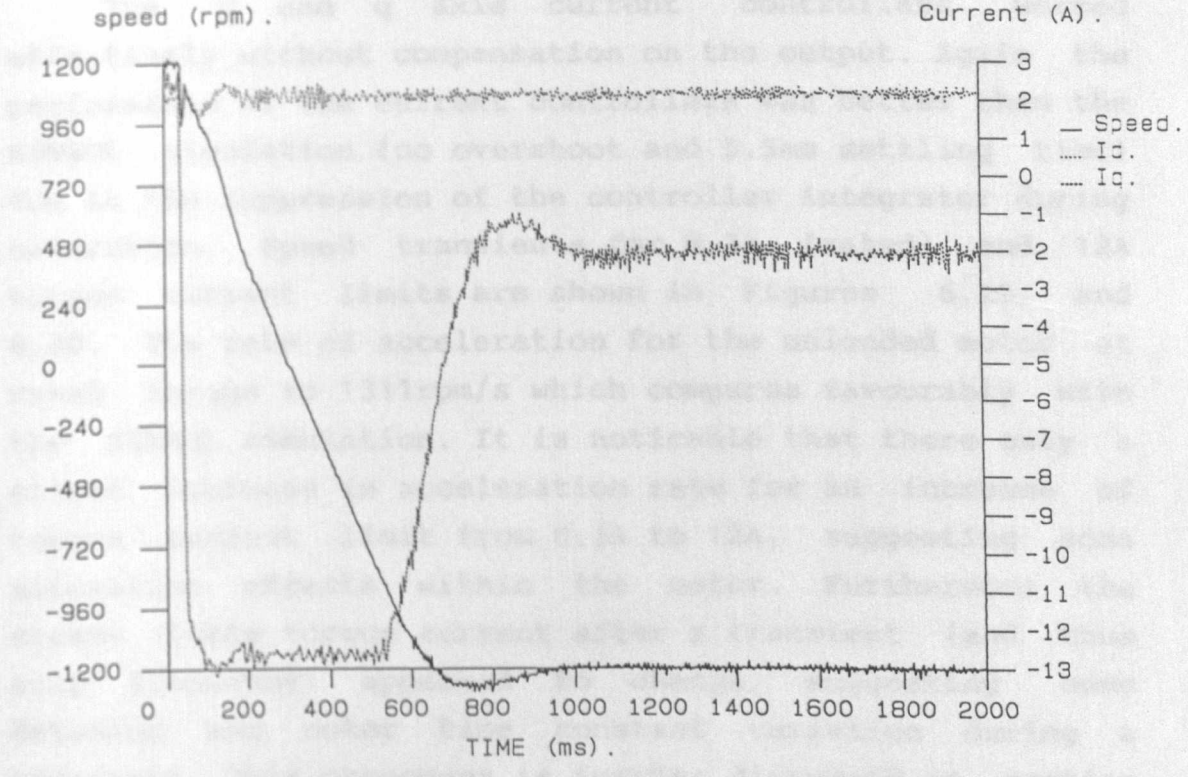


Figure 6.27 Speed Transient for the loaded Cage Motor with a 13.5A Torque Current Limit with VI Type Control

See footnote on page 114

should yield average acceleration times of 75ms and 600ms respectively).

It was occasionally noticed that when higher acceleration currents were used, there was sometimes a slight "kink" in the acceleration curve around 0 rpm. This was noticable in about 5% of the transients captured. When this occurred the final value obtained for torque current after the transient differed slightly from the value before the transient, as shown in Figure 6.28. This suggests that in some instances the rotor flux tracking is being lost due to a significant change in T_r. Under these circumstances it is thought that the change is more likely due to saturation effects on L_r rather than temperature effects on R_r. This hypothesis is discussed further in section 6.8.

6.6.2)The Wound Rotor Induction Motor

The d and q axis current controllers worked effectively without compensation on the output. Again the performance of the current controllers was better than the SIMBOL simulation (no overshoot and 5.5ms settling time) due to the suppression of the controller integrator during saturation. Speed transients for 8.3A (rated) and 12A torque current limits are shown in Figures 6.29, and 6.30. The rate of acceleration for the unloaded motor at rated torque is 1311rpm/s which compares favourably with the SIMBOL simulation. It is noticable that there only a slight increase in acceleration rate for an increase of torque current limit from 8.3A to 12A, suggesting some saturation effects within the motor. Furthermore the steady state torque current after a transient (and thus slip frequency) appeared to change, suggesting some detuning and rotor time constant variation during a transient. This phenomena is further discussed in section 6.8.

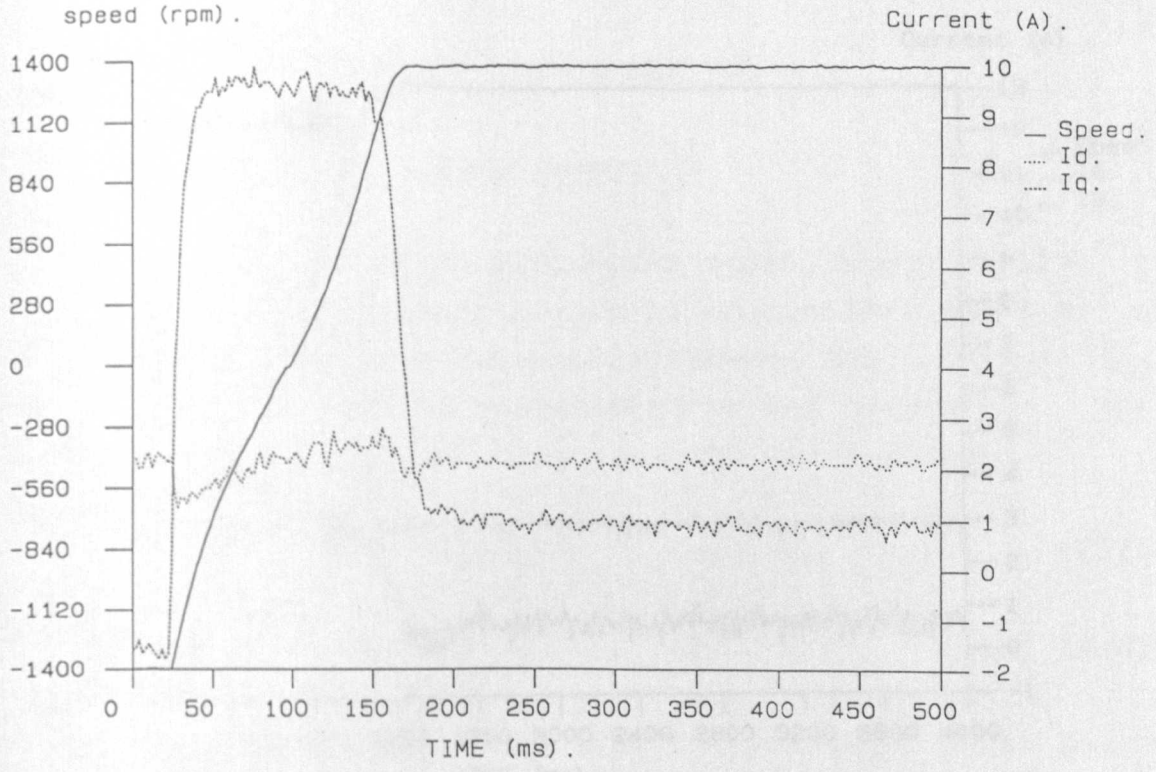


Figure 6.28 Speed Transient for the Unloaded Cage Motor with a 9A Torque Current Limit with VI Type Control Showing Loss of Flux Tracking.

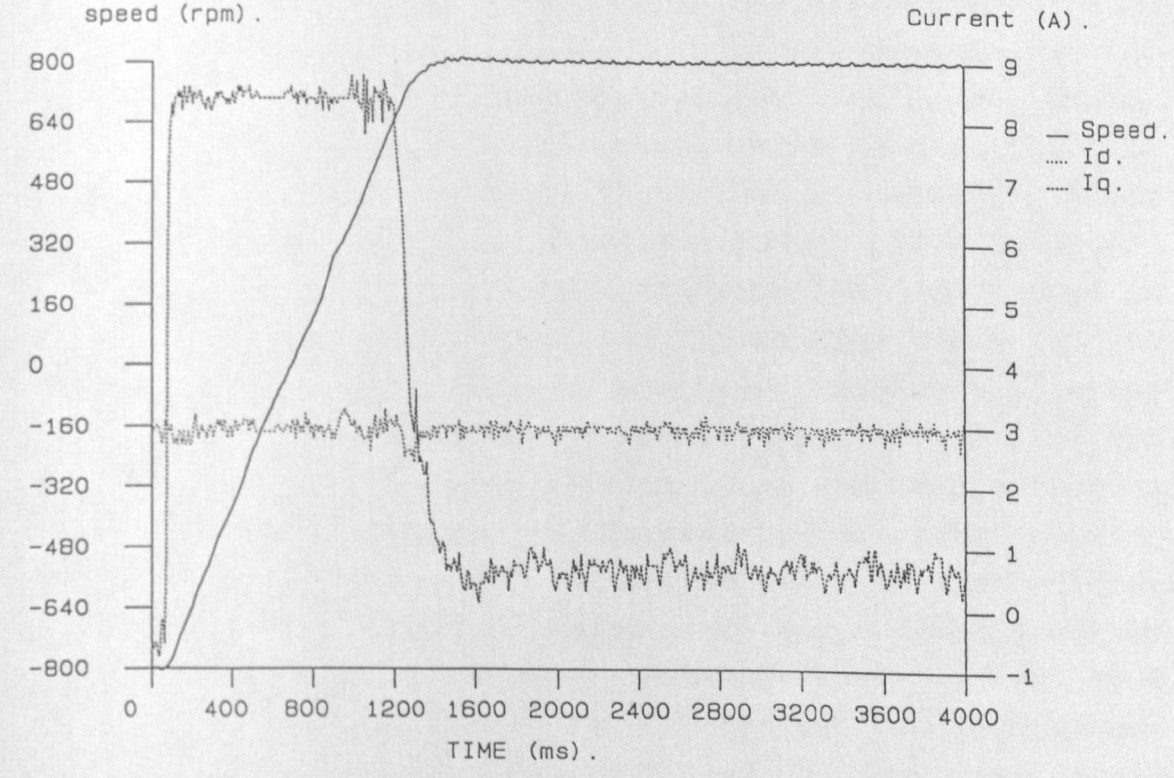


Figure 6.29 Speed Transient for the Unloaded Wound Motor with a 8.3A Torque Current Limit with VI Type Control

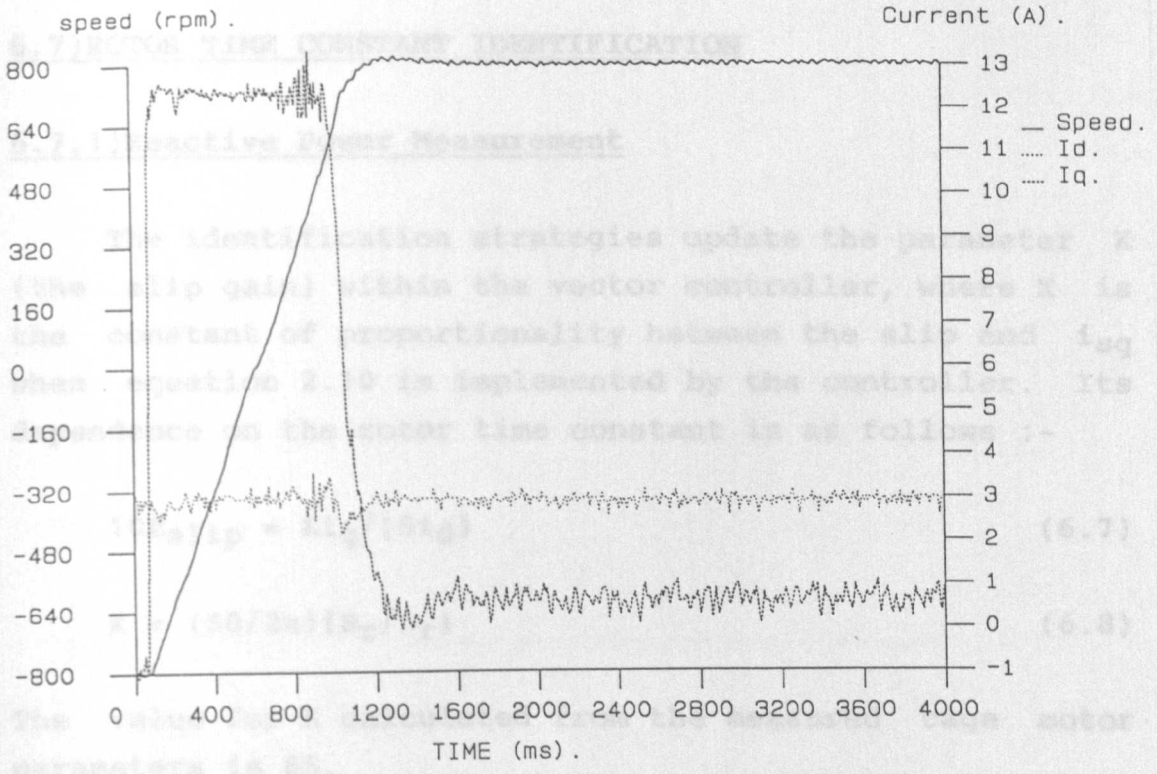


Figure 6.30 Speed Transient for the Unloaded Wound Motor with a 12A Torque Current Limit with VI Type Control

6.7)ROTOR TIME CONSTANT IDENTIFICATION

6.7.1)Reactive Power Measurement

The identification strategies update the parameter K (the slip gain) within the vector controller, where K is the constant of proportionality between the slip and i_{sq} when equation 2.10 is implemented by the controller. Its dependence on the rotor time constant is as follows :-

$$10f_{slip} = Ki_q/(5i_d) \qquad (6.7)$$

$$K = (50/2\pi)(R_r/L_r) \qquad (6.8)$$

The value for K calculated from the measured cage motor parameters is 65.

Figure 6.31 shows a speed transient for the unloaded cage motor with a torque current limit of 9A when the value of K is "detuned" from the actual motor T_r. In this case K = 100. It can be seen that the discrepancy in K from the measured value leads to a slight degradation of the acceleration curve, but because the load is so small, steady state error of the torque current is difficult to perceive. Figure 6.32 illustrates a steady state implementation of the reactive power identification strategy with K initially 100. The expected final value is k = 65, and the trace shows that the routine hones in on a value within 10% of this within 500ms. Figure 6.33 shows a transient taken immediately after identification and the improvement of the acceleration curve can be seen in a reduction of 30 ms for the transient time. The average acceleration of Fig. 6.33. is very similar to that of Fig. 6.22. which is good evidence of the identification strategy working effectively. Again it was noticed that for a few post transient identification implementations (about 5%) some discrepancy occurred, in this case being the calculation of a value of K different to that

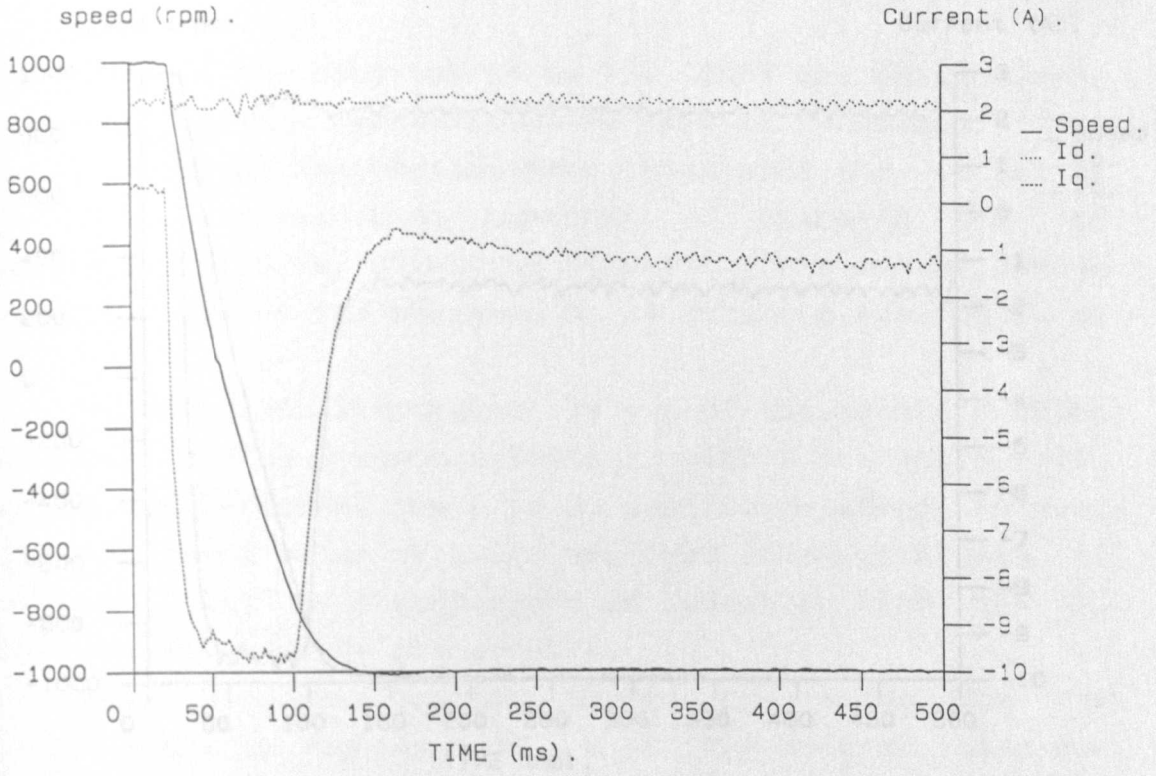


Figure 6.31 Speed Transient for the "Detuned" Unloaded Cage Motor (K = 100)

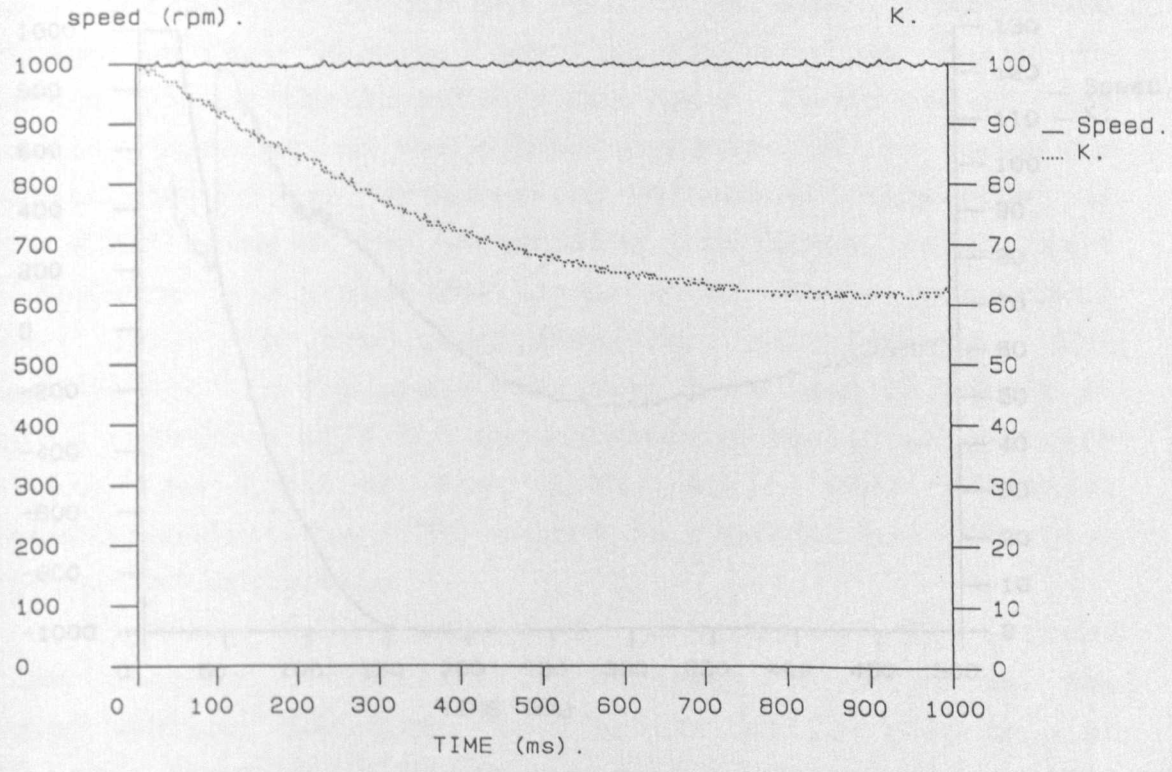


Figure 6.32 Steady State Tr Identification using Reactive Power Measurement (K = 100 Initially)

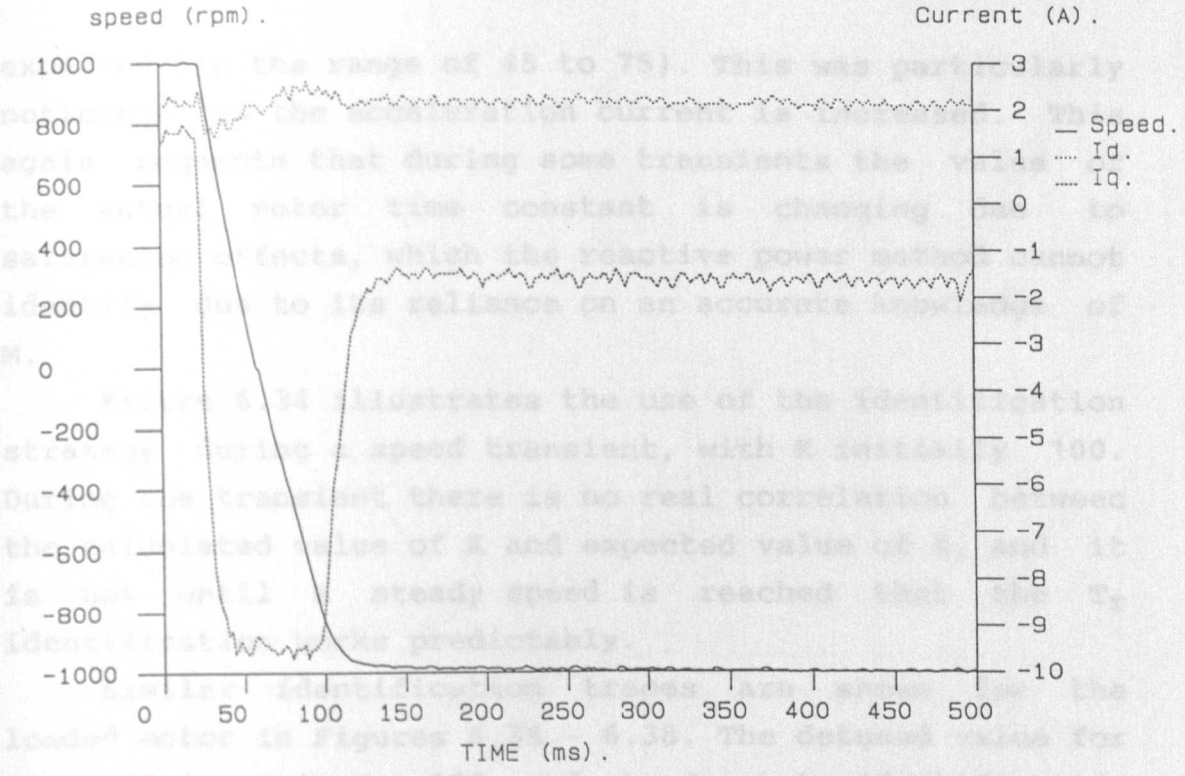


Figure 6.33 Speed Transient for the Unloaded Cage Motor After Steady State Tr Identification

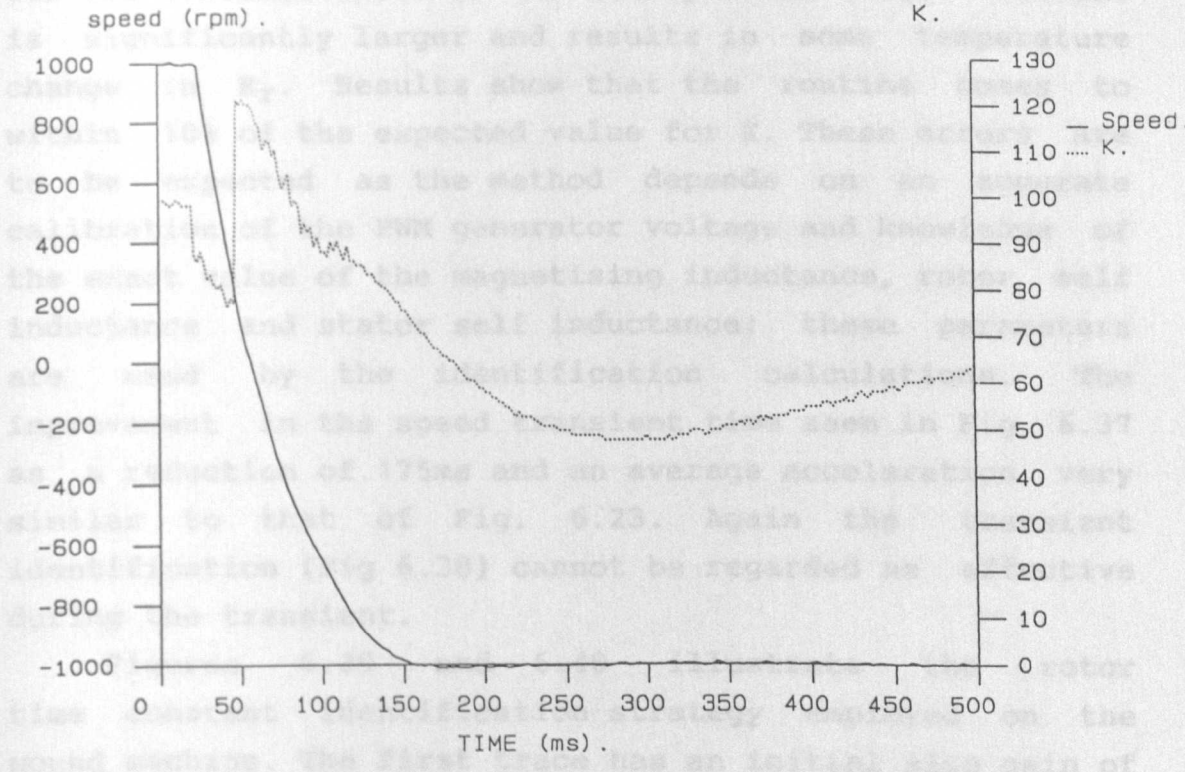


Figure 6.34 Transient Tr Identification for the Unloaded Cage Motor (K = 100 Initially)

expected (in the range of 45 to 75). This was particularly noticable as the acceleration current is increased. This again suggests that during some transients the value of the actual rotor time constant is changing due to saturation effects, which the reactive power method cannot identify due to its reliance on an accurate knowledge of M.

Figure 6.34 illustrates the use of the identification strategy during a speed transient, with K initially 100. During the transient there is no real correlation between the calculated value of K and expected value of K, and it is not until a steady speed is reached that the T_r identification works predictably.

Similar identification traces are shown for the loaded motor in Figures 6.35 - 6.38. The detuned value for Fig 6.35 is again K = 100, and steady state identification gives a correction to K = 62 (Fig 6.36). This value is expected to be slightly different from the value found for the unloaded motor as the steady state torque current is significantly larger and results in some temperature change in R_r. Results show that the routine comes to within 10% of the expected value for K. These errors are to be expected as the method depends on an accurate calibration of the PWM generator voltage and knowledge of the exact value of the magnetising inductance, rotor self inductance and stator self inductance; these parameters are used by the identification calculations. The improvement in the speed transient time seen in Fig 6.37 as a reduction of 175ms and an average acceleration very similar to that of Fig. 6.23. Again the transient identification (Fig 6.38) cannot be regarded as effective during the transient.

Figures 6.39 and 6.40 illustrate the rotor time constant identification strategy employed on the wound machine. The first trace has an initial slip gain of 35 and hones in on the measured slip gain of 110 within 500ms. The second trace starts with K = 160 and identifies

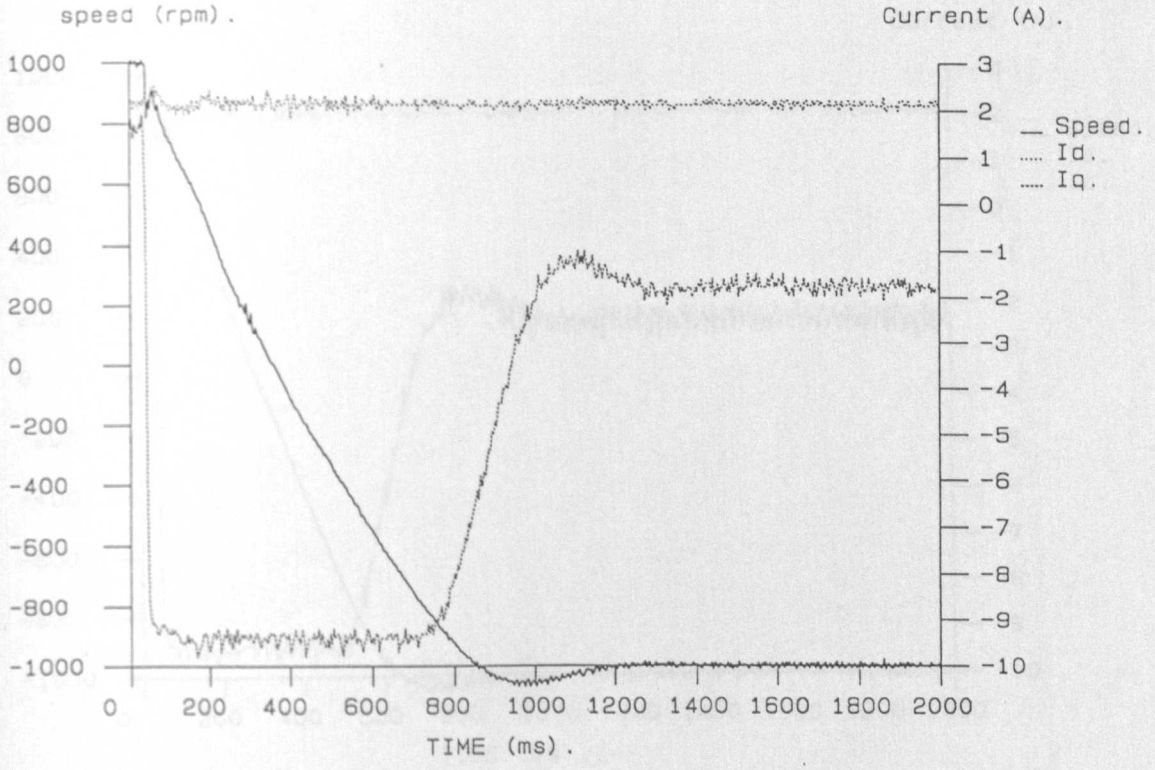


Figure 6.35 Speed Transient for the "Detuned" loaded Cage Motor (K = 100)

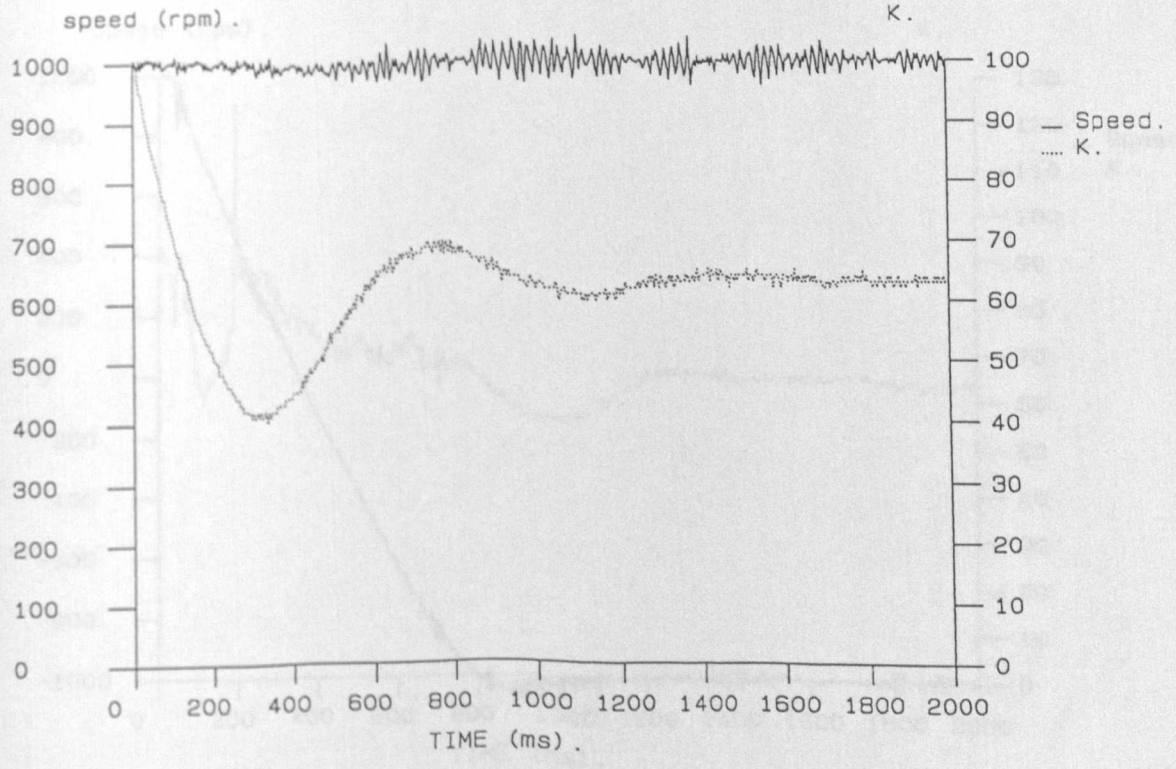


Figure 6.36 Steady State Tr Identification using Reactive Power Measurement (K = 100 Initially)

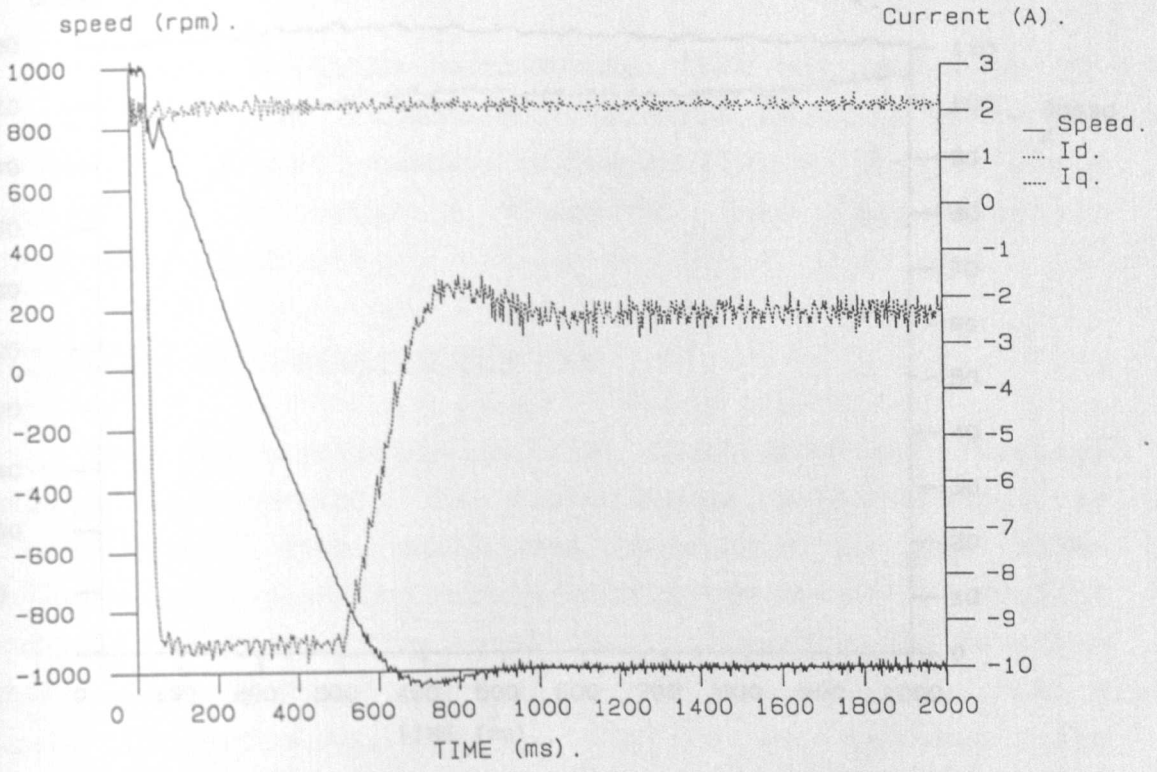


Figure 6.37 Speed Transient for the loaded Cage Motor After Steady State Tr Identification

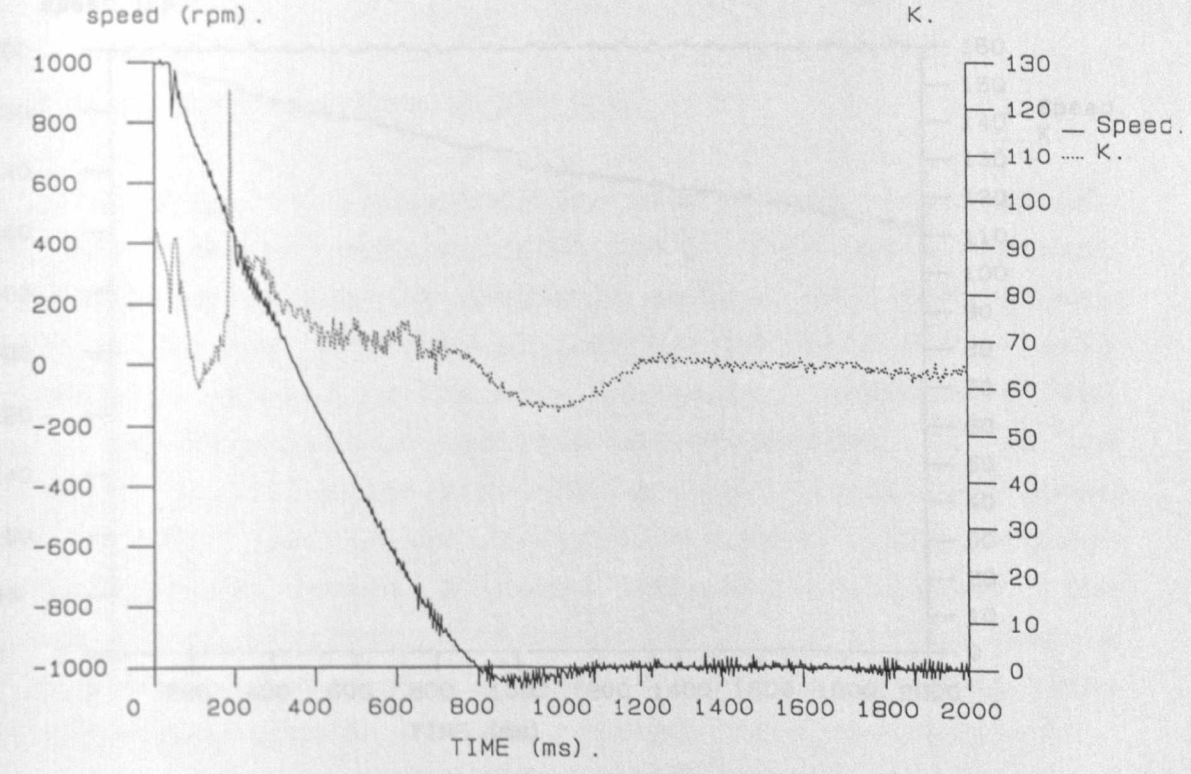


Figure 6.38 Transient Tr Identification for the loaded Cage Motor (K = 100 Initially)

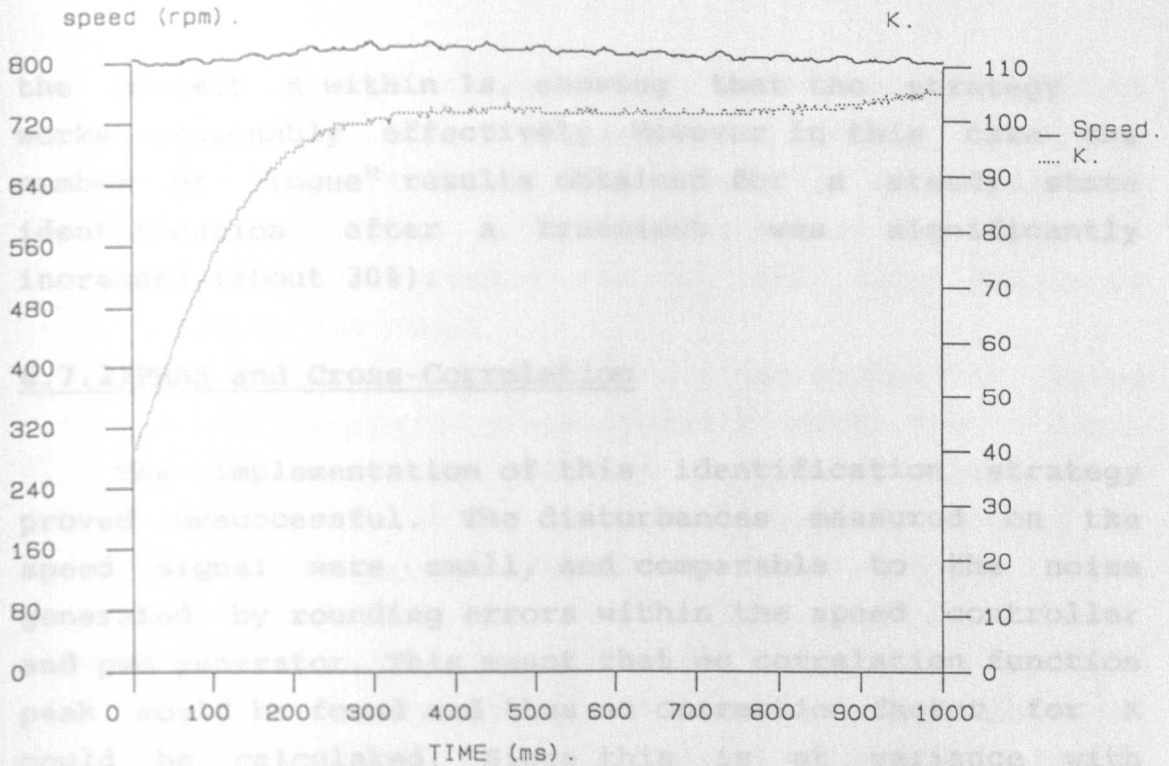


Figure 6.39 Steady State Tr Identification for the Wound Motor using Reactive Power Measurement (K = 35 Initially)

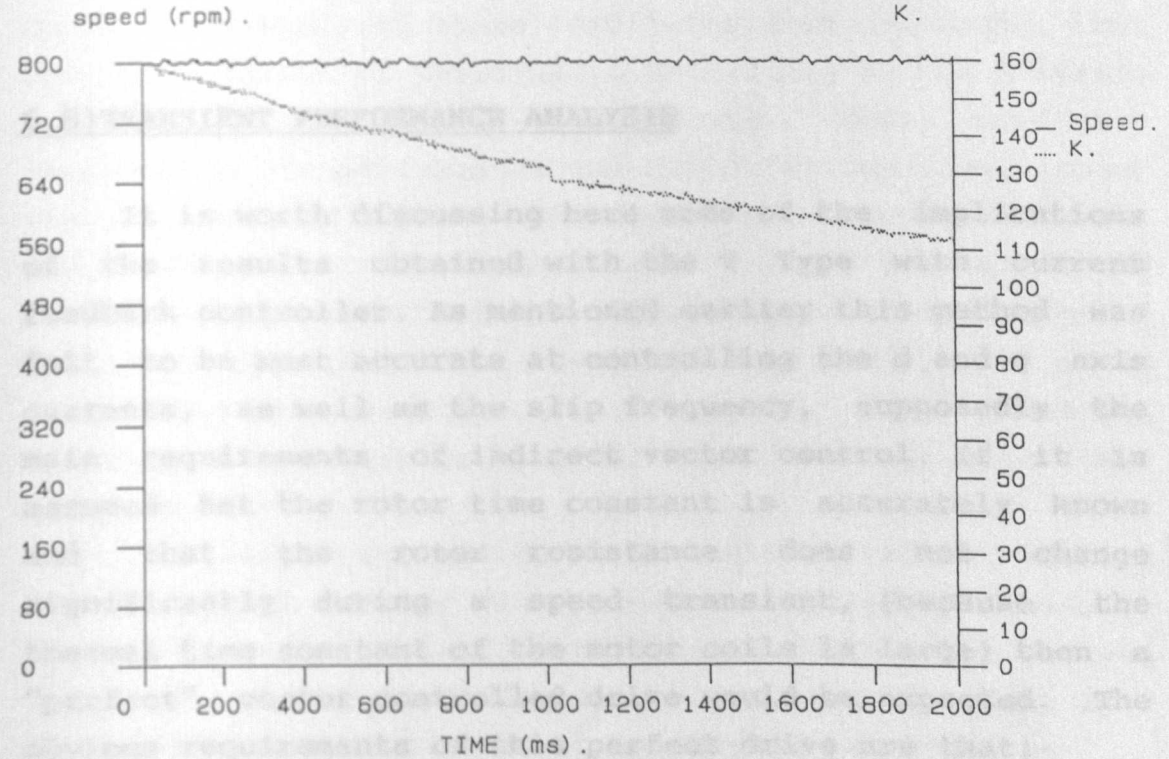


Figure 6.40 Steady State Tr Identification for the Wound Motor using Reactive Power Measurement (K = 160 Initially)

the correct K within 1s, showing that the strategy works reasonably effectively. However in this case the number of "rogue" results obtained for a steady state identification after a transient was significantly increased (about 30%).

6.7.2)PRBS and Cross-Correlation

The implementation of this identification strategy proved unsuccessful. The disturbances measured on the speed signal were small, and comparable to the noise generated by rounding errors within the speed controller and pwm generator. This meant that no correlation function peak could be found and thus no correction factor for K could be calculated. Since this is at variance with published results more work is necessary here. It is thought for example that the PRBS magnitude injected onto the d axis current should be increased in order to increase the magnitude of the disturbances.

6.8)TRANSIENT PERFORMANCE ANALYSIS

It is worth discussing here some of the implications of the results obtained with the V Type with current feedback controller. As mentioned earlier this method was felt to be most accurate at controlling the d and q axis currents, as well as the slip frequency, supposedly the main requirements of indirect vector control. If it is assumed hat the rotor time constant is accurately known and that the rotor resistance does not change significantly during a speed transient, (because the thermal time constant of the motor coils is large) then a "perfect" vector controlled drive would be expected. The obvious requirements of this perfect drive are that:-

a)For a given rotor flux magnitude and torque limit successive transients imposed on the controller

should prove consistent ie. straight line acceleration curves (assuming a fairly constant load torque), and same initial rate of acceleration.

b) After transients have been imposed as in a), the steady state torque current and slip frequency should be the same.

c) The acceleration rate for a given change in speed should increase proportionally with the torque current limit set at the output of the speed controller. In theory this should only be limited by the current capabilities of the motor windings or the inverter used.

d) If a steady state identification strategy is employed after a transient there should be no large change in the calculated controller slip gain. A small change may be expected due to thermal changes in the rotor resistance during the transient.

Results taken for the cage motor show that it is close to satisfying these conditions when unloaded, but some degradation of performance as defined by the criteria (a) - (d) is observed when loaded. More noticable degradation in performance was noted for the wound rotor machine. It has be shown that (c) does not occur for the wound rotor machine, and that (a), (b), and (d) only occur for 70% of the results taken. Operating conditions after a transient do vary, and when this does occur, if a T_r identification strategy is employed, the resultant calculation of slip gain varies significantly with that expected.

The basic requirement of an indirect vector control scheme is that a rotor flux vector be imposed of constant magnitude, at an angle Θ_e with respect to the stator fixed axis. Its magnitude is controlled according to eqn. 2.11 which for a constant value of Φ_{rd} can be rewritten as

$$\Phi_{rd} = Mi_d \qquad (6.9)$$

The rotor flux vector can also be considered in terms of the actual stator and rotor current vectors, as given in Appendix B, such that :-

$$\underline{\Phi}_{rd} = \underline{\Phi}_R = \underline{\Phi}_r + \underline{\Phi}_{rs} \qquad (6.10)$$

where Φ_r is the total flux linking the rotor "coil" due to the rotor currents only, and Φ_{rs} is the flux linking the rotor due to the stator currents. This can be rewritten as :-

$$\underline{\Phi}_R = L_r\underline{i}_r + M\underline{i}_s e^{-j\varepsilon} \qquad (6.11)$$

where ε is the angle between the stator reference current axis and the rotor reference current axis. It can be seen that the action of the vector controller will inevitably increase $\underline{i}_r$ and $\underline{i}_s$ to high values during a transient, and thus the magnitude of Φ_R is controlled by the angle ε by vector addition. Thus under controlled conditions the rotor flux magnitude will not exceed its preset value and the motor will not go further into saturation. However, should there be any errors introduced into the flux angle imposed on the motor, due to say a slight inaccuracy in K, compounded by poor speed (and thus angular) resolution at low speeds, then flux vector misalignment will occur and the high value of $\underline{i}_s$ will attempt to increase or reduce the rotor flux magnitude, depending on the sign of the angular error. This is illustrated in Figures 6.41a. and 6.41b. Note that the rotor time constant for the two machines is 124ms (cage motor) and 111ms (wound motor). Some variation in the flux magnitude due to detuning under high $\underline{i}_s$ is thus very possible given the time duration of the high torque current. For the case of increasing flux, the imposition of a large $\underline{i}_s$ will result in the iron going further into saturation causing a reduction in the values of L_s, L_r and M. The rotor time constant will thus change. Since motors are generally operated with the iron in near saturation, it also follows that a reduction in flux will

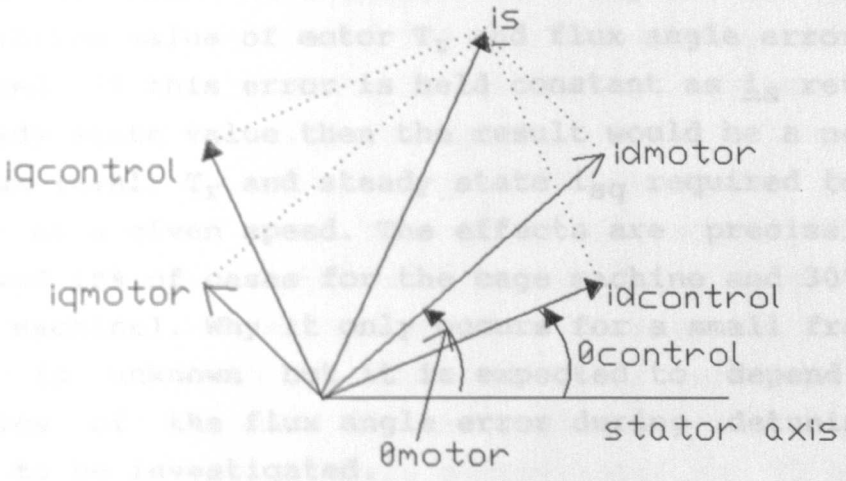


Figure 6.41a. Increase in Rotor Flux due to Detuning.

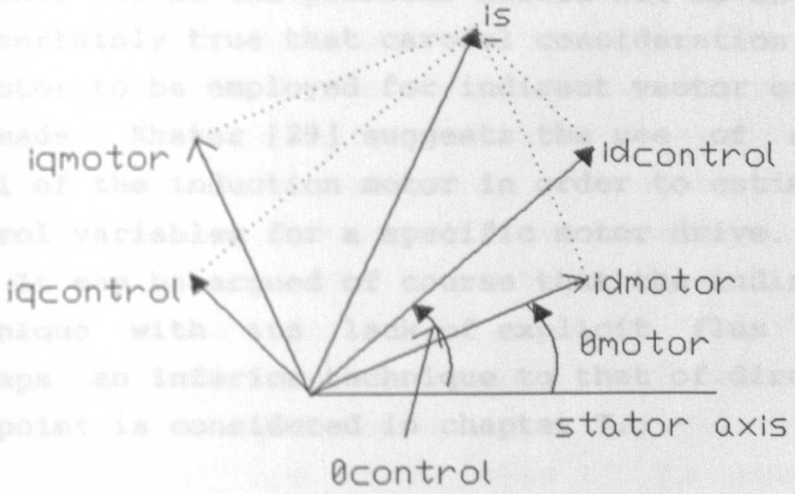


Figure 6.41b. Decrease in Rotor Flux due to Detuning

result in an increase in L_r and T_r. This discrepancy between the motor T_r and the controller T_r will further affect the error in the rotor flux angle. Eventually an equilibrium value of motor T_r and flux angle error will be obtained. If this error is held constant as $\underline{i}_s$ returns to a steady state value then the result would be a net change in flux level, T_r and steady state i_{sq} required to run the motor at a given speed. The effects are precisely those observed (5% of cases for the cage machine and 30% for the wound machine). Why it only occurs for a small fraction of cases is unknown but it is expected to depend on the dynamics of the flux angle error during detuning. This needs to be investigated.

This hypothesis seems to be supported by the different incidence of discrepancy for this phenomena for the cage and wound machine. The rated current of the wound machine is nearly three times the value of the magnetising current used, and the effect of $\underline{i}_s$ causing iron saturation saturation problems would certainly be expected. The rated current of the cage machine is only twice the magnetising current, and so the problems should not be so evident. It is certainly true that careful consideration of the type of motor to be employed for indirect vector control should be made. Khater [29] suggests the use of a saturation model of the induction motor in order to estimate the best control variables for a specific motor drive.

It may be argued of course that the indirect control technique with its lack of explicit flux control, is perhaps an inferior technique to that of direct control. The point is considered in chapter 7.

6.9)TRANSPUTER UTILISATION

The performance of the parallel implementation of the vector control strategies may be assessed by the α and β parameters (introduced in chapter 3) for the processes

running on T2 and T3. The assessment is complicated by the fact that there are two timing loops, 250μs and 2ms corresponding to the sample times of the fast current loops and the slower speed loop respectively. Within the philosophy described the latter timing loop can only be obtained by activating the speed processing loop every 8 cycles of the t_s = 250μs loop. Furthermore the procedures executed for the speed loop processing can be farmed out to successive 250μs cycles. Thus the times t_p and t_t vary according to whether the speed processing is active and according to which speed processing procedure is active. In Table 1 of Figure 6.42. the best and worst case values of α and ß are given (where appropriate) for T2 and T3 for each implementation.

The results of Table 1 are rather enlightening albeit in a rather negative sense. The problem derives from the ratio of α/ß as shown in Table 1. This ratio represents the percentage work load taken up in communicating data from one process to another. The absolute values of α and ß are low, and indicate that a faster process sample time could be employed by the transputer network.

For the V Type control strategy, T2, for example spends most of its time merely communicating data (α/ß = 100%; row 2) and acting as a buffer between T1, T3 and sampling external signals. Only when speed processing and deriving the vector control frequency demand does T2 begin to do "real" work. For the V Type control with current feedback, α and ß for T2 falls to 48% (for the non speed processing 250μs loop) on account of T2 being responsible for the i_d and i_q feedback loops. For this strategy, the best ratio of α/ß for T2, when T2 is processing both current loops, the speed loop, the vector control and the 3-2 phase current transformations, is 55%. This illustrates the performance of a microcomputer with high arithmetic speed and slow communication time.

IMPLEMENTATION	TRANSPUTER PROCESS	WITH/WITHOUT SPEED PROCESSING	α (%)		ß (%)		α/ß (%)	
			BEST	WORST	BEST	WORST	BEST	WORST
V TYPE CONTROL	T3	-	30		48		63	
	T2	WITHOUT	10		10		100	
	T2	WITH	16	26	28	26	62	100
V TYPE WITH CURRENT CONTROL	T3	-	30		48		63	
	T2	WITHOUT	16		33		48	
	T2	WITH	22	32	40	49	55	65
I TYPE CONTROL	T3	-	42		80		53	
	T2	WITHOUT	10		14		72	
	T2	WITH	16	26	25	30	64	87

Figure 6.42. TABLE 1.

Parallel Implementation Performance Factors (per process)

α: communication time/sample time

ß: active process time/sample time

CHAPTER 7

FURTHER DISCUSSION AND CONCLUSIONS

7.1)INDIRECT VECTOR CONTROL

7.1.1)V-Type Control

V Type control has been seen to give a fairly good transient speed control, although the current control was rather crude. It is obvious that current control without current feedback will be somewhat deficient due to the slow sample time employed, and the accuracy of the machine parameters used by the controller. However if the maximum current demanded by the controller is set to well within the ratings of the inverter and motor, then this is a means for getting reasonable transient performance from the drive with only a speed feedback signal.

The transputer implementation performance data indicates that this form of control (both vector control calculations and PWM calculations) could be implemented on a single transputer or high performance sequential processor, albeit with very limited monitoring capabilities. This may be of interest to drive manufacturers.

7.1.2)I-Type Control

The main problems arising from this type of control arise from the fairly low switching frequency of the 4kHz bang-bang controller, which results in a long inverter delay. Lead-Lag and D-Q axis compensation do have an effect on the current control of this strategy, and it is felt that the fast current loop should be implemented in analogue (downstream of the transputer) in order to

increase accuracy of the current control and minimise the inverter delay. If this were the case, then if a reasonably high performance drive is required with no T_r identification, it is felt that this system could also be implemented on a single processor.

7.1.3)V-Type Control with Current Feedback

This form of indirect vector control appears to provide the best foundation for an all digitally controlled high performance drive. The results indicate that current and speed control are carried out effectively using the pre-measured values for T_r and current loop design. The use of two processors is mandatory for such a controller. However if a PWM carrier frequency of 4kHz could be employed the control and actuation routines would make optimum use of the processing power of the transputer network.

7.1.4)Rotor Time Constant Identification

The need for rotor time constant identification has been demonstrated, and initial results indicate that the Reactive Power Measurement Strategy certainly improves the performance of the drive. It is felt however that under the experimental conditions used, the variation of the rotor time constant that tends to cause detuning result from flux change and hence change of the mutual inductance rather that variations in the rotor resistance due to temperature. This should make the method of identification somewhat invalid due to its dependence on accurate pre-measured values of L_s, L_r, and M. The exact nature of the problem of flux variation during a speed transient - its cause and effect, and stability - is still a subject of research.

It is felt that until the problem of rotor time constant detuning is fully understood and compensated for,

direct vector control is a superior technique for obtaining a high performance induction motor drive. Whether the increase in performance justifies the machine modification required is a question only answered by a particular drive specification.

7.2) FUTURE WORK

The ease of alteration to particular parts of the vector control scheme allows the investigation of various implementations of the vector control algorithm and actuation routines. Work in this area may include the use of modern control theory for the current control loops as described for example by Lorenz [20], Taufiq [30],[31], and Kumamoto [32]. It may also include the assessment of the other PWM strategies such as synchronous PWM, the Five Mode PWM control suggested by Nordby [27] and the recently proposed voltage vector control strategy [33]. The implementation of an analogue bang-bang controller for further appraisal of the I Type control strategy is also under consideration.

The control network constructed here provides the processing power and the potential for further study into the area of on line global and rotor parameter adaption and the influence of motor saturation. The network not only provides the computational facilities but is also able to take in more signal measurements (DC Link voltage, rotor currents and direct rotor flux measurement if required) without modification. The possible application and evaluation of Kalman Filtering techniques for optimal estimation [34],[35], would require the "number crunching" capability of the transputer network. Such a rig would truly be able to assess the effectiveness of T_r identification schemes by measuring and comparing the controller and machine rotor flux vectors.

The drive rig constructed also provides the ability

to appraise Self-Commissioning of induction motor drives, as proposed by Schierling [36],[37]. The capability of taking an "off the shelf" motor and connecting it directly to a vector controller without the need for extensive machine tests to determine the machine parameters would be of particular interest to drive manufacturers.

7.3)THE TRANSPUTER AND PARALLEL PROCESSING FOR MOTOR DRIVES

The functions of a high performance field orientated drive controller exhibit a natural parallism. In particular a three way parallelism exists between control, supervisor/communication, and background computing functions. The latter is likely to become increasingly more important in future as a single drive package becomes more capable of optimised performance over a wide range of user applications. It is worth noting that at least one manufacturer has utilised the parallelism between the control and supervisory functions using two processors. This has led to the capability of remote commissioning and monitoring and it is likely that this too will be a growth area.

The effectiveness of the transputer for real time control applications has been demonstrated. The transputer system has significant advantages, particularly if modularity and ease of modification to meet multifarious specifications are the main criteria. It is for example ideal for use in drive research rigs for evaluating algoriths over the whole spectrum of drive controller functions. Although hardware development and board costs are low, the transputers are not low cost devices at the moment and for commercial drives, their application would be expected to be restricted to the high power range. For low and medium performance it is envisaged that a hardware specific parallelism involving conventional processors

with a defined, restrictive specification is likely to remain the most economic option for some time. In conclusion it can be said that the transputer behaves as a powerful computer with a slow communication time: this results in fast computation with a lot of communication which is not particularly elegant. The use of T800 transputers with their associated faster communication rate (20 Mbits/s resulting in t_c less than 5µs) will improve this. On the positive side the ease of high level language programming, the high degree of on line user interaction whilst the drive is operational, the absence of interrupt processing and the simplicity of process synchronisation, and the ease and efficiency with which fundamentally different schemes can be implemented with no change in hardware must be emphasised. If flexibility of the controller hardware is the principle criteria for a motor drive research rig then the selection of a transputer system is eminently justified.

REFERENCES

[1] LIPO, T.A.: "Recent Progress in the Development of Solid- State AC Motor Drives", IEEE Trans. Power Electronics, vol 3, April 1988, pp 105 - 117.

[2] FINNEY, D.,: "Variable Frequency AC Motor Drive Systems.", Peter Peregrinus Ltd., London, 1988.

[3] BOSE, B.K.: "Power Electronics and AC Drives", Prentice Hall New Jersey, 1986.

[4] KUME, T. and IWAKANE, T.: "High-Performance Vector-Controlled AC Motor Drives: Applications and New Technologies", IEEE Trans. Ind. Appl., vol 1A-23, September/October 1987, pp 872 - 880.

[5] BLASCHKE, F.: "The Principle of Field Orientation as Applied to the New TRANSVEKTOR Closed Loop Control System for Rotating Field Machines", Siemens Review, 1972, pp 217.

[6] LEONHARD, W.: "Control of Electrical Drives", Springer Verlag, 1985.

[7] LEONHARD, W.: "Field-Orientation for Controlling AC Machines - Principle and Application", IEE PEVD Conference, London, July 1988, pp 277 - 285.

[8] GABRIEL, R., LEONHARD, W., and NORDBY, C.J.: "Field-Oriented Control of a Standard AC Motor Using Microprocessors", IEEE Trans. Ind. Appl., vol 1A-16, March/April 1980, pp 186 - 192.

[9] HARASHIMA, F., KONDO, S., OHNISHI, K., KAJITA, M., and SUSONO, M.: "Multimicroprocessor-Based Control System for Quick Response Induction Motor Drive", IEEE Trans. Ind. Appl., vol 1A-21, May/June 1985, pp 603 - 609.

[10] KUBO, K., WATANABE, M., OHMAE, T., and KAMIYAMA, K.: " A Fully Digitalized Speed Regulator using Multimicroprocessor System for Induction Motor Drives", IEEE Trans. Ind. Appl., vol 1A-21, July/August 1985, pp 1001 - 1007.

[11] GARCES, L.J.: "Parameter Adaption for the Speed-Controlled Static AC Drive with a Squirrel-Cage Induction Motor",IEEE Trans. Ind. Appl., vol 1A-16, March/April 1980, pp 173 - 178.

[12] MATSUO, T., and LIPO, T.A.: "A Rotor Parameter Identification Scheme for Vector-Controlled Induction Motor Drives", IEEE Trans. Ind. Appl., vol 1A-21, May/June 1985, pp 624 - 632.

[13] KOYAMA, M., YANO, M., KAMIYAMA, I., and YANO, S.: "Microprocessor-Based Vector Control System for Induction Motor Drives with Rotor Time Constant Identification Function", IEEE Trans. Ind. Appl., vol 1A-22, May/June 1986, pp 453 - 459.

[14] HOULDSWORTH, J.A. and ROSINK, W.B.: "Introduction to PWM Speed Control System for 3 Phase AC Motors", Electronic Components and Applications, Vol. 2, No. 2, February 1980.

[15] BROD, D.M.,and NOVOTNY, D.W.,: "Current Control of VSI-PWM Inverters.", IEEE Trans. Ind. Appl., vol 1A-21, May/June 1985, pp 562 - 570.

[16] KRISHNAN, R., and DORAN, F.C.: "Study of Parameter Sensitivity in High Performance Inverter-Fed Induction Motor Drive Systems ", IEEE Trans. Ind. Appl., vol 1A - 23, July/August 1987, pp 623 - 635.

[17] HO, E.Y.Y., and SEN, P.C.: "Decoupling Control of Induction Motor Drives", IEEE Trans. Ind. Electronics, vol 35, May 1988, pp 253 - 262.

[18] SUGIMOTO, H., and TAMAI, S.: "Secondary Resistance Identification of an Induction-Motor Applied Model Reference Adaptive System and Its Characteristics", IEEE Trans. Ind. Appl., vol 1A-23, March/April 1987, pp 296 - 303

[19] KRISHNAN, R., and DORAN, F.C.: "A Method of Sensing Line Voltages for Parameter Adaption of Inverter-Fed Induction Motor Servo Drives", IEEE Trans. Ind. Appl., vol 1A-23, July/August 1987, pp 617 - 622.

[20] LORENZ, R.D., and LAWSON, D.B.: "Performance of Feedforward Current Regulators for Field-Oriented Induction Machine Controllers", IEEE Trans. Ind. Appl., vol 1A-23, July/August 1987, pp 597 - 601.

[21] "IMS C011 Link Adapter", INMOS Data Sheet.

[22] "8253 Programmable Interval Timer", INTEL Data Sheet, INTEL Corperation, 1983.

[23] HOROWITZ, P. and HILL, W.: "The Art of Electronics", Cambridge University Press, 1989.

[24] BOWES, S.R., and MOUNT, M.J.: "Microprocessor Control of PWM Inverters", IEE Proc., vol 128, Pt. B, No. 6, November 1981, pp 293 - 305.

[25] BOWES, S.R., and DAVIES, T.: "Microprocessor-Based Development System for PWM Variable-Speed Drives", IEE Proc. vol 132, Pt. B, No. 1, January 1985, pp 18 - 45.

[26] BOSE, B.K., and SUTHERLAND, H.A.: "A High-Performance Pulsewidth Modulator for an Inverter-Fed Drive System Using a Microcomputer", IEEE Trans. Ind. Appl., vol 1A-19, March/April 1983, pp 235 - 243.

[27] ZUBEK, J., ABBONOANTI, A., and NORDBY, C.J.: "Pulsewidth Modulated Inverter Motor Drives with Improved Modulation.", IEEE Trans. Ind. Appl., vol 1A -11, November/December 1975, pp 695 - 703.

[28] "SIMBOL User Manual", Cambridge Control Publication, Cambridge Control Ltd., Cambridge, CB3 OEl, UK, 1987.

[29] KHATER, F.M.H., LORENZ, R.D., NOVOTNY, D.W., and TANG, K.: "Selection of Flux Level in Field-Oriented Induction Machine Controllers with Consideration of Magnetic saturation Effects", IEEE Trans. Ind. Appl., vol 1A- 23, March/April 1987, pp 276 - 281.

[30] GARCIA-CERRADA, A., and TAUFIQ, J.,: "Convergence of Rotor Flux Estimation in Field Orientated Control.", IEE PEVD Conference, London, July 1988, pp 283 - 286.

[31] TAUFIQ, J., and GARCIA-CERRADA, A.,: "State Feedback Control of Induction Motor Drives.", IEE Colloquium on Control of Induction Motors, London, April 1989.

[32] KUMAMOTO, A., and HIRANE, Y.,: "An Experimental Implementation of a Variable Speed Induction Motor Drive System Based on Modern Adaptive Control Theory.", EPEC, Grenoble, France, Sept. 1987, pp 889 - 894.

[33] VAN DER BROECK, H.W., SKUDELNY, H.C., and STANKE, G.V.,: "Analysis and Realisation of a Pulse Width Modulator Based on Voltage Space Vectors.", IEEE Trans. Ind. Appl., Vol 24, Jan/Feb 1988, pp 142 -149.

[34] ATKINSON, D.J., and ARCARNLEY, P.P.,: "Parameter Identification Techniques for Induction Motor Drives.", EPEC, Aachen, West Germany, Oct. 1989, pp 307 - 312.

[35] ZAI, L.V., and LIPO, T.A.,: "An Extended Kalman Filter Approach to Rotor Time Constant Identification in PWM Induction Motor Drives.", IEEE IAS, 1987, pp 177 - 183.

[36] SCHIERLING, H.,: "Self Commissioning - A Novel Feature of Modern Inverter Fed Induction Motor Drives.",IEE PEVD Conference, London, July 1988, pp 287 - 291.

[37] JOETTEN, R., and SCHIERLING, H.,: "Adaptive and Self-Commissioning Control for a Drive with Induction Motor and Voltage Source Inverter.", EPEC , Grenoble, France, Sept. 1987, pp 883 - 889.

[38] FLEMMING, P.J. (editor): "Parallel Processing in Control: the transputer and other architectures.", Peter Peregrinus Ltd., London, 1988.

[39] "The Transputer Data Book", INMOS Publication, INMOS Ltd., Bristol, BS12, 4SQ, UK, 1989.

[40] "IMS D700 User Manual", INMOS Publication, INMOS Ltd, Bristol, BS12, 4SQ, UK.

[41] "Occam Tutorial", INMOS Publication, INMOS Ltd., Bristol, BS12, 4SQ, UK, 1989.

[42] GHEE, S.: "IMS B004 IBM PC Add In Board", INMOS Technical Note 11, INMOS Ltd, Bristol, BS12, 4SQ, UK.

[43] "The Transputer Reference Manual", INMOS Publication, INMOS Ltd., Bristol, BS12, 4SQ, UK, 1989.

[44] VADHER, A.: "IMS B006 Board", INMOS Technical Note 14, INMOS Ltd, Bristol, BS12, 4SQ, UK.

[45] SUMNER, M. and ASHER, G.M.: "PWM Induction Motor Drive using the INMOS Transputer Parallel Processor", IEEE APEC 88 Conference, New Orleans, pp 121 - 129.

[46] ASHER, G.M. and SUMNER, M.: "Real Time Motor Control using a Transputer Parallel Processor Network", EPEC, Aachen, 1989.

APPENDIX A

SPECIFICATIONS

A.1)WOUND ROTOR MACHINE

The parameters of the three phase, 50Hz, 380V, star connected, 5hp 8.6A, 930rpm slip ring induction motor are:

R_s	2Ω	R_r	2.1Ω
L_s	0.235H	L_r	0.234H
M	0.224H	σ	0.0875
Rated Torque	38 Nm.		
Moment of Inertia	$0.32kgm^2$ (coupled to DC generator)		

A.2)SQUIRREL CAGE MACHINE

The parameters of the three phase, 50Hz, 415V, delta connected, 5.5hp 8.1A, 1420rpm squirrel cage induction motor are:

R_s	5.4Ω	R_r	4.8Ω
L_s	0.600H	L_r	0.596H
M	0.585H	σ	0.043
Rated Torque	27 Nm.		
Moment of Inertia	$0.023kgm^2$		

A.3)THE PWM INVERTER

Supplied by Control Techniques plc, the transistorised PWM inverter has the following ratings :-

Input	3ph/415V +/- 6% 50-60 Hz
Power Output	22kW
Rated Current	48A
Recommended "lockout time"	20µs
Recommended transistor "minimum on time"	20µs
Recommended transistor "minimum off time"	20µs

A.4)THE SPEED ENCODER

Type	Muirhead Vactric Components Ltd. H25D-2500-P4ZC-88C30-EM18
Counts per shaft turn	10,000
Supply Requirements	5V, 175 mA
Output signals	clockwise,counter-clockwise and index
Output format	complementary pair using CMOS line driver

A.5)THE CURRENT TRANSDUCER

Type	LEM SA LT-80-P
Ratio	1 : 1000
Accuracy	1%
Linearity	0.1%
Band Width	100kHz
Supply Req.	+/- 15V

APPENDIX B

THE MATHEMATICAL MODEL OF THE INDUCTION MOTOR

The voltage across the "a" phase stator coil of the induction motor can be written as :-

$$v_a = R_s i_a + d(\Phi_a)/dt \qquad (B.1)$$

where Φ_a is the total flux linked to the coil. Similar equations can be written for the "b" and "c" phase coils of the motor, displaced from the "a" phase in space by 120^o and 240^o respectively. The resultant voltage vector obtained by complex addition in the Argand plane can be written as :-

$$\underline{v}_s = v_a + v_b e^{j2\pi/3} + v_c e^{j4\pi/3}. \qquad (B.2)$$

Similar vectors can be written for the stator current and stator flux linkage, and thus B.1 can be rewritten in terms of vectors as :-

$$\underline{v}_s = R_s \underline{i}_s + d(\underline{\Phi}_s)/dt \qquad (B.3)$$

It should be noted that these equations are in the stator fixed frame of reference.

The rotor voltage vector equation can be written as :-

$$0 = R_r \underline{i}_r + d(\underline{\Phi}_r)/dt \qquad (B.4)$$

where the rotor vectors are defined in the rotor reference frame, rotating with an angular velocity w_r with respect to the stator frame. If the instantaneous rotor position with respect to the stator frame is defined by the angle ε

then rotor vector quantities are transferred to the stator frame of reference by a multiplication factor $e^{j\varepsilon}$.

The stator flux vector may be rewritten as :-

$$\underline{\Phi}_s = L_s\underline{i}_s + M\underline{i}_r e^{j\varepsilon} \qquad (B.5)$$

where L_s is the stator self inductance, and M is the mutual inductance between the stator and rotor. It is assumed that the turns ratio of the stator to the rotor is unity. Similarly the rotor flux vector may be rewritten as :-

$$\underline{\Phi}_r = L_r\underline{i}_r + M\underline{i}_s e^{-j\varepsilon} \qquad (B.6)$$

where L_r is the rotor self inductance. Thus equations B.3 and B.4 can be rewritten as :-

$$\underline{v}_s = R_s\underline{i}_s + L_s d(\underline{i}_s)/dt + Md(\underline{i}_r e^{j\varepsilon})/dt \qquad (B.7)$$

$$0 = R_r\underline{i}_r + L_r d(\underline{i}_r)/dt + Md(\underline{i}_s e^{-j\varepsilon})/dt \qquad (B.8)$$

The torque generated from the interaction of the stator currents with the rotor flux vector can be written as :-

$$T_e = \text{Imag } K\{\underline{i}_s\underline{\Phi}_r^*\} \qquad (B.9)$$

$$T_e = \text{Imag } KM\{\underline{i}_s(\underline{i}_r e^{j\varepsilon *})\} \qquad (B.10)$$

where * represents the complex conjugate.

Equations B.2 and B.3 constitute a three to two phase transformation which does not conserve power. A correction factor of (2/3) must be included in the torque equation. The number of pole pairs (p) must also be included as the angle used for the derivation of the torque equation is in mechanical radians, whereas ε is measured in electrical radians. Equation B.10 thus becomes :-

$$T_e = (2/3)(Mp)\ \text{Imag}\ \{\underline{i}_s(\underline{i}_r e^{j\varepsilon *})\} \qquad (B.11)$$

This model of the induction motor can now be transformed to a synchronously rotating frame of reference whose instantaneous angle is defined as Θ_e where

$$\Theta_e = \int (w_e)\, dt \qquad (B.12)$$

Transformation of equation B.7 gives :-

$$\underline{v}_s e^{-j\Theta} = R_s \underline{i}_s e^{-j\Theta} + L_s(d(\underline{i}_s)/dt)e^{-j\Theta} + M(d(\underline{i}_r)/dt)e^{-j\Theta} \qquad (B.13)$$

or

$$\underline{v}_{se} = R_s \underline{i}_{se} + L_s d(\underline{i}_{se})/dt + jw_e L_s \underline{i}_{se} + M d(\underline{i}_{re})/dt + jw_e M \underline{i}_{re} \qquad (B.14)$$

where the subscript e denotes vectors defined in the synchronous frame of reference. The rotor currents are inaccessible quantities in a cage machine and so it is more convenient to rewrite the synchronous rotor current vector in terms of the synchronous stator current vector and the synchronous rotor flux vector i.e. :-

$$\underline{i}_{re} = \underline{\Phi}_{re}/L_r - (M/L_r)\underline{i}_{se} \qquad (B.15)$$

Equation B.14 now becomes :-

$$\underline{v}_{se} = R_s \underline{i}_{se} + \sigma L_s d(\underline{i}_{se})/dt + jw_e \sigma L_s \underline{i}_{se} + (M/L_r) d(\underline{\Phi}_{re})/dt + jw_e (M/L_r)\underline{\Phi}_{re} \qquad (B.16)$$

where σ is the leakage coefficient such that

$$\sigma L_s = (L_s L_r - M^2)/L_r \qquad (B.17)$$

Transforming the rotor dynamic equation B.8 gives :-

$$0 = -(MR_r/L_r)\underline{i}_s + (R_r/L_r)\underline{\Phi}_{re} + d(\underline{\Phi}_{re})/dt + jw_{slip}\underline{\Phi}_{re}$$

(B.18)

Equations B.17 and B.18 can be rewritten in terms of real (d axis) and imaginary (q axis) parts such that :-

$$v_{sd} = (R_s + L_s P)i_{sd} - w_e \sigma L_s i_{sq} + (PM/L_r)\Phi_{rd} - w_e(M/L_r)\Phi_{rq}$$

(B.19)

$$v_{sq} = w_e \sigma L_s i_{sd} + (R_s + \sigma L_s P)i_{sq} + w_e(M/L_r)\Phi_{rd} + (PM/L_r)\Phi_{rq}$$

(B.20)

$$0 = -(MR_r/L_r)i_{sd} + ((R_r/L_r) + P)\Phi_{rd} - w_{slip}\Phi_{rq} \qquad (B.21)$$

$$0 = -(MR_r/L_r)i_{sq} - w_{slip}\Phi_{rd} + ((R_r/L_r) + P)\Phi_{rq} \qquad (B.22)$$

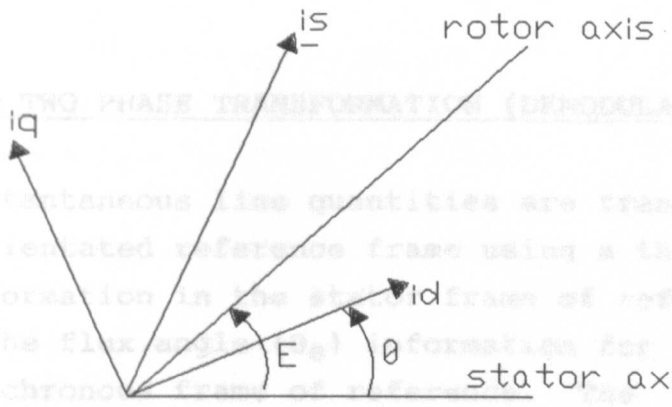
which corresponds to the 4th order differential equation set eqn. 2.3. The angular relationship of the d and q axis currents to the stator current vector is illustrated in Figure B.1.

The torque equation can be rewritten as :-

$$T_e = (2/3)pM \text{ Imag } \{\underline{i}_s e^{j\Theta}(\underline{i}_r e^{j(\Theta-\varepsilon)} e^{j\varepsilon})^*\} \qquad (B.23)$$

$$T_e = (2/3)pM \text{ Imag } \{\underline{i}_{se}(\underline{\Phi}_{re}/L_r)^* \} \qquad (B.24)$$

$$T_e = (2/3)(pM/L_r)(i_{sq}\Phi_{rd} - i_{sd}\Phi_{rq}) \qquad (B.25)$$



Angular Relationship of Current Vectors

Figure B.1.

APPENDIX C

TRANSFORMATION EQUATIONS

C.1)THREE TO TWO PHASE TRANSFORMATION (DEMODULATION)

The instantaneous line quantities are transformed to the field orientated reference frame using a three to two phase transformation in the stator frame of reference, and then using the flux angle (Θ_e) information for conversion to the synchronous frame of reference. The stator two phase quantities are denoted by the subscripts α and β. Transformation of the current vector is shown as an example.

The stator current vector is defined as :-

$$\underline{i}_s = i_a + i_b e^{j2\pi/3} + i_c e^{j4\pi/3} \qquad (C.1)$$

Three to two phase transformation is achieved using equations C.2 and C.3. :-

$$i_\alpha = (3/2)i_a \qquad (C.2)$$

$$i_\beta = ((\sqrt{3})/2)(i_b - i_c) \qquad (C.3)$$

Transformation to field co-ordinates then follows :-

$$\underline{i}_s e^{-j\Theta e} = (i_\alpha + ji_\beta)(\cos\Theta_e - j\sin\Theta_e) = i_{sd} + ji_{sq} \qquad (C.4)$$

$$i_{sd} = i_\alpha \cos\Theta_e + i_\beta \sin\Theta_e \qquad (C.4)$$

$$i_{sq} = i_\beta \cos\Theta_e - i_\alpha \sin\Theta_e \qquad (C.5)$$

C.2)TWO TO THREE PHASE TRANSFORMATION (MODULATION)

Modulation is achieved using the flux angle information as follows :-

$$\underline{i}_s = (i_\alpha + ji_\beta) = (i_{sd} + ji_{sq})e^{-j\theta e} \quad (C.6)$$

$$i_\alpha = i_{sd}\cos\theta_e - i_{sq}\sin\theta_e \quad (C.7)$$

$$i_\beta = i_{sq}\cos\theta_e + i_{sd}\sin\theta_e \quad (C.8)$$

Phase splitting is as follows :-

$$i_a = (2/3)i_\alpha \quad (C.9)$$

$$i_b = (-1/3)i_\alpha + (1/\sqrt{3})i_\beta \quad (C.10)$$

$$i_c = (-1/3)i_\alpha - (1/\sqrt{3})i_\beta \quad (C.11)$$

The coefficients in the equations above are consistent with the constant (2/3) used in the torque equations (B.23) to (B.25).

APPENDIX D

THE TRANSPUTER

D.1) INTRODUCTION

Standard computer architectures have until recently been based upon the premise that the processor is more expensive than the memory which it serves and this has resulted in the use of the "Van Neumann" architecture for system design. This principle has a single central processor surrounded by a large array of memory and peripherals, all communicating via shared buses. Performance improvement is achieved by speeding up the clock speed of the system and enhancing memory access times. However the performance which can be achieved by a single processor is now approaching a physical limit. The future of computer architecture design has been radically altered by the advent of Very Large Scale Integration (VLSI) whereby processor design is no longer a restriction and alternatives to the Von Neumann architecture can now be examined.

The two principle alternatives so far developed are the pipeline architecture found in Digital Signal Processors (DSPs) and the parallel or concurrent architecture. Both appear to offer an increase in speed, but the suitability of these architectures seems to be application dependent. The DSP achieves its speed by exploiting specific hardware for successive multiply-shift-add operations. This results in a system which is ideal for "number crunching" applications such as matrix manipulation and signal analysis, but such power is wasted on simple algorithms such as text manipulation. Parallel processing can be formally defined as "a technique for increasing the computation speed for a task by dividing

the algorithm into several sub tasks and allocating multiple processors to execute multiple sub tasks simultaneously" [38]. It can be immediately seen that the use of a parallel processing network will only increase processor speed significantly if the algorithm can be split into concurrent routines. The inherent parallelism associated with Vector control for induction motors is discussed in chapter 3, and thus the following discussion is limited to the development of parallel processing architectures using the transputer.

D.2)OCCAM AND THE TRANSPUTER

The Transputer, commercially introduced by INMOS Ltd. in 1986 has provided a significant alternative to conventional processing techniques, as it has been designed in accordance with the OCCAM model for parallel processing applications [38],[39]. It is not intended for use as a stand alone processor performing sequential tasks (although it can work effectively in this manner) but as a component part of a network of transputers where concurrency can be exploited to extend system performance and increase processing speed. Before describing the transputer device itself it is worth outlining the occam model for parallelism and describing the main features of the OCCAM parallel processing language itself.

D.2.1)THE OCCAM MODEL FOR CONCURRENCY

An excellent introduction to parallel processing architectures is provide by Flemming [38], and this section will summarise the main features of occam. The basis of this language is the modular "process model" whereby a process is defined as an independent sequential routine which performs a set of operations and then

terminates. Several processes can be defined to run simultaneously, and inter-process data transfer is via point to point links called communication channels. Parallel processes do not share access to global variables. Each process has its own local variables, and communication is solely through channels. Data transfer is synchronised so that communication takes place only when the transmitter and receiver are ready. If either process becomes ready for data transfer, then it will automatically have to wait until the other process is ready.

D.2.2)THE OCCAM LANGUAGE

A more formal definition of the language structures is provided by [40],[38],[41] and the following is intended as a summary.

Occam is a high level language, which allows for a fairly straightforward programming strategy, but compiled occam is as efficient as hand written machine code. As mentioned earlier the basis for the language is the process, the most basic of which are assignment, input and output. These operate on two basic data structures namely the variable, used to store data, and the channel, used to transfer data between processes. Assignment (represented by :=) sets the value of a variable according to an expression. The language provides the basic mathematical operators (multiply, divide, add, subtract, remainder), boolean operators (greater than, less than, equal to, AND, OR, NOT), shift operators (<<,>>) and bitwise operators (\/,/\). Real number routines, trigonometrical routines and power routines are provided as software packages. Input and output (represented by ? and ! respectively) are used for inter process communications. A process itself may be built up from other processes by using the constructs SEQ, IF, PAR and ALT.

The SEQuential construct defines that the processes that follow should be executed sequentially and is similar to programs written in conventional high level programming languages. The occam pseudo-code format for this construct is illustrated here where ... represents a process and the indentation following the constructor indicates that those processes are governed by that particular constructor.

```
SEQ
  ... process 1
  ... process 2
  ... process 3
```

The PARallel construct defines that the processes following should be executed concurrently (with communication channels having been previously defined), as illustrated :-

```
PAR
  ... process 1
  ... process 2
  ... process 3
```

Note that processes 1, 2 and 3 will invariable be constructs themselves, but are treated as independent processes by this construct.

The IF construct is the conditional construct which determines which process to execute on the basis of a boolean guard. Again this construct is already employed in conventional high level languages. This construct is shown here :-

```
IF
  x = 1
    ... process 1
  x = 2
    ... process 2
```

```
    TRUE
      ... process 3
```

Note that the "TRUE" statement employed here provides a process to be executed should none of the guards be TRUE.

The **ALT**ernative construct is illustrated :-

```
ALT
  channel.1.input ? x
    ... process 1
  channel.2.input ? x
    ... process 2
  channel.3.input ? x
    ... process 3
```

This construct uses the first input from several channels to become ready to decide which of its component processes to execute. Once a channel has become ready for data transfer then only its associated process is performed. No equivalent construct is available for output.

Priority can be assigned to PAR and ALT constructs. If the PAR statement were replaced by a PRI PAR statement then this would assign a high priority to process 1 and low priority to processes 2 and 3. The uses and implications of this are explained later. Similarly if ALT were replaced by PRI ALT then if all the inputs to the three channels became available simultaneously, then process 1 would always be executed. Previously the determination of which process to execute under those conditions was undefined.

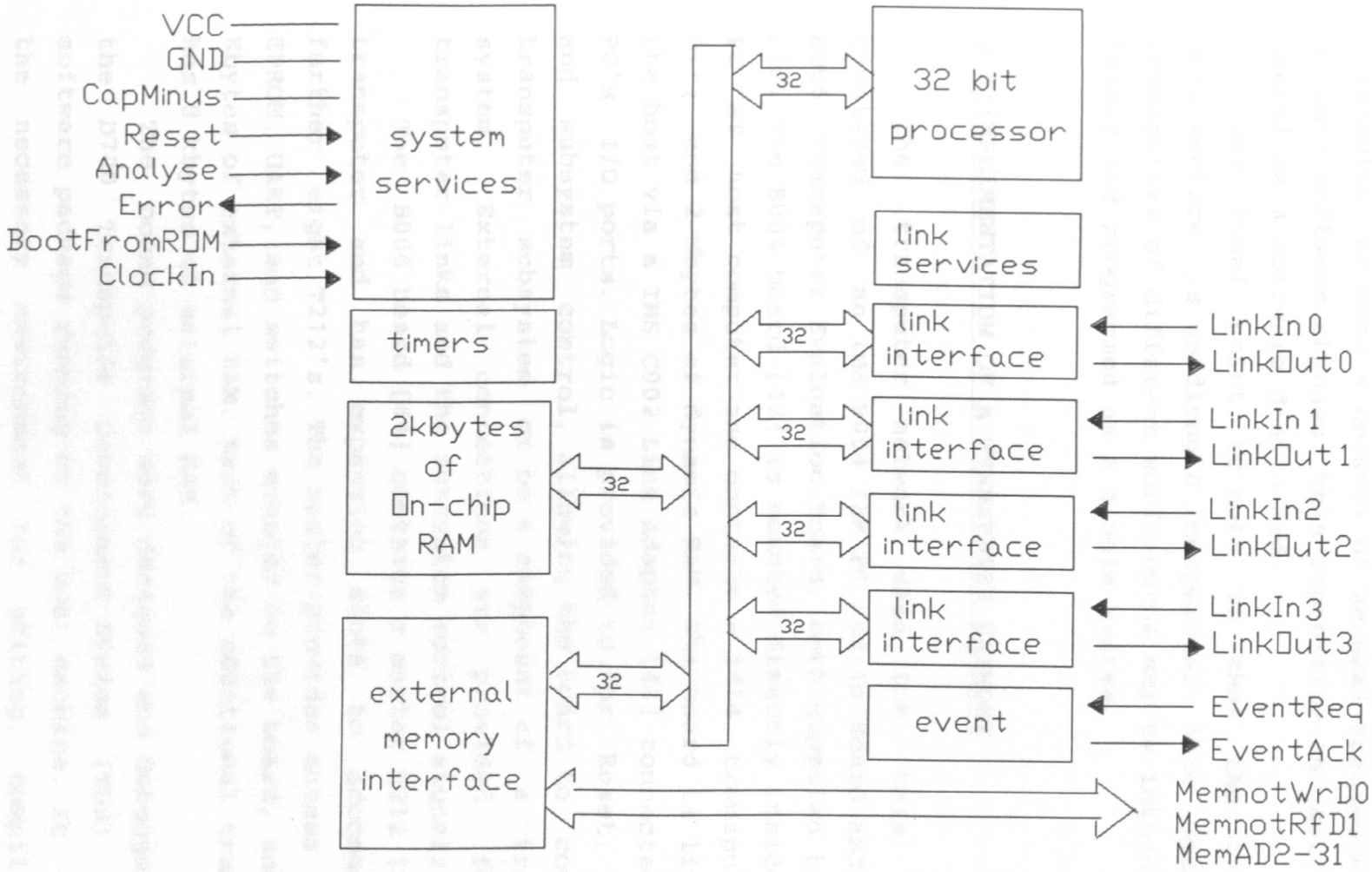
One further feature of the occam language and its implementation on the transputer is the access to two on chip timers which can be used for internal measurement or for real time scheduling. The high priority timer can only be accessed by a high priority routine and operates with a 1 µs tick. The low priority timer can be accessed by all other processes and operates with a 64 µs tick.

D.2.3)THE TRANSPUTER

The transputer is available as either a 16 bit (T212) or 32 bit (T414) microcomputer, containing all the resources required by a computer system. A simplified block diagram of the internal structure of the T414 is shown in Fig. D.1.

The T414 [39],[42] device itself contains a medium speed microprocessor with an integer multiply time of 2.5 μs when operated at a clock frequency of 15 MHz. This processor features a Reduced Instruction Set whereby the number of instructions available is limited to about sixty in order to increase processor speed. This provides an efficient implementation of high level languages and allows the processor to operate at a rate of 10 MIPS. 2 k Bytes of on-chip RAM are provided with a nominal access time of 80 MBytes/s. A configurable memory/peripheral interface is also provide which can access up to 4 gigabytes of external memory at a nominal rate of 26.6 Mbytes/s. Four bi-directional serial communication links are provided for point to point data transfer to or from other transputers or transputer look-alike peripherals. These operate at a rate of 10 Mbits/s which results in a transfer time of 5 μs per byte. This is slow and is thus a significant factor when trying to assess the performance of the transputer network in comparison with other parallel structures. The Event pin provides a means for synchronising separate processes and is in effect an interrupt capability. System services include reset, error reporting and analysis, clockin and the power supply.

The transputer is designed so that it can implement a set of parallel processes. This is achieved using a time slicing technique based on the following laws :- a high priority process will proceed until it has to wait for a communication or time-out; once it has to wait low priority processes will be executed on a time sliced basis. The transputer is also designed such that it



IMS T414 Transputer Block Diagram

Figure D.1.

appears as a process to a surrounding network. Thus a parallel processing algorithm can be mapped onto a single transputer or onto a network of several transputers with minimal software changes. This capability is surprisingly useful as a software design aid.

One final point to note is that the transputer architecture is wordlength independent. This means that transputers of different wordlengths may be interconnected freely and programmed as a single system.

D.3) IMPLEMENTATION OF A TRANSPUTER NETWORK

The transputer network used for this project consisted of an IMS B004 IBM PC Add-In Board and a IMS B006 Transputer Evaluation Board, both supplied by INMOS Ltd. The B004 board [42] is mounted directly inside an IBM PC AT host computer and contains a T414 transputer (32 bit) and 2 Mbytes of dynamic RAM. The board is linked to the host via a IMS C002 Link Adapter [43] connected to the PC's I/O ports. Logic is provided to for Reset, Analyse and subsystem control, allowing the board to control a transputer subsystem or be a component of a transputer system. External connections are provided for the transputer links and the subsystem control signals.

The B006 board [44] contains a master T212 (16 bit) transputer and has expansion slots to accommodate a further eight T212's. The master provides access to the EPROM, UART, and switches mounted on the board, and has 56 Kbytes of external RAM. Each of the additional transputers has 8 kbytes of external RAM.

The occam programs were designed and debugged using the D700 Transputer Development System (TDS) [40], a software package running on the host machine. It provides the necessary environment for editing, compiling and configuring the occam routines to run on the transputer network. The separate processes are created and compiled

individually, and then a higher level editor is invoked to assign a particular process and communication links to each transputer or peripheral in the network. The TDS then downloads the resulting code into the corresponding system and it is executed. The TDS also provides on line access to the host's keyboard and screen, or by using programs written in most high level languages, routines can be provided to access to the disk files or graphics capabilities.

The combination of the TDS system with a B004 board provides an excellent development system for transputer applications. Particularly useful are the folding editor [40] and the ability to simulate multiprocessor networks on a single transputer. The addition of a B006 board provides a network of transputers easily interconnected and controlled by the B004 master transputer.

APPENDIX E

DIGITAL IMPLEMENTATION OF LAPLACE COMPENSATION FUNCTIONS

E.1) INTRODUCTION

The discretisation of the Laplace transfer functions employed in this project was achieved using the Bilinear Transformation given in eqn. E.1.

$$s = (2/T)((1 - z^{-1})/(1 + z^{-1})) \qquad (E.1)$$

where :-

s: is the Laplace Operator.
z^{-1}: is a unit delay operator
T: is the system sampling time.

Discretisation of the original transfer function is accomplished by substituting eqn. E.1. into the the Laplace equation and manipulating the resultant equation to give a transfer function for the current sampled signals in terms of previously sampled signals. The following definitions are required for the digitised transforms described, :-

y_k: current transfer function output.
e_k: current transfer function input.
$y(k - 1)$: previous transfer function output.
$e(k - 1)$: previous transfer function input.

E.2)FIRST ORDER LOW-PASS FILTER

A simple first order low-pass filter is given by the Laplace transfer function :-

$$y/e = w/(s + w) \qquad (E.2)$$

where w is the cut off angular velocity. Substitution of the bilinear transformation and rearranging gives :-

$$y(1 + ((w - 2/T)/(w + 2/T))z^{-1}) = e(w/(w + 2/T))(1 + z^{-1})$$

$$(E.3)$$

The resultant digital filter is thus implemented as :-

$$y_k = y_{(k - 1)}((2/T - w)/(2/T + w)) +$$

$$(e_k + e_{(k - 1)})(w/(w + 2/T) \qquad (E.4)$$

E.3)PROPORTIONAL PLUS INTEGRAL COMPENSATION

The Laplace transform for a PI controller is given as

$$y/e = K(s + w)/s \qquad (E.5)$$

Substitution of the bilinear transformation and rearranging gives :-

$$y(1 - z^{-1}) = e(KT/2)[((2/T) + w) - z^{-1}((2/T) - w)] \qquad (E.5)$$

The resultant digital PI controller is thus :-

$$y_k = y(k - 1) + (K + (wKT)/2)e_k - (K - (wKT)/2)e_{(k - 1)}$$

$$(E.6)$$

E.3)LEAD LAG COMPENSATION

The Laplace transform for a lead lag compensator is given as :-

$$y/e = (s + w_1)/(s + w_2) \qquad (E.7)$$

Substitution of the bilinear transformation and rearranging gives :-

$$y((2/T + w_2) - z^{-1}(2/T - w_2)) = e((2/T + w_1) - z^{-1}(2/T - w_1))$$

(E.8)

The resultant digitised lead lag compensator is thus :-

$$y_k = y_{(k-1)}(2/T - w_2)/(2/T + w_2) + e_k(2/T + w_1)/(2/T + w_2) - e_{(k-1)}(2/T - w_1)/(2/T + w_2)$$

(E.9)