THE TRANSPUTER CONTROL OF INDUCTION MOTOR DRIVES

by

Patrick Chi-Kwong LUK, BSc(Hons), MPhil, AMIEE

A dissertation submitted to the Council for National Academic Awards
(CNAA) in support of an application for the degree of
Doctor of Philosophy in Electronic Engineering.

Department of Electronics and Information Technology
The Polytechnic of Wales

May 1992

Declaration

This thesis has not been, or is being currently submitted for the award of any other degree or similar qualification.

Patrick Chi-Kwong, LUK

Abstract

The inherent advantages of the induction motor in variable speed drive applications can now be realised in a cost-effective manner as a result of recent advances in power electronics and microelectronics. This thesis is devoted to the advancement of the use of induction motors in variable speed applications, and describes the analysis, simulation and implementation of a variable speed induction motor drive.

The state-space method lends itself as an ideal approach both for digital computer modelling and design of modern controller and was therefore adopted for the analysis and simulation of the drive system. The simulation was developed by means of a low cost personal computer package called MATLAB that has been designed to facilitate matrix operations. The use of such a specialized software package provided a 'user-friendly' operating environment with error messages identifying problem areas during program development. The resulted computer model of the drive system offers high flexibility and modularity and can be readily incorporated into further analysis and real-time controller design. Experimental results of the drive demonstrated good correlation with the model at both steady and transient states and the validity of the model is therefore confirmed.

The experimental drive system was developed by means of transputers and its associated programming language **occam**. It was a flexible and comprehensive drive system comprising: (i) an on-line user interactive environment facilitated by the Transputer Development System; (ii) a 3-phase inverter bridge as the power conditioning unit; and (iii) a signal processing unit by means of a multi-transputer network system. The adoption of the transputer and **occam** enabled parallel processing to be achieved cost effectively in the drive system. The specifications of the drive system developed included on-line speed change, dynamic braking and programmable soft-start. Vector-control was also incorporated for good dynamic response. Experimental results of the specified functions of the drive are provided to confirm the proposed specifications of the drive.

Further research areas on the present system are proposed, so that a viable industrial implementation may be contemplated.

Acknowledgment

I am most grateful for the continuous help, discussions and interest of my Director of Studies, Dr.M.G.Jayne, and my second supervisors Dr.D.Rees and Dr.R.R.Payne, during the period of research to which this thesis relates. I am also thankful to the industrial collaboration of Black Clawson International of Newport, Gwent.

I would also like to thank the technical and secretarial staff of the Department of Electronics and Information Technology for their assistance throughout.

May I dedicate this thesis to my faraway family, especially my parents, who have been so patient and supportive throughout the period of my research in the U.K.

Last but not least is my heartfelt appreciation of the unequivocal support and expert help from my wife, Tsz Kwan, who has made the completion of the investigation and this thesis possible.

List of Principal Symbols

i_{dr} ..... *Instantaneous d-axis rotor current*
i_{ds} ..... *Instantaneous d-axis stator current*
i_{qr} ..... *Instantaneous q-axis rotor current*
i_{qs} ..... *Instantaneous q-axis stator current*
v_{dr} ..... *Instantaneous d-axis rotor voltage*
v_{ds} ..... *Instantaneous d-axis stator voltage*
v_{qr} ..... *Instantaneous q-axis rotor voltage*
v_{qs} ..... *Instantaneous q-axis stator voltage*
X_r ..... *Rotor reactance*
X_s ..... *Stator reactance*
X_{lr} ..... *Rotor leakage reactance*
X_{ls} ..... *Stator leakage reactance*
L_m ..... *Magnetizing inductance*
L_h ..... *Leakage inductance*
L_r ..... *Rotor inductance*
L_s ..... *Stator inductance*
L_{lr} ..... *Rotor leakage inductance*
L_{ls} ..... *Stator leakage inductance*
L_{dm} ..... *d-Axis magnetizing inductance*
L_{qm} ..... *q-Axis magnetizing inductance*
p ..... *Laplace operator, Number of poles*
P_g ..... *Air gap power*
P_m ..... *Mechanical output power*
P_{sl} ..... *Slip power*
R_r ..... *Rotor resistance*
R_s ..... *Stator resistance*
S ..... *Slip per unit (also the Laplace operator)*
T_e ..... *Developed torque*
T_L ..... *Load torque*
T_s ..... *Sampling time*
t_{on} ..... *Turn-on time*
t_{off} ..... *Turn-off time*
σ ..... *Leakage factor*
ψ_{dr} ..... *d-Axis rotor flux linkage*
ψ_{ds} ..... *d-Axis stator flux linkage*
ψ_{qr} ..... *q-Axis rotor flux linkage*
ψ_{qs} ..... *q-Axis stator flux linkage*
ω_e, ω_o ..... *Stator frequency*
ω_m ..... *Rotor mechnical speed*
ω_r ..... *Rotor electrical speed*
ω_{sl} ..... *Slip frequency*

Main Suffices:

a,b,c; A,B,C ..... *pertaining to 3-phase*
d,q;D,Q ..... *pertaining to direct and quadrature axes*
max..... *maximum*
min..... *minimum*

Others:

$\underline{V}$ ..... *Underscore denotes V is a vector quantity*

List of Principal Abbreviations

ASIC ..... *Application Specified Integrated Circuit*
BJT ..... *Biplor Junction Transistor*
BLDCM ..... *Brushless DC motor*
Control-C ..... *A VAX-based Software Simulation Package*
EMF ..... *Electromagnetic force*
EXE ..... *Executable fold in the transputer development system (TDS)*
FET ..... *Field Effect Transistor*
GTO ..... *Gate Turn Off Transistor*
HVIC ..... *High Voltage Integrated Circuit*
IGBT ..... *Insulated Gate Bipolor Transistor*
`inmos`..... *Manufacturer of the Transputer*
LA ..... *Link Adapter*
Link0, Link1, Link2, Link3
..... *Reserved names for the 4 links of the transputer*
LSB..... *Least Significant Bit*
MATLAB ..... *MATrix LABoratory Software Package*
MIMO ..... *Multiple Input Multiple Output*
MMF..... *Magnetromagnetic Force*
MOSFET ..... *Metal Oxide Semiconductor FET*
MSB..... *Most Significant Bit*
`occam`..... *Programming language specially suited to the Transputer*
PSU ..... *Power Supply Unit*
PWM..... *Pulse Width Modulation*
RFI ..... *Radio Frequency Interference*
SISO ..... *Single Input Single Output*
SIT ..... *Static Induction Transistor*
SMPS ..... *Switch Mode Power Supply*
TDS ..... *Transputer Development System*
TRAM ..... *TRAnsputer Module, the board level transputer product from* `inmos`
VLSI ..... *Very Large Scale Integration*
VVVF ..... *Variable Voltage Variable Frequency*

Table of Content

Chapter 1 : Introduction - Review of Variable Speed Drives

Some ten years after the invention of the induction motor by Nikola Tesla in 1886, Ward Leonard published his classic paper [Leonard,1896], in which the now well-known Ward Leonard system was introduced. In the paper, he emphasized ''the desirability of operating an electric motor under perfect and economical control at any desired speed from rest to full speed''. The Ward Leonard system, essentially a voltage control for the armature of a DC machine, was the ''implied'' desirable variable speed drive (VSD) system.

Since then DC drives have dominated the field of adjustable speed electrical drives. Rotary converters driven by constant speed turbines, diesel engines or AC motors, have been used as the power conditioning unit for the DC drive in different types of applications. About half a century ago, these mechanical rotary converters were gradually replaced by static equipment, discharge tubes (thyratrons), mercury arc converters or magnetic amplifiers. Nevertheless, the DC motor, with its good dynamic response and simple requirements for a control system, remained the standard drive motor whenever a wide speed control range and four-quadrant operation of the torque-speed plane were required for an application.

The mechanical commutator of the DC motor, however, presented a serious problem for the otherwise ideal motor for variable speed drives. The sliding contact between the commutator and the carbon brushes, where the transfer of power from the stator to the rotor occurred, was the weakest power link in the system, limiting the power and speed of the drive, increasing the power-to-space ratio and requiring periodic maintenance. The qualification of the DC drives as being ''perfect and economical'' in Leonard's terms needed to be reassessed and Leonard's aspiration in the search of a ''perfect and economical'' drive had not diminished. Much work and research were made over the years to replace the machine commutation by means of static and electronic means external to the motor — the inverter. This consequently led to the development of high power electronic switches capable of rapidly opening and closing inductive circuits such as the induction motor.

Mercury-arc valves were then satisfactorily operated in line-commutated cycloconverters and load commutated synchronous drives. However, it was not until the invention of the thyristor by the Bell Laboratories in 1957 and its commercial introduction by the General Electric Company in the following year that sufficiently powerful, efficient and fast-switching components became available at an acceptable cost so that the design of an adjustable speed drive using an AC machine could be considered. The advantages of the AC machine — in particular robustness, low maintenance, high power-to-weight ratio and low cost of the induction motor, which had been capitalized in the field of constant speed applications, were then available for exploitation in variable speed applications. Because of the complex dynamic interactions that occurs within the control structure of an induction motor, considerable effort has been devoted to the development of control strategies to eliminate the undesirable interactions. These new strategies, however, could not be applied in real-time to the control system because of the inadequacies of the microelectronics technique then.

The introduction of 'field-coordinate control', also known as 'field-oriented control' or more commonly 'vector control', by Hasse in 1969 [Hasse,1969], proved that AC-machines could have controlled characteristics similar to those of a DC motor. This was achieved by means of control schemes functioning in moving coordinates defined by the flux wave or rotor position. The concept of 'vector control' can be likened to the unified theory treatment of machines, where all machines can be analyzed in term of the 'Kron' primitive machine. The subsequent development of different 'vector control' methods enabled the transformation of the complex structure of an AC machine to that of an equivalent DC machine. This enabled the design procedure to become a more straight-forward task. In the early 70's, however, when control circuits were mainly analogue, the requirements of 'vector control' demanded considerable signal processing that could not be engineered satisfactorily by analogue techniques. This was mainly due to the DC drift and thermal related problems associated with analogue circuits. Thus, such new schemes as 'vector control' were considered as academic exercises, rather than a viable industrial innovation.

The introduction of the first practical digital microprocessor ended the dormant period in the application of 'vector control' to industrial drives that lasted for nearly the whole of

the decade of the 70's. This was because the complex control functions could be implemented in software by the microprocessor, thus reducing the amount of hardware required. Further development in microelectronics would further reduce the amount of hardware, with further integration in microelectronics systems. The availability of more powerful microprocessors further increased the range of complex control techniques that can be applied to electrical drive systems.

Against the background of the historical development of the AC drive is the ever-increasing demand for high performance servo systems in robotics, machine tools and aerospace applications that have been previously dominated by DC servo systems. Besides, the conservation and economic issues have gained much public awareness over recent years and have led to some new developments in variable speed motor drives. For example, the electrical traction drives have become increasingly important due to the environmental impact of the conventional internal combustion engines' vehicle. With two-fifths of the carbon dioxide gas in the atmosphere in the UK coming from vehicles [Econews,1990], the advantages which electrical vehicles have to offer are obvious. The viability of the application of induction motors in vehicles is demonstrated by the system developed by Chan [Chan,1987]. The possibility of applying modern control theories to electrical vehicles by means of a multi-processor system is further manifested in the electrical propulsion system proposed by the same author [Chan,1988]. It should also be noted that electrical handling systems have already been widely used in British industry. It may be premature to judge whether the induction motor is environmentally more 'friendly' than DC motors, as other areas such as their development and manufacturing should also be taken into consideration. It is, however, certain that a wider application area for variable speed drives has been opened for the induction motor. Apart from the influence of the environmental issues, the competitiveness of the induction motor in the future variable speed drives market can be best assessed in the following three important aspects.

Firstly, the rapid development in microelectronics and power electronics over recent years has made a great impact on many sectors of engineering. It is interesting to see from Fig.1.1 that the computing power of processing devices has increased ten-fold over each

of the last three decades, and is predicted to increase by the similar amount over the next decade. The increase in computing power now offers the opportunity of implementing in real-time control strategies such as 'vector control', which have been theoretically available for many years. An equally important trend that is not revealed in Fig.1.1 is that the cost of computing power has also fallen rapidly over previous decades and is predicted to fall in the future. It is also interesting to note from Fig.1.2 that further demand for power integrated circuits (ICs) will outweigh the demand for discrete power and microelectronics devices in the near future. This means that further requirements of electrical drives can be satisfied by such customized chips as application specified integrated circuit (ASIC), for example.

Secondly, the ever increasing use of semiconductors in electrical machine technology not only changes the performance of such systems, but also changes the terminology used in the subject area. In the past, motors were exclusively fed directly from the mains, or via a metadyne transformer, or DC generator in the case of DC motors. The drive system then mainly consisted the motor itself, or a motor-generator set as in the Ward-Leonard system. Nowadays, however, the term 'drive' is used to identify the motor and the semiconductor converter that controls the motor. Thus, the 'drive equation' has changed from one that consisted of mainly copper and iron (the motor), to one that consists of copper and iron (motor), and silicon (power electronics and microelectronics). The recently introduced high voltage integrated circuit (HVIC), and the 'smart power' [Emerald,1990] are examples of the latest efforts to combine control logic into power modules, and intelligence into the drive system. Integrated circuits and microprocessors [Signetics,1988] dedicated to waveform generation and close-loop controls are now naturally considered as part of the drive system. The drive system of the future will therefore be expected to have a silicon element that plays a very important role in the area of signal processing and a power element that determines the means by which the power is utilised. It is proposed [Bose,1989] that expert system-based control will play a major role in automating system modelling, control design, simulation study, real time control, test and diagnostics.

Thirdly, great progress has also been made in new types of motors intended in variable speed applications. For example, the switched reluctance motor presents an emerging and

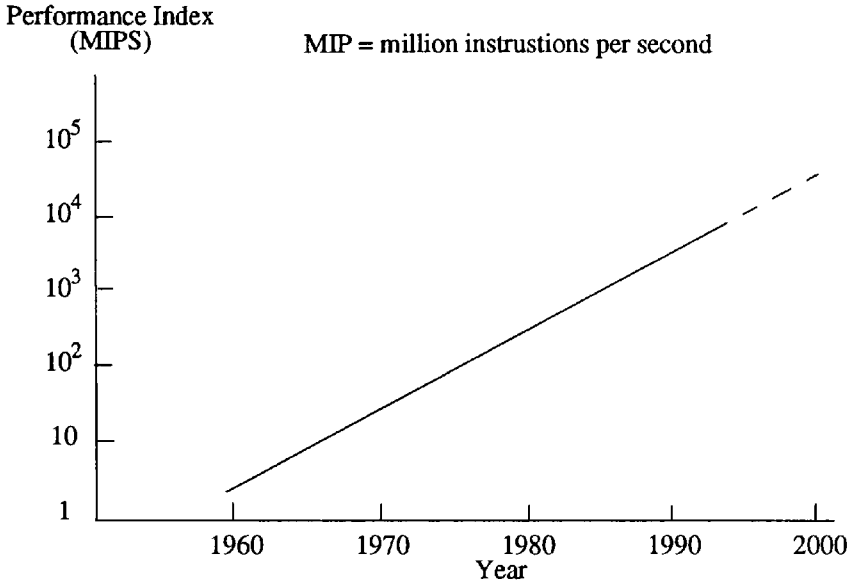


Fig.1.1 Increase of processing power due to advancement in in VLSI technology

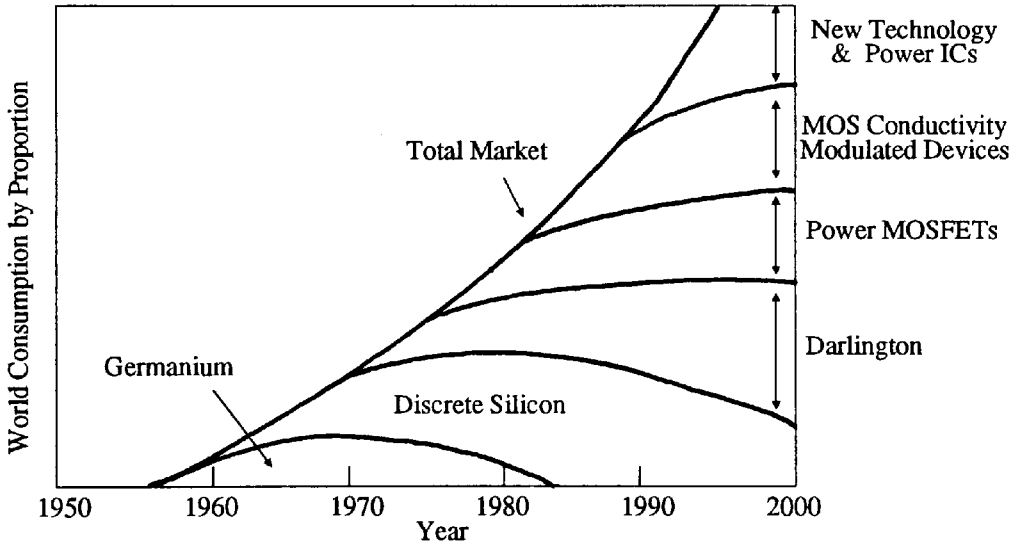


Fig.1.2 Projected use of power switching devices through the year 2000 [Cogan,1985]

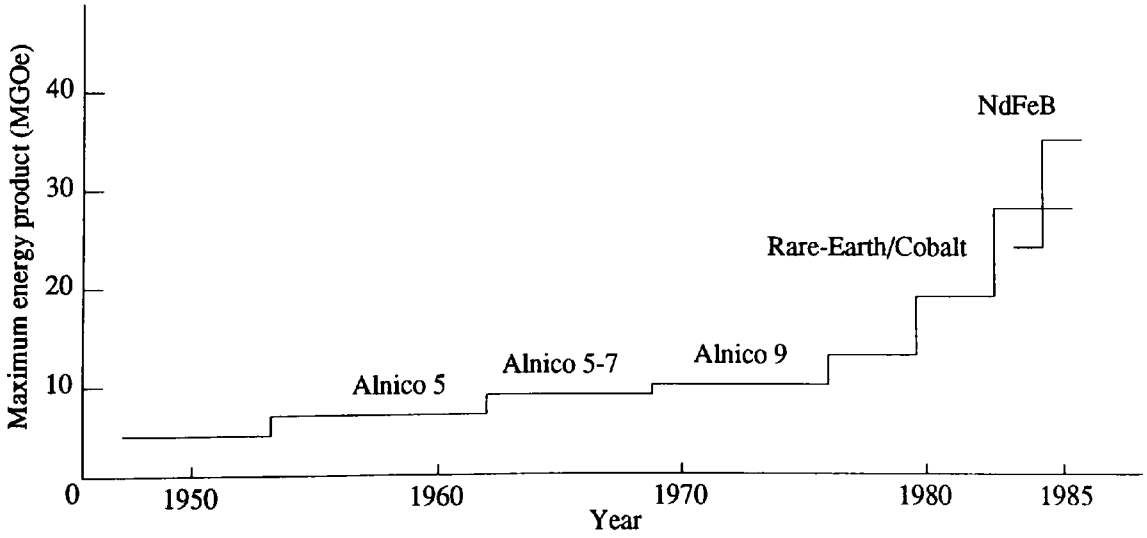


Fig.1.3 Recent development of magnet materials [Miller,1989]

promising contender in the electrical drives' market. This type of drive system mainly consists of a custom designed switching package that is used to excite the stator field of the salient-pole rotor of a reluctance motor. The machine has salient poles on both the stator and the rotor. The rotor is made of laminated steel with no windings or magnets whereas the stator poles are wound with concentrated windings. Current switching in the stator windings takes place in a manner determined by the position of the rotor to produce reluctance torque. This developed torque is a nonlinear function of the rotor position and current. It offers the possibility of very precise control even in open loop applications. Although still in its infancy, the switched reluctance drive may well capture a large part of the market for variable speed drives, especially at the lower power area end. The development of magnet materials such as rare earth magnet and Neodymium Iron Boron [Miller,1989] over the recent years, as illustrated in Fig.1.3, has increased the power level of these machines and hence their applications. An example of the present proliferation of permanent magnet machines is the brushless DC motor (BLDCM), which has become one of the prime contenders for variable speed drives market. The increasing interest in the BLDCM over recent years is reflected by the large amount of literature available [Kusko,1987;Luk,1989]. The BLDCM can be seen as an inside-out construction of the conventional DC motor, that is to say, the armature (stator) is stationary and the field system (rotor) rotates. A typical 3-phase BLDCM has its stator being fitted with a balanced three-phase winding and the rotor poles consisting of permanent magnets. Its potential high power-to-weight and high torque-to-inertia ratios, together with low maintenance and spark-free features make it ideal for the aerospace, mining and chemical industries. Because there is no power dissipation in the rotor, the motor enjoys a higher efficiency and a better thermal rating. The disadvantages of BLDCM, however, mainly consist of extra drive and logic circuit requirement, the possibility of cogging torque when high flux density rare-earth magnets are used, and the requirement of accurate knowledge of the rotor position for switching purposes.

The application of AC induction motors to wide range variable speed industrial drives has been restricted by the extensive requirement of the power converter. For example, considerable difficulty has been experienced in producing a low cost variable voltage, variable frequency (VVVF) source with low harmonic distortion. This has resulted in the overall cost of the AC variable speed drive being comparable to that of an equivalent DC

motor drive. It should also be stressed that the complexity required of the control strategy for an AC drive is very much greater than that of required for an equivalent DC drive system. Indeed, the emphasis on AC variable speed drive system design has only recently shifted from the task of achieving efficient and reliable power conversion to improvement of its transient response. From a control point of view, an induction motor has a complex multiple-input multiple-output (MIMO) control structure. 'Vector control', as previously mentioned, is a method intended to improve the dynamic response of the induction motor. 'Vector control' has been under constant development for many years by such people as Blaschke, Hasse, and more recently, by Leonhard and others. The literature [Leonhard,1985] suggests that one of the major setbacks in the development of a digital or microprocessor-based 'vector control' system has been the inadequate computing speed of the microprocessors available. More recently, however, powerful signal processors and multi-processors' systems [Kenji,1985;Harashima,1985] have been proposed for the efficient implementation of 'vector control' strategy.

Since the AC machine has a non-linear multi-variable structure, and such variables as rotor currents are not accessible without modifications being made to the motor, an advanced and fully digitized control strategy would require very fast processors possibly working in a parallel fashion. Present-day sequential multi-processor systems, when using conventional bus system, present considerable hardware and software development complexity to the design engineer. The problem of 'bus contention' in a bus system, when more than one processor are outputting data to the data bus at the same time in a multi-processor network, is a well-known design problem. In addition, the software of such systems should be programmed in some concurrent languages, such as *Ada* and *Modulus*, which are much more difficult to use than the conventional sequential languages, such as *Basic* and *Fortran* [Howe,1987]. The signal processor, though provides very high computing speed, may be considered as an inflexible design tool. It presents considerable difficulty in the design of a multi-processor system due to its intrinsic serial architecture. The use of single conventional microprocessors or microcontrollers, however, has been heavily exploited especially in variable drive systems requiring less stringent performance. Nevertheless, the application of parallel processing in high performance control of induction motors has remained a virtually uncharted area.

The **inmos** transputer [inmos,1988], which became available in 1986, was originally conceived and designed as a processor primarily intended to be used in embedded control systems, with an architecture that facilitated parallel processing. Although the processing power of the transputer has in practice led to many other applications such as image processing, it remains an excellent tool for implementing real-time control systems. The main object of the research project described in this thesis was to develop and implement a transputer-based 'vector control' system for variable speed induction motor drives. The potential of identifying and applying control algorithms that could exploit parallel processing and greatly improve the dynamic response of such drive, offered a great challenge to the researcher. The recent fall in price of existing transputers and the introduction of very much more powerful next generation of transputers should have a great influence on the future of motor control systems thereby allowing the implementation of very sophisticated strategies at an acceptable cost.

Chapter 2 : Theory and Analysis of AC-Machine Drives

2.1 INTRODUCTION

This chapter concentrates on the establishment of the electrical machine theory and its application to the analysis of the dynamics of AC machines. It is mainly the 3-phase squirrel-caged induction motor that is analysed, although much of the analysis can be equally applied to other AC machines, or, in some cases, even DC machines. In the analysis, a number of assumptions are made, for example, the power dissipated in acoustic noise and radio wave propagation radiation to the surroundings are neglected. It should also be noted that a number of basic electromagnetic principles governing the operation of electrical machines are assumed without reference. It was found necessary to make such assumptions in order to establish a basis upon which the analysis of induction machines could be based.

The review of the analytical methods that can be applied to three-phase induction motors is presented in order to compare and justify the method chosen for the analysis of the motor to which this thesis relates. The methods of analysis included in the review mainly consist of the Laplace transform, Fourier series, complex variable and state variable. The advantage of using the state variable method over other methods for modelling the drive system investigated becomes apparent, when consideration is given to the matrix-oriented computer software available.

The detailed mathematical analysis of an induction motor drive developed in the reference [Lipo,1975], in which a sound mathematical knowledge in matrix algebra is assumed, was found to be of great assistance when writing the simulation program in the VAX-based matrix-oriented computer software called 'Control-C'. It was, however, replaced by a much more cost effective PC-based compatible software called 'PC-MATLAB', which became available at a later stage in the research, in developing motor drive models described in the following chapters.

The validity of the drive model simulated in 'Control-C' was justified by the good agreement of the simulation results with those given by Lipo. This agreement also justifies the use of state space method in the analysis of induction motor drives. The advantages of

using a matrix oriented software approach in the simulation of a motor drive system in terms of state space analysis is further emphasized by the conciseness of the program code listing in the Appendix-B.

2.2 CONSTRUCTION AND BASIC THEORY OF INDUCTION MACHINES

2.2.1 Physical Structure of the Induction Motor

An idealized 3-phase 2-pole cylindrical induction motor is depicted in Fig.2.1. In the arrangement shown, there is an outer stationary member and an inner rotating member mounted in bearings fixed to the stationary member. The two elements carry cylindrical

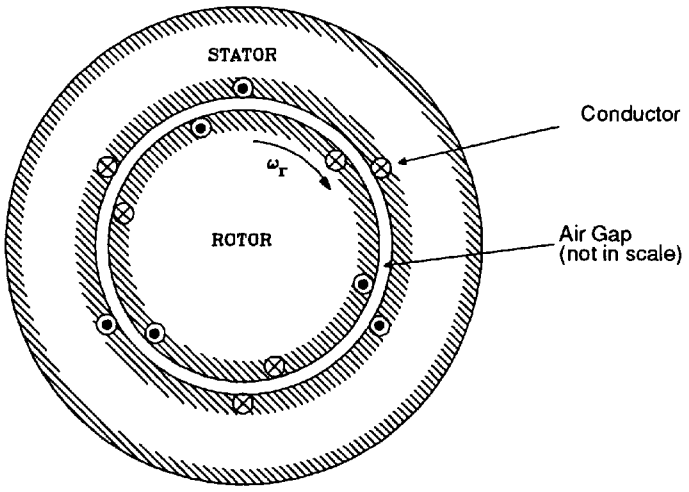


Fig.2.1 A simplified 2-pole 3-phase cylindrical induction machine

iron cores separated by an air-gap. Embedded in the respective iron cores are the stator and rotor windings. However, with all induction motors, it is the air gap that forms the critical part so far as machine performance and behaviour are concerned. It is to this region that most theory and analysis are concerned.

A rotating magnetic field is set up by the voltage applied to the stator windings and the speed of rotation is uniquely determined by the number of pole pairs and supply frequency such that

$$\omega_S = \frac{60 * f}{p} \qquad (2.1)$$

where ω_s is the synchronous speed in rev/min; f is frequency in Hz; and p is the number of pole pairs. The rotating field induces currents in the rotor windings which interact with the rotating field to produce the torque. This torque accelerates the rotor until finally the rotor revolves at some speed ω_r, which is less than the synchronous speed ω_s. As the rotor requires a torque to balance the load torque and this same torque is experienced by the rotating air gap field, a situation occurs in which two rotating members experience the same torque but have a different speed. Therefore, a power difference exists between the motor stator power input ($\omega_s T$) and the rotor power output ($\omega_r T$) where T is the common torque and ω_s and ω_r are the angular speeds of the air gap field and the rotor respectively. This power difference is fundamental to all induction machines. The concept of 'slip' is introduced to define the rotor speed as:

$$\omega_r = (1 - s) * \omega_s \tag{2.2}$$

where the slip, s, has a value between 0 and 1 in the motoring mode. The power difference, P_{slip} is called the 'slippage power', and can also be defined as follows:

$$P_{slip} = (\omega_s - \omega_r) * T = s * \omega_s * T = s * P_s \tag{2.3}$$

where P_s is air gap power, which is the part of the stator power transmitted across the air gap. The power flow diagram for the induction motor is depicted in Fig.2.2.

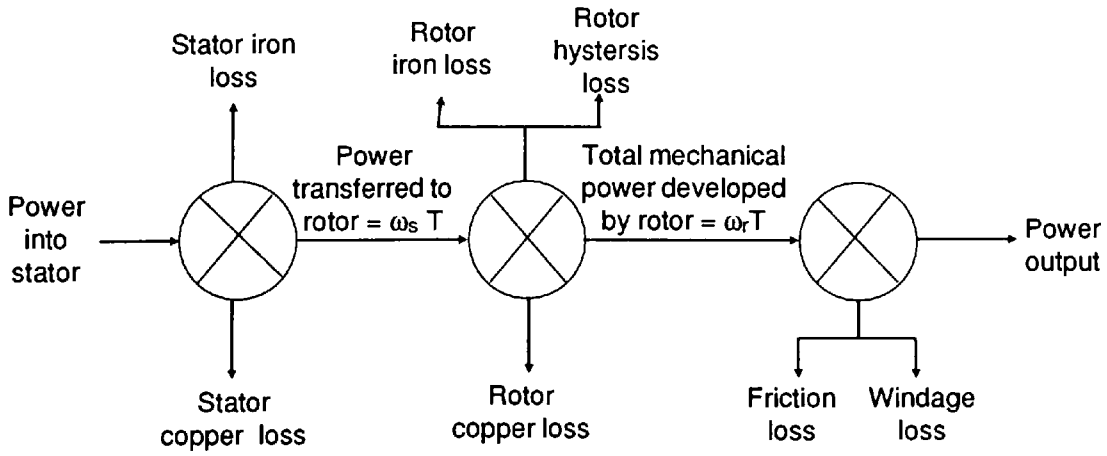


Fig.2.2 Power flow diagram of an induction motor

A unique feature of the induction machine arises from the effect of the rotor motion. At standstill, or when the rotor is locked, there can be no motional EMF and this is similar to a transformer, in which pulsation of the stator (primary) field causes an electrical-to-electrical conversion. The energy will be dissipated as rotor copper loss. If the rotor is allowed to run, each rotor coil will experience the pulsation of the primary magnetic field at a rate reduced in proportion to the increase of rotor speed. This motional EMF that is proportional to $(1-s)$, and the rotor current due to the transformer EMF that is proportional to s, gives electromechanical conversion at all speeds other than synchronous and standstill where the product of this motional EMF and rotor current is zero.

2.2.2 Fundamental Principles

The principle of electromagnetic induction is fundamental to the basic theory of machines and can be expressed as follows:

$$e = N\left(\frac{d\Phi}{dt}\right) \tag{2.4}$$

where e is the generated electromagnetic force (EMF) in volts, N is the number of turns, and Φ is the mean flux per turn in webers. The concept of magnetic flux lines, Φ webers, and their linkage with the N turns of a coil is introduced in equation (2.4) here and depicted in Fig.2.3. Since not all the flux lines link all of the turns, the term 'flux linkage' is introduced and is defined by the equation $\Phi = \lambda/N$. The flux linkage (λ) is given by the sum of the product of the number of turns encircled by each flux line. A simple case is demonstrated in Fig.2.3. Since the total flux linkage of an inductor of L henrys carrying a current i amperes is Li, the induced EMF e, from equation (2.4), is given by:

$$e = \frac{d}{dt}(L\,i) = i\frac{dL}{dt} + L\frac{di}{dt} \tag{2.5}$$

The principles of 'interaction' and 'alignment' are also commonly used [Say,1985] to analyze and explain the performance of electrical machines. The principle of 'interaction' is best expressed by the vector equation $\underline{f} = \underline{B} \times \underline{i}$, where the force $\underline{f}$ on a conductor is the cross product of the current $\underline{i}$ it carries and the a magnetic field $\underline{B}$ it is in. The 'alignment' principle states that when high-permeability materials such as iron, are placed in low-permeability material medium such as air medium, where a magnetic field is established, a developed mechanical force will cause the iron to align itself with the field direction in such a way as to minimize the reluctance of the system. It is this principle that is responsible for and can be used for predicting the 'reluctance torque' in electrical machines.

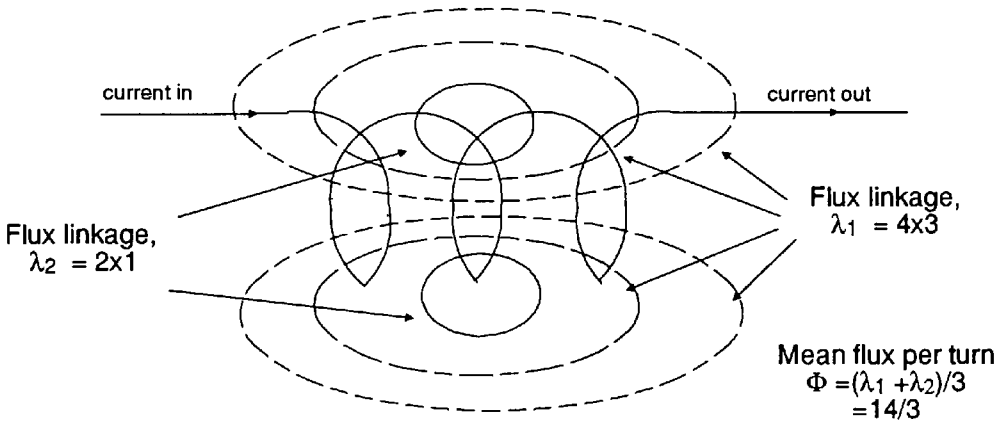


Fig.2.3 Magnetic flux establised by a 3-turn coil

2.2.3 Development of Machine Equations

2.2.3.1 The 6-coil Machine Model

For the purpose of the present analysis, core losses, magnetic saturation effects and magnetromagnetive force (MMF) space harmonics have been neglected and the resistances of stator and rotor are assumed constant. A 3-phase induction motor can then be envisaged as two sets of coils spatially displaced as shown in Fig.2.4. There are 6 coupled windings, on *A-B-C* stator and on *a-b-c* rotor. The voltage equation of each winding has a term for the voltage drop across its resistance, self inductance and 5 mutual inductances. The voltage equation for the phase-A winding of the stator is:

$$v_A = R_A i_A + p(L_{AA} i_A) + p(L_{AB} i_B) + p(L_{AC} i_C) + p(L_{Aa} i_a) + p(L_{Ab} i_b) + p(L_{Ac} i_c) \quad (2.6)$$

where R_A is the resistance of phase A winding;
L_{AA} is the self inductance;
L_{AB}, L_{AC} are the mutual inductances between the respective stator windings;
L_{Aa}, L_{Ab}, L_{Ac} are the mutual inductances between the stator and the respective rotor windings; and
p is the d/dt operator.

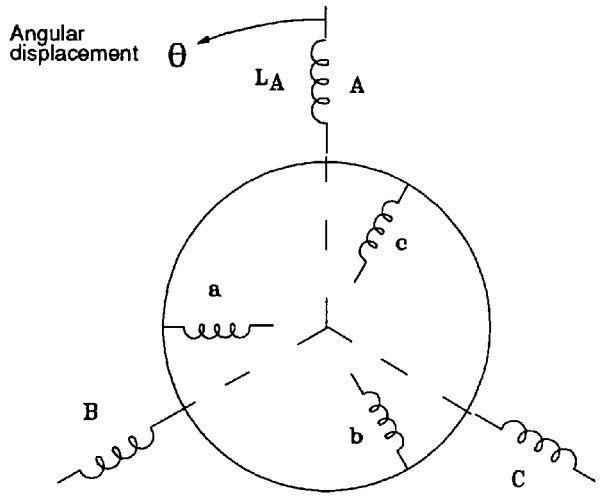


Fig.2.4 The 6-coil model of a 3-Phase induction motor
(N.B. A solid bar is also used to denote an inductance in this thesis)

This expression includes resistance voltage drop, the transformer voltage, and the motional voltages that are torque related. Similarly, voltage equations for other phase windings of the machine may be written down in the same straight-forward manner. The equations for the six windings can be expressed concisely in matrix form as:

$$\underline{v} = R\,\underline{i} + p\,(L\,\underline{i}) \quad (2.7)$$

where R and L are $6x6$ square matrices of the winding coefficients; $\underline{v}$ and $\underline{i}$ are 6-element column vector.

The mutual inductances between the coils means that the current in each coil depends upon the current in the other coils in the machine. This interdependence makes analytical solution of the voltage equations of the 3-phase machine difficult, not least the number (six) of equations involved. Furthermore, the changes in rotor position

affect spatial orientation of the rotor coils with respect to the stator coils and consequently affect their mutual inductances. Further complications arise if the rotor has saliency. Changes of rotor position may also affect the magnetic circuit of the machine due to, for example, leakage inductance, and change the self inductances of the coils on the same member. For a 3-phase squirrel cage induction motor with a uniform air gap, however, the self inductances and mutual inductances between stator phases and rotor phases will be independent of the rotor position. By expanding equation (2.7) as follows:

$$\underline{v} = R\underline{i} + L\frac{d\,i}{dt} + \underline{i}\,\frac{dL}{dt} = R\underline{i} + L\frac{d\,i}{dt} + \underline{i}\,\frac{dL}{d\theta}\frac{d\theta}{dt} \qquad (2.8)$$

and writing $\omega_r = \frac{d\theta}{dt}$ and $G = \frac{dL}{d\theta}$, then equation (2.7) can be written as :

$$\underline{v} = R\underline{i} + \omega_r G\underline{i} + L\frac{d\,i}{dt} \qquad (2.9)$$

where G is a $6x6$ square matrix of the winding mutual inductance, and ω_r is the rotor speed.

It is noteworthy that in solving directly the $6x6$ matrix equations, computational failure results due to the redundant information of the equation. This redundant information comes about because $ic = -(ia + ib)$ in a 3-wire machine drive system, and because the three phase voltages can be replaced by two line voltages. Such a transformation, mainly motivated from a computational point of view and described in great detail in [Hindmarsh,1982], incidentally results in an equivalent machine that is easier to solve computationally. However, it still involves mutual inductance terms as a function of rotor position. A transformation of special interest is the one in which the machine windings are transformed into a d-axis and q-axis array of stator and 'pseudo-stationary' rotor coils, which results in considerable simplification of the analysis of the squirrel cage induction motor.

2.2.3.2 The Two-axis Primitive Machine

The name primitive machine is given to the *d-q* array of coils for which connection coils are absent and each coil is supplied with a separate externally applied voltage. This machine is commonly known as the 'Kron' primitive machine. The transformation of variables is equivalent to the physical action of the commutator. This enables AC and DC machines to be expressed in identical form, and a routine process permits these equations to be set up by inspection [Adkins,1975].

The transformation can be referred to a *d-q* axis system fixed either on the stator or the rotor or rotating in synchronism with the applied voltages. It should be noted that other frames of reference, such as that referred to the rotor, are also possible. The following sections describe the two most common transformations.

(a) Transformation Referred to Stator

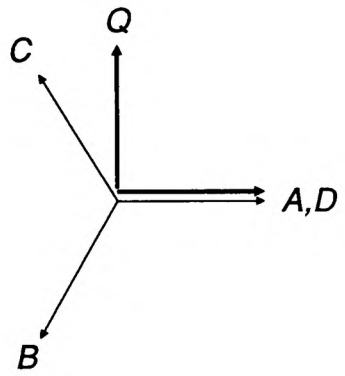
The transformation referred to the stator can be realised in two steps as illustrated in Fig.2.5. The first step involves the three-phase to two-phase transformation with no change of space-frame, as shown in Fig.2.5(b). The current transformations from frame $\{A,B,C\}$ to $\{D,Q,\Gamma\}$ for the stator, and $\{a,b,c\}$ to $\{\alpha,\beta,\chi\}$ for rotor, can be expressed in matrix forms as follows:

$$\underline{i}_{DQ\Gamma} = D\,\underline{i}_{ABC} \tag{2.10}$$

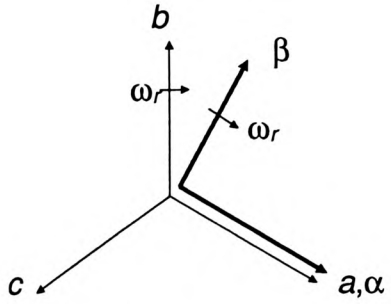
$$\underline{i}_{\alpha\beta\chi} = D\,\underline{i}_{abc} \tag{2.11}$$

where $D = \sqrt{\frac{2}{3}}\begin{bmatrix} 1 & 1/2 & 1/2 \\ 0 & -3/2 & 3/2 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

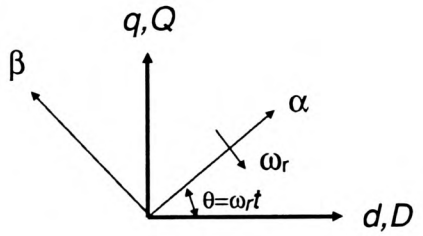
Similar relationships can be obtained for the voltages. The second step involves the change of space-frame for the rotor, as shown in Fig.2.5(c). The transformation of current from frame $\{\alpha,\beta,\chi\}$ to frame $\{d,q,\chi\}$, can be expressed in matrix forms as follows:



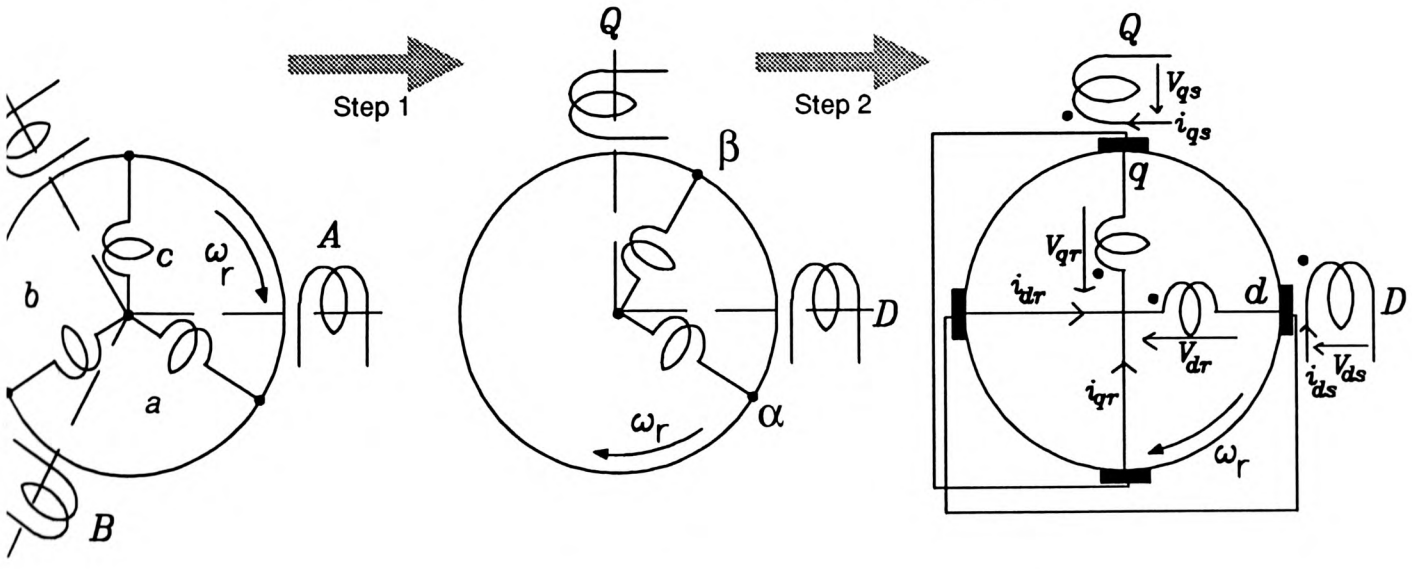
(for stator)



(for rotor)



(for rotor only)



(a) (b) (c)

Fig.2.5 *A-B-C* to *d-q* transformation of induction motor

$$\underline{i}_{dq\chi} = E\,\underline{i}_{\alpha\beta\chi} \tag{2.12}$$

where $E = \begin{bmatrix} \cos\omega_r t & \sin\omega_r t & 0 \\ -\sin\omega_r t & \cos\omega_r t & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Combining equations (2.11) and (2.12) gives:

$$\underline{i}_{dq\chi} = E\,\underline{i}_{\alpha\beta\chi} = E\,D\,\underline{i}_{abc}$$

or

$$\underline{i}_{dq\chi} = C\,\underline{i}_{abc} \tag{2.13}$$

where

$$C = E\,D = \sqrt{\frac{2}{3}}\begin{bmatrix} \cos\omega_r t & \cos(\omega_r t+120^o) & \cos(\omega_r t-120^o) \\ -\sin\omega_r t & -\sin(\omega_r t+120^o) & -\sin(\omega_r t-120^o) \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

Similar relationships can be obtained for the voltages. The *6x6* matrix equation (2.7) of the motor, when transformed to *d-q* axis and referred to the stator can therefore be expanded as:

$$\begin{bmatrix} v_{qs}^{\,s} \\ v_{ds}^{\,s} \\ v_{qr}^{\,s} \\ v_{dr}^{\,s} \end{bmatrix} = \begin{bmatrix} R_s + L_s p & 0 & L_m p & 0 \\ 0 & R_s + L_s p & 0 & L_m \\ L_m p & -\omega_r L_m & R_r + L_r p & -\omega_r L_r \\ \omega_r L_m & L_m p & \omega_r L_r & R_r + L_r p \end{bmatrix} \begin{bmatrix} i_{qs}^{\,s} \\ i_{ds}^{\,s} \\ i_{qr}^{\,s} \\ i_{dr}^{\,s} \end{bmatrix} \tag{2.14}$$

where the superscript s denotes the quantities being referred to the stator. The order of the matrices in (2.14) is reduced by 2 as a result of the *d-q* transformations.

The electromagnetic torque T_e is given by:

$$T_e = n\,\underline{i}\,G\,\underline{i}^{\,T} \tag{2.15}$$

where n is the number of pole-pairs and $\underline{i}^{\,T}$ is the transpose of $\underline{i}$. Equations (2.14) and (2.15) may be combined to form the following fifth-order equation, which describes the electrical and mechanical dynamics of the complete drive system.

$$\begin{bmatrix} v_{qs}{}^{s} \\ v_{ds}{}^{s} \\ v_{qr}{}^{s} \\ v_{dr}{}^{s} \\ T_L \end{bmatrix} = \begin{bmatrix} r_s + L_s p & 0 & L_m p & 0 & 0 \\ 0 & R_s + L_s p & 0 & L_m & 0 \\ L_m p & -\omega_r L_m & R_r + L_r p & -\omega_r L_r & 0 \\ \omega_r L_m & L_m p & \omega_r L_r & R_r + L_r p & 0 \\ i_{dr} L_m & i_{qr} L_m & -i_{qr} L_s & i_{dr} L_r & Jp + R_F \end{bmatrix} \begin{bmatrix} i_{qs}{}^{s} \\ i_{ds}{}^{s} \\ i_{qr}{}^{s} \\ i_{dr}{}^{s} \\ \omega_r \end{bmatrix} \tag{2.16}$$

where J is the moment of inertia of the rotating mass, and R_F is a mechanical coefficient representing dissipation due to friction and windage.

(b) Transformation Referred to Rotating Frame of Applied Stator Voltage

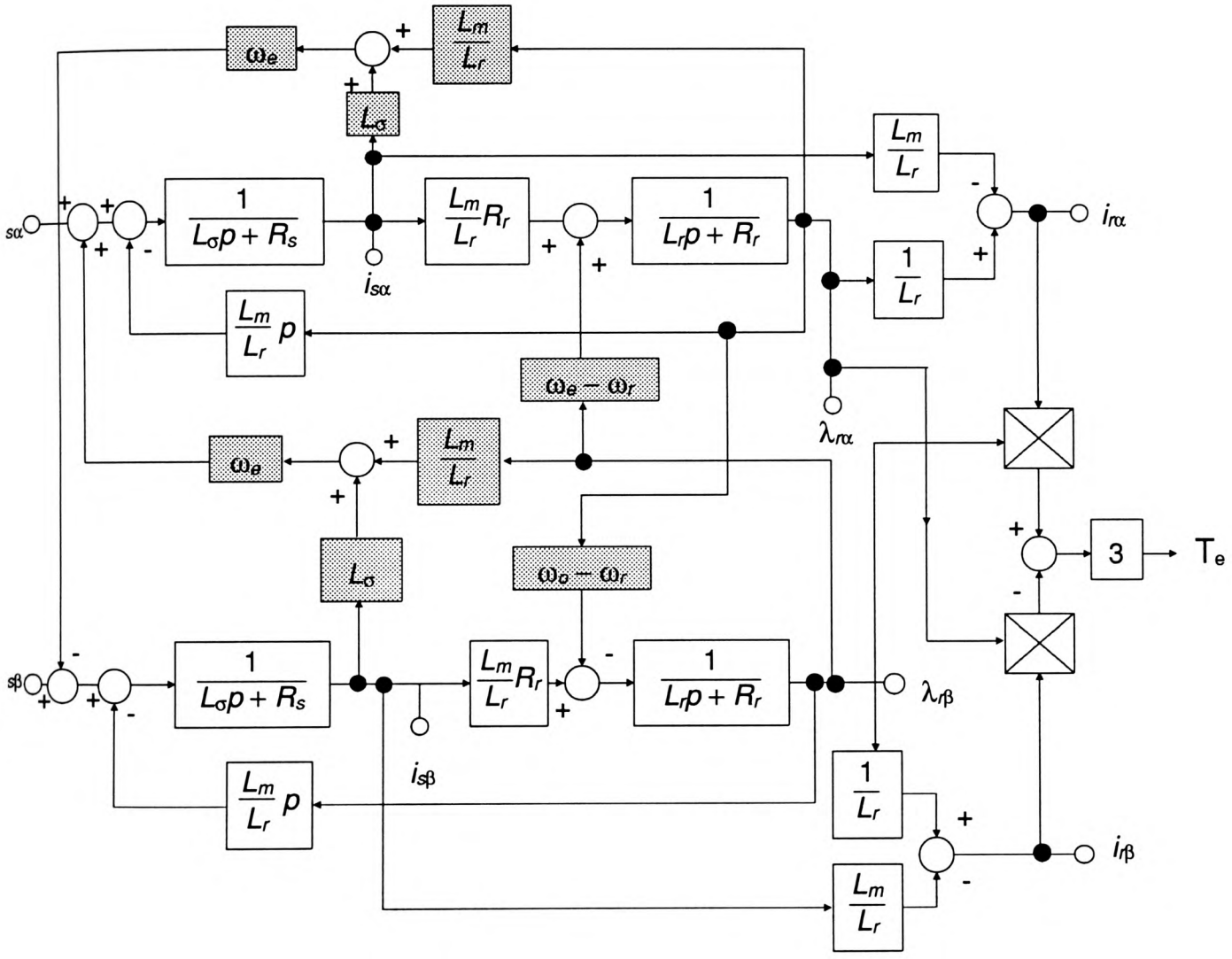
The transformation referred to the rotating frame in synchronous with the applied voltage has been popular among a number of authors [Harashima,1985; Chan,1988] in the analysis and investigation of vector control in induction motors. In this transformation, rotor fluxes are used instead of rotor currents in formulating the motor equations. The relationship between the rotor currents ($i_{r\alpha}$,$i_{r\beta}$) and fluxes ($\psi_{r\alpha}$, $\psi_{r\beta}$) in the frame of reference $\{\alpha,\beta,\chi\}$, are given by equation (2.17) as follows:

$$\psi_{\alpha r} = L_r * i_{r\alpha} + L_m * i_{s\alpha} \tag{2.17a}$$

$$\psi_{\beta r} = L_r * i_{r\beta} + L_m * i_{s\beta} \tag{2.17b}$$

When the rotor flux equations (2.17) are substituted into the motor equation (2.14), the following equations for the induction motor in α–β frame may be obtained:

$$\begin{bmatrix} v_{s\alpha} \\ v_{s\beta} \\ v_{r\alpha} \\ v_{r\beta} \end{bmatrix} = \begin{bmatrix} R_s + L_\sigma p & -L_\sigma \omega_e & \frac{L_m}{L_r} p & -\frac{L_m}{L_r}\omega_e \\ L_\sigma \omega_e & R_s + L_\sigma p & \frac{L_m}{L_r}\omega_e & \frac{L_m}{L_r} p \\ -\frac{L_m}{L_r} R_r & 0 & \frac{R_r}{L_r} + p & -(\omega_e - \omega_r) \\ 0 & -\frac{L_m}{L_r} R_r & \omega_e - \omega_r & \frac{R_r}{L_r} + p \end{bmatrix} \begin{bmatrix} i_{s\alpha} \\ i_{s\beta} \\ \lambda_{r\alpha} \\ \lambda_{r\beta} \end{bmatrix} \tag{2.18}$$



Note: indicates a coupling term

Fig.2.6 Block diagram of induction motor in α–β Frame

where $L_\sigma = L_s - \frac{L_m^2}{L_r}$

The electromagnetic torque T_e is given by:

$$T_e = 3\,(\lambda_{r\beta} i_{r\alpha} - \lambda_{r\alpha} i_{r\beta}) \qquad (2.19)$$

The block diagram of the induction motor referred to the α–β frame is depicted in Fig.2.6. It may be seen from the figure that the interconnections of blocks are responsible for the non-linearity of the characteristic of the induction motor. Such interconnections, if not decoupled, give rise to the sluggish dynamic response that is characteristic of induction motors.

It will be seen how the model of motor referred to this frame of reference is applied to the simulation of vector control of the drive system being investigated in the next two chapters.

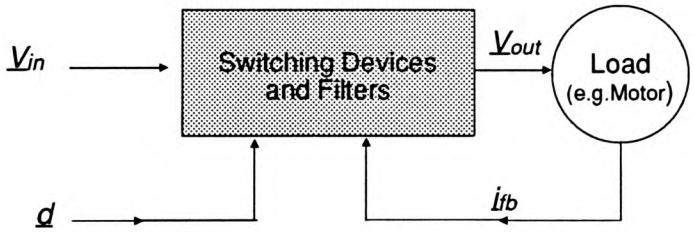


Fig.2.7 Block diagram of power converter model

2.4 REVIEW OF ANALYSIS TECHNIQUES

2.4.1 Power Converter Model

The general block diagram of a power converter system is shown in Fig.2.7. The purpose of the circuit is to efficiently convert an input voltage V_{in} of a particular waveform to the output voltage V_{out} of a different waveform. The conversion is achieved by a set of power semiconductor switches which are opened and closed in a particular sequence defined by the input, d. However, the nature of the switches and

the load may be such that the conduction intervals of some types of switches such as the thyristor, and hence the output voltage $\underline{V}_{out}$, also depend upon the load current. This complication is indicated in Fig.2.7 by the feedback current signal i_{fb}. A major difficulty arises from the discontinuous nature of the binary control signal, $\underline{d}$, which in the case of a PWM drive, may be a set of pulse trains with the mark-to-space ratio governed by a certain strategy. This problem is compounded by the effect of the load current, as mentioned above, on the conduction intervals of the power semiconductor switches and consequently the load voltage. Further difficulty arises when the load, such as an induction motor drive, is non-linear. The analysis of a power converter circuit is therefore often very involved because of such complexity and the non-linearity of the equations describing these operations. It is rarely possible to determine all the variables in the converter circuit as a function of time without some appropriate simplifications. The approximate solutions obtained from such simplifications, however, may provide only limited information on the current waveforms or stability under some operating conditions.

2.4.2 Laplace Transform-Unit Step Function

Many power semiconductor circuits can be analyzed by a straightforward application of Laplace transforms, although the algebra usually becomes quite involved. A thorough treatment of this approach is given in the reference [Takeuchi,1968]. The method is most often applied to phase-controlled circuits, where the load voltage consists of a number of segments of sinusoidal waveform. The analysis entails the representation of the voltage to the load as an infinite series in which each term is the product of a sine wave and a switching function consisting of the difference of unit step functions, such that each term represents one period, or quasi-period, during transient states, of the voltage.

Once the $\underline{V}_{out}$ is determined, its Laplace transform is obtained by using the well known transform formulas and the time-domain shifting theorem. Then the equation for the load current i_{out} as a function of $\underline{V}_{out}(t)$ is written and transformed to the s-domain.

Using substitution the load current is found as a function of s and possibly other unknowns such as the thyristor extinction angle in line commutated circuits. Applying the inverse Laplace transformation yields an explicit function for $i_{out}(t)$ as an infinite series. The main advantage of this expression is its compactness while retaining exactness even during the starting transient. Thus the approach is a more direct procedure for obtaining the current than a repeated solution of the circuit differential equation over each period.

The approach is illustrated in its application to 3-phase induction motor, which is based on the reference [Bedford,1972]. Based on the assumptions of balanced three-wire, steady-state operation and completely known stator phase voltages:

$$\begin{aligned} v_1(t) &= v(t)\, u(t) \\ v_2(t) &= v(t-\frac{2\pi}{3\omega})\, u(t-\frac{2\pi}{3\omega}) \\ v_3(t) &= v\,(t-\frac{4\pi}{3\omega})\, u\,(t-\frac{4\pi}{3\omega}) \end{aligned} \qquad (2.20)$$

The analysis begins by writing the Laplace transform, $V(s)$, of one of the phase voltages. These voltages are balanced and periodic, so the Laplace transforms of the other two phase voltages can be expressed in terms of $V(s)$ using the shifting theorem. Taking the transformation, the stator voltages can be expressed in terms of the single quantity $V(s)$.

$$\begin{aligned} V_1(s) &= V(s) \\ V_2(s) &= V(s)\, e^{-as} \\ V_3(s) &= V(s)\, e^{-2as} \end{aligned} \qquad (2.21)$$

where $a = 2\pi/(3\omega)$

Taking the *d-q* transformation for equation (2.21) as described in section 2.2.3.2, results in:

$$V_{sd}(s) = \sqrt{2/3}\, V(s)\,(1 - \tfrac{1}{2}e^{-2as}) = \sqrt{1/6}\, V(s)\,(2 + e^{-as})\,(1 - e^{-as}) \qquad (2.22)$$
$$V_{sq}(s) = \sqrt{2/3}\, V(s)\,(-\sqrt{3}/2\, e^{-as} + \sqrt{3}/2\, e^{-2as}) = -\sqrt{1/2}\, V(s)\, e^{-as}\,(1 - e^{-as})$$

Similarly, the stator and rotor currents can be expressed in terms of $I_s(s)$ and $I_r(s)$, which are the transforms of the unknown stator and rotor currents in one phase. Expressions can be written for the unknown currents:

$$\begin{aligned} I_{sd}(s) &= \sqrt{1/6}\, I_s(s)\,(2 + e^{-as})\,(1 - e^{-as}) \\ I_{sq}(s) &= -\sqrt{1/2}\, I_s(s)\, e^{-as}\,(1 - e^{-as}) \\ I_{rd}(s) &= \sqrt{1/6}\, I_r(s)\,(2 + e^{-as})\,(1 - e^{-as}) \\ I_{rq}(s) &= -\sqrt{1/2}\, I_r(s)\, e^{-as}\,(1 - e^{-as}) \end{aligned} \qquad (2.23)$$

Assuming a short-circuited rotor, thus the rotor voltages become zero, the motor equation (2.14) becomes as follows:

$$\begin{bmatrix} R_S + sL_S & 0 & sL_m & 0 \\ 0 & R_S + sL_S & 0 & sL_m \\ sL_m & -\omega_r L_m & R_r + sL_r & -\omega_r L_r \\ \omega_r L_m & sL_m & \omega_r L_r & R_r + sL_r \end{bmatrix} \begin{bmatrix} I_{sd}(s) \\ I_{sq}(s) \\ I_{rd}(s) \\ I_{rq}(s) \end{bmatrix} = \begin{bmatrix} V_{sd}(s) \\ V_{sq}(s) \\ 0 \\ 0 \end{bmatrix} \qquad (2.24)$$

Combining equations (2.22) and (2.23) with (2.24), and through simplfying, the following equation may be obtained:

$$\begin{bmatrix} R_S + sL_S & sL_m \\ sL_m & R_r \dfrac{2 + e^{-as}}{2 + e^{-as}\,(1 + \sqrt{3}\dfrac{\omega_r}{s})} + sL_r \end{bmatrix} \begin{bmatrix} I_s(s) \\ I_r(s) \end{bmatrix} = \begin{bmatrix} V(s) \\ 0 \end{bmatrix} \qquad (2.25)$$

Eliminating $I_r(s)$ results in a transfer function in the following form:

$$\frac{I_s(s)}{V(s)} \triangleq Y(s) = \frac{L_r}{L_S L_r - L_m^2} H(s)$$

where

$$H(s) = \frac{2(s+\frac{1}{\tau_r}) + e^{-as}(s+\frac{1}{\tau_r}+\sqrt{3}\omega_r)}{2[s^2 + s(\frac{1}{\tau_1}+\frac{1}{\tau_2}) + \frac{1}{\tau_r\tau_1}] + e^{-as}[s^2 + s(\sqrt{3}\omega_r + \frac{1}{\tau_1}+\frac{1}{\tau_2}) + \frac{1}{\tau_1}(\frac{1}{\tau_r}+\sqrt{3}\omega_r)]} \quad (2.26)$$

The values of τ_r , τ_1 , τ_2 are given by:

$$\tau_r = \frac{L_r}{R_r}, \quad \tau_1 = \frac{L_S L_r - L_m^2}{L_r R_S}, \quad \tau_2 = \frac{L_S L_r - L_m^2}{L_S R_r}$$

From a star-connected motor, the transfer function can be expressed in terms of line quantities as follows:

$$V_L(s) = V_1(s) - V_2(s) = V(s)\,(1 - e^{-as}) \quad (2.27)$$

$$I_L(s) = \sqrt{2/3}\, I_{s\alpha} = I_S(s)\,\frac{(2 + e^{-as})\,(1 - e^{-as})}{3} \quad (2.28)$$

$$K(s) \triangleq \frac{I_L(s)}{V_L(s)} = H(s)\,\frac{2 + e^{-as}}{3} \quad (2.29)$$

Equation (2.29), depicted in the block diagram of Fig.2.8, is the transfer function in its final form. If the applied voltage is simple enough, an analytical solution may be carried out, but in most cases a numerical solution using fast Fourier transforms or numerical Laplace transform inversion is more practical.

2.4.3 Fourier Series Techniques

Fourier series may be used to determine the load currents by means of superposition if the load of the switching circuit can be considered linear, and the voltages being applied to the load by the converter are known. The analysis begins by finding the Fourier series representation of the load voltages and the equivalent circuit of the load at each

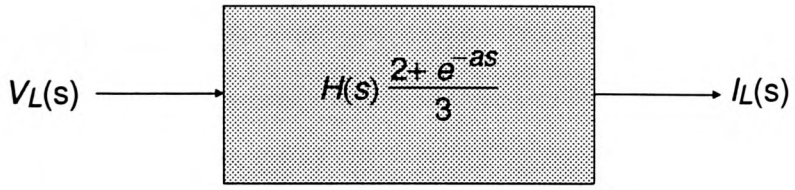
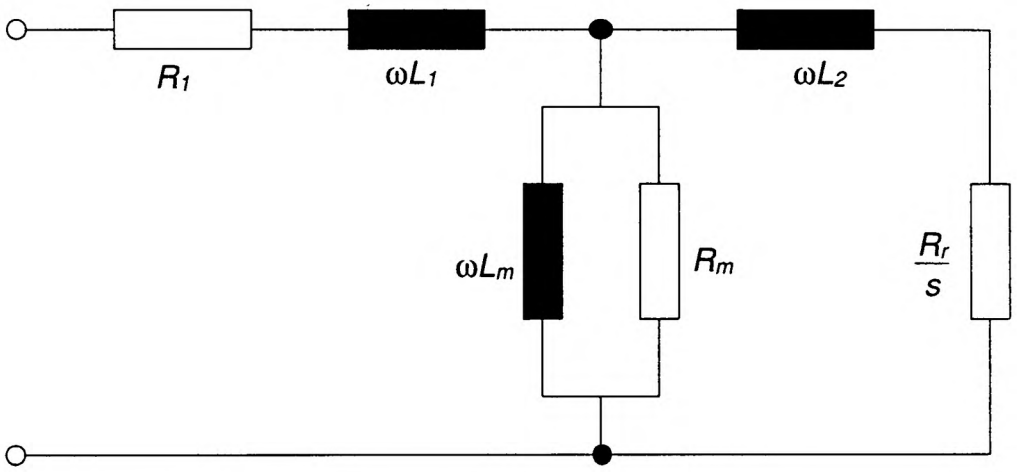
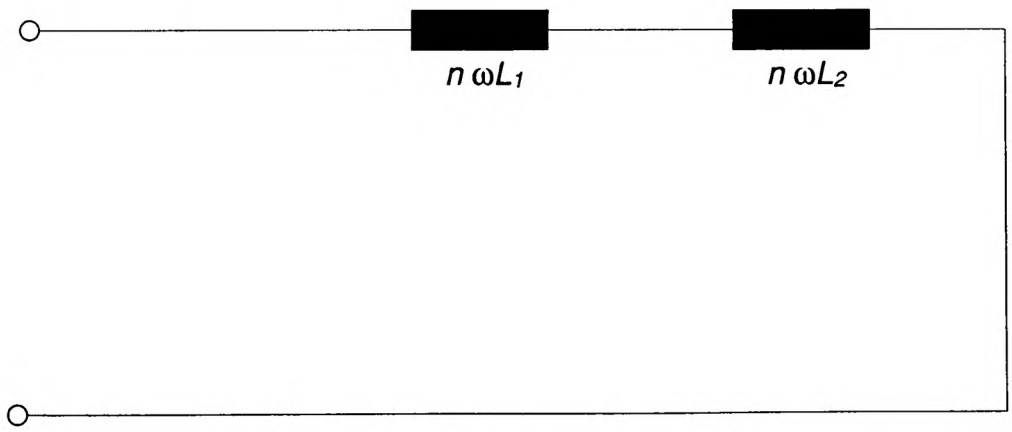


Fig.2.8 Transfer function of induction motor by Laplace Transform Method



(a) Approximate equivalent circuit for fundamental frequency



(b) Approximate equivalent circuit for the nth harmonic
Frequencies for n = 5, 7, 11, 13, ...

Fig.2.9 Analysis of induction motor by the Fourier Series Method

significant frequency in the series. The corresponding current for each harmonic voltage is found by solving the circuit that results from applying the harmonic voltage to the respective harmonic equivalent load circuit. These currents are summed to obtain the approximate load current waveform. If the load is truly linear, the load voltage is known accurately, and enough terms are retained in the series, the current waveform will be accurate. The technique may be applied to circuits in which there remains a degree of uncertainty in the voltage waveform. For example, if the thyristor extinction angle is unknown, it may be left as a variable in the Fourier series of the voltage. With Fourier series technique, the extinction angle is found by solving for the zero crossing of the resulting current waveform.

The application of Fourier series to a 3-phase, three-wire induction motor fed from a six-step bridge inverter with 180^0 device conduction is illustrated. The motor line voltages can be considered as square waves with only odd harmonics, of which only non-triplen harmonics need to be considered. The well known fundamental frequency per phase equivalent circuit of the motor is shown in Fig.2.9(a). To develop the harmonic equivalent circuit of the motor, R_2/s is neglected as slip s is high at harmonic frequencies, and the magnetizing reactance ωL_m is neglected as it is shunted by the much smaller secondary leakage reactance $n\omega L_2$, and the stator resistance R_1 is neglected in comparison with the leakage reactances at harmonic frequencies. This leads to the harmonic equivalent circuit shown in Fig.2.9(b).

Assuming that the applied stator voltage is an odd function and that C_n is the signed magnitude of the n^{th} harmonic voltage calculated in the Fourier series,

$$v_s \triangleq \sum_{n=1,5,7,11,\ldots}^{\infty} C_n \sin n\omega t \qquad (2.30)$$

Defining I_n and $\cos \varphi_n$ to be the n^{th} harmonic current amplitude and power factor,

$$I_1 = \frac{C_1}{|Z_o|} \qquad (2.31)$$

$$\cos \varphi_1 = \frac{\mathrm{Re}(Z_o)}{|Z_o|} \tag{2.32}$$

where

$$Z_o = R_{1+} j\omega L_1 + \frac{(R_2/s + j\omega L_m)\, j\omega L_m}{(R_2/s) + j\omega\,(L_2 + L_m)}$$

$$= R_{1+} j\omega L_2 + \frac{(R_2/s)\omega^2 L_m^2 + j\omega L_m\,[(R_2/s)^2 + \omega^2 L_2(L_2 + L_m)]}{(R_2/s)^2 + \omega^2\,(L_2 + L_m)^2} \tag{2.33}$$

$$I_n = \frac{C_n}{n\omega(L_1 + L_2)}, \cos\varphi_n = 0;\ n = 5, 7, 11, 13, \ldots \tag{2.34}$$

It is shown that [Jacovides,1973a] the assumption of a square wave for the applied voltage in this analysis is adequate for predicting motor performance. It is, however, necessary to consider the device turn-off and dead-time effects if accurate waveforms are to be predicted.

2.4.4 Complex Variable Analysis

It can be shown that complex variables may be applied to simplify the analysis of an inverter-fed 3-phase induction motor because of the symmetry present in the system. Complex variables reduce the steady-state voltage source inverter system problem to a second order linear system, and the current source inverter problem to a first order linear system, and closed-form solutions are amenable to either hand or simple computer calculations. In the *d-q* frame where the motor quantities are expressed in direct and quadrature axes, it is possible to use $\underline{f}$ to represent voltage, current or flux in the following complex variable form:

$$\underline{f} = \underline{f_d} + j\,\underline{f_q} \tag{2.35}$$

where $\underline{f_d}$ and $\underline{f_q}$ are quantities in direct and quadrature axes respectively. Applying equation (2.35) to the machine equation will reduce the order of the machine matrix from (4x4) to (2x2). A detailed illustration is shown in reference [Bowes,1983]. The method, however, has not been widely applied due to the apparent complexity of mathematics that could be involved in the transformation. Furthermore, there may be little gain in terms of programming and computing times when a software that can solve complex matrices efficiently is not available.

2.4.5 State Variable Analysis

2.4.5.1 Introduction

It is widely known that computers are efficient to perform repetitive task such as matrix addition and multiplication, but much less efficient for equation manipulation, Laplace transform, Fourier and the like. A major advantage of representing system equations in state variable form lies in the fact that it is systematic and thus easy to adapt for computer simulation. Moreover, whereas the classical control theory is only best suited for design of single-input single-output (SISO) linear time-invariant systems, multivariable systems having multiple input and multiple output (MIMO), which are either linear, non-linear, or time varying can best be analysed by modern control theory based on the state variable concept.

The generality of the state variable method of system representation makes it applicable to a wide range of power converter circuits. The basic principle of the state variable approach is that, given the differential equations describing a system and all the inputs $\underline{u}$ to the system, there is a minimum set of n-numbers, called the state vector $\underline{x}$, which completely determines the behaviour of the system at all times. As an illustration, if the effects of gravity and friction on a projectile are known and its position and velocity (the state variables) are known at some instant during its flight, its path can be completely determined. In general, the choice of variables for the state vector is not unique. In electrical systems, however, it is common to choose inductor currents and capacitor voltages as the state variables.

The state representation of an n^{th} order system expresses the system equations in vector form as n first-order differential equations in the following standard form:

$$\frac{dx}{dt} \triangleq \dot{\underline{x}} = A\,\underline{x} + B\,\underline{u} \tag{2.36}$$

where A and B are matrices characteristic of the system. If the desired output $\underline{y}$ of the system is different from the state vector, a second equation is used:

$$\underline{y} = C\,\underline{x} + D\,\underline{u} \tag{2.37}$$

where C and D are the appropriate matrices.

In switching circuits such as power inverters the state variable approach is complicated by the different conduction modes caused by the switching of the devices. A single circuit may therefore consist of many different systems depending on which devices conduct. Moreover, the length of conduction intervals is not always known. The usual approach is to choose a single state vector for all conduction modes and determine its solution for each mode. Continuity conditions at the switching instants are then used to piece together the solution. If the conduction intervals are unknown, iteration is normally used to solve the resulting transcendental equations. It is also suggested that [Nayak,1976] the solution time can be reduced by using the same system matrices in different modes and redefining the state and input vectors.

2.4.5.2 Application of State Space Method to an Inverter Drive System

The application of the state space method is best illustrated by an example from the reference [Lipo,1975], which uses the state space approach in the analysis of a 120^{o} conduction interval three-phase inverter, supplying an induction motor in the steady-state. As shown in Fig.2.10, the system studied consists of an ideal dc source, an LC filter, an ideal inverter and an induction machine. Due to the half-wave and three-phase symmetry of the current waveform, the steady-state analysis only needs to extend over one-sixth of a cycle, as will be shown. However, current is discontinuous

when the gating period is 120^{o}. This means that two modes are present during the interval studied.

Some of the variables are defined in Fig.2.10 and Fig. 2.11. Fig. 2.12 shows the equivalent circuits of the system during the two modes: mode 1, the beginning of positive conduction in phase *A*; and mode 2, the zero current interval for phase *A*. The *d-q* equations (2.14) of the motor, where v_{qr} and v_{dr} are set to zero, may be rewritten as follows:

$$\begin{bmatrix} v_{qs}^{s} \\ v_{ds}^{s} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_s + L_s * p & 0 & L_m * p & 0 \\ 0 & R_s + L_s * p & 0 & L_m \\ L_m * p & -\omega_r * L_m & R_r + L_r * p & -\omega_r * L_r \\ \omega_r * L_m & L_m * p & \omega_r * L_r & R_r + L_r * p \end{bmatrix} * \begin{bmatrix} i_{qs}^{s} \\ i_{ds}^{s} \\ i_{qr}^{s} \\ i_{dr}^{s} \end{bmatrix} \qquad (2.38)$$

The superscript *s* of equation (2.38) denotes the quantity being referred to be stator. The *q*-axis is assumed coincident with the phase *A*-axis for the stator and therefore based on the absence of zero-sequence quantities, relations between phase and *d-q* variables are as given in the following two equations:

$$v_{qs}^{s} = v_{as}, \quad v_{ds}^{s} = \frac{1}{\sqrt{3}} * (v_{cs} - v_{bs}) \qquad (3.39a)$$

$$i_{qs}^{s} = i_{as}, \quad i_{ds}^{s} = \frac{1}{\sqrt{3}} * (i_{cs} - i_{bs}) \qquad (3.39b)$$

The current and voltage equations for the filter illustrated in Fig.2.12 are as follows:

$$i_l + p * C * v_C - i_R = 0 \qquad (2.40a)$$

$$V_R = (r_L + p * L) * i_R + v_C + r_C * (i_R - i_l) \qquad (2.40b)$$

During model 1, when current flows in all three phases of the induction motor, the system equivalent circuit is as shown in Fig.2.12(b), from which the voltage and current equations can be obtained:

$$v_{as} - v_{bs} = v_l, \quad v_{bs} = v_{cs}, \quad i_l = i_{as} \qquad (2.41a)$$

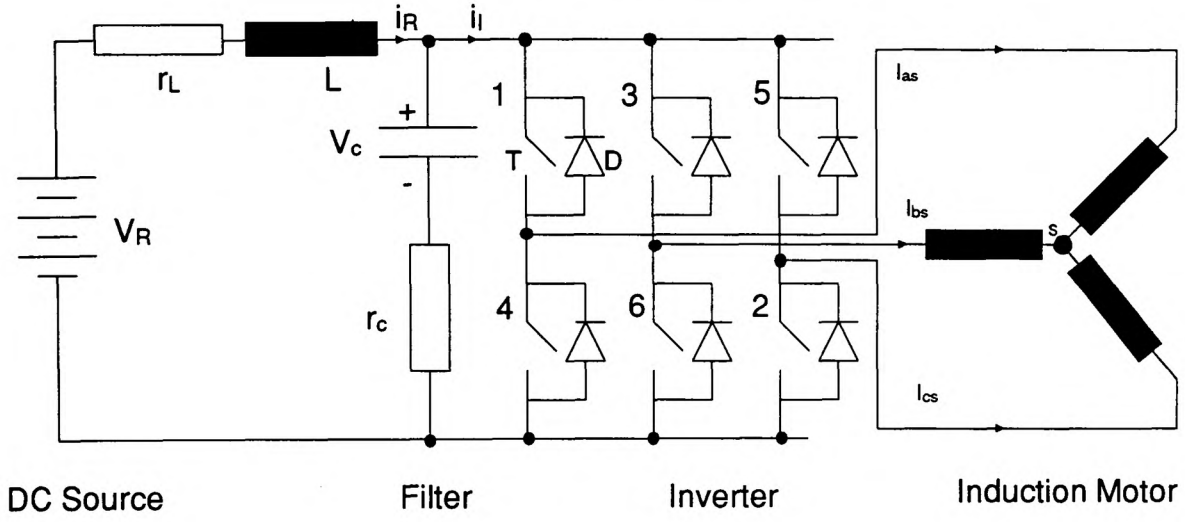


Fig.2.10 Induction motor drive system

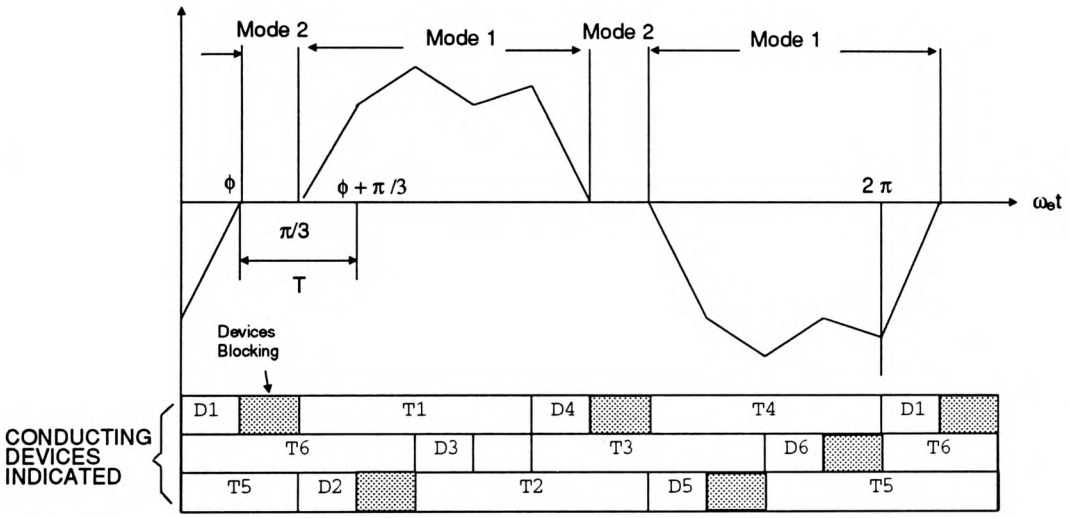


Fig.2.11 Typical steady-state line current waveform and inverter operation

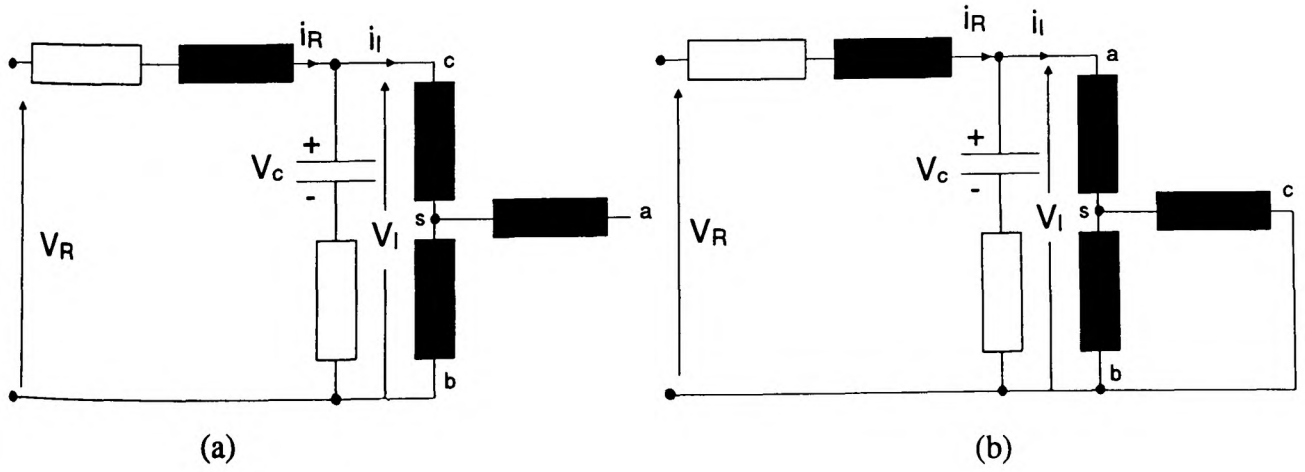


Fig.2.12 Circuit modes
(a) Mode 2, valid for $0 < \omega t < \pi/3$, (b) mode 1, valid for $\pi/3 < \omega t < \pi/3 + \phi$

$$v_{qs}^{s} = \frac{2}{3}*v_l = \frac{2}{3}*[v_c + r_c*(i_R - i_l)], \quad v_{ds}^{s} = 0, \quad i_l = i_{qs}^{s} \tag{2.41b}$$

Combining equations (2.38), (2.40), and (2.41) gives the following:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_R \end{bmatrix} = \begin{bmatrix} r_s + 2/3*r_c + L_s*p & 0 & L_m*p & -2/3 & -2/3*r_c \\ 0 & r_s + L_s*p & o & 0 & 0 \\ L_m*p & -\omega_r*L_m & r_r + L_r*p & 0 & 0 \\ \omega_r*L_m & L_m*p & \omega_r*L_r & 0 & 0 \\ 1 & 0 & 0 & p*C & -1 \\ -r_c & 0 & 0 & 1 & r_L + r_c + p*L \end{bmatrix} * \begin{bmatrix} i_{qs}^{s} \\ i_{ds}^{s} \\ i_{qr}^{s} \\ i_{dr}^{s} \\ v_c \\ i_R \end{bmatrix} \tag{2.42}$$

When equation (2.41) is written in standard state form, it becomes as follows:

$$\frac{dx}{dt} = A_1 \underline{x} + B_1 \underline{u} \tag{2.43}$$

where the quatities x, u, B_1 and A_1 are given as:

$$\underline{x} = \begin{bmatrix} i_{qs}^{s} \\ i_{ds}^{s} \\ i_{qr}^{s} \\ i_{dr}^{s} \\ v_c \\ i_R \end{bmatrix}, \quad \underline{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_R \end{bmatrix}, \quad B_1 = \begin{bmatrix} L_s & 0 & L_m & 0 & 0 & 0 \\ 0 & L_s & 0 & L_m & 0 & 0 \\ L_m & 0 & L_r & 0 & 0 & 0 \\ 0 & L_m & 0 & L_r & 0 & 0 \\ 0 & 0 & 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 & 0 & L \end{bmatrix}$$

$$A_1 = -B_1 \begin{bmatrix} r_s + 2/3 r_c & 0 & 0 & 0 & -2/3 & -2/3 r_c \\ 0 & r_s & 0 & 0 & 0 & 0 \\ 0 & \omega_r L_m & r_r & -\omega_r L_r & 0 & 0 \\ \omega_r L_m & 0 & \omega_r L_r & r_r & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ -r_c & 0 & 0 & 0 & 1 & r_L + r_c \end{bmatrix}$$

During mode 2, when $i_{as} = 0$, as shown in Fig.2.11, and the equivalent circuit for the motor drive system given in Fig 2.12(a), the voltage and current equations are as follows:

$$v_{cs} - v_{bs} = v_l \tag{2.44a}$$

$$i_l = i_{cs} = -i_{bs} \tag{2.44b}$$

Combining equations (2.44) and (2.39) gives the following equations for V_l and i_l in the d-q axis terms:

$$v_{ds}^{s} = \frac{1}{\sqrt{3}} * v_l, \quad v_{qs}^{s} = L_m * p * i_{qr}^{s} \tag{2.45a}$$

$$i_{qs}^{s} = 0 \tag{2.45b}$$

$$i_l = -\frac{1}{2} i_{qs}^{s} + \frac{\sqrt{3}}{2} i_{ds}^{s} \tag{2.45c}$$

Combining equations (2.39), (2.41), and (2.45) and eliminating the terms V_l and i_l results into the matrix equation:

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_R \end{bmatrix} = \begin{bmatrix} r_s+L_sp & 0 & 0 & 0 & 0 & 0 \\ -(\sqrt{3}/2)r_c & r_s+r_c/2+L_sp & 0 & L_mp & -1/\sqrt{3} & -r_c/\sqrt{3} \\ L_mp & -\omega_r L_m & r_r'+L_r'p & -\omega_r L_r' & 0 & 0 \\ \omega_r L_m & L_mp & \omega_r L_r' & r_r'+L_r'p & 0 & 0 \\ -1/2 & \sqrt{3}/2 & 0 & 0 & pC & -1 \\ 1/2\, r_c & -(\sqrt{3}/2)r_c & 0 & 0 & 1 & r_L+r_c+pL \end{bmatrix} \begin{bmatrix} i_{qs}^{s} \\ i_{ds}^{s} \\ i_{qr}^{s} \\ i_{dr}^{s} \\ v_c \\ i_R \end{bmatrix} \tag{2.46}$$

Equation (2.46) may be represented in standard state space form as:

$$\frac{dx}{dt} = \dot{\underline{x}} = A_2\underline{x} + B_2\underline{u} \tag{2.47}$$

where the quantities B_2 and A_2 are given as follows:

$$B_2 = \begin{bmatrix} L_s & 0 & 0 & 0 & 0 & 0 \\ 0 & L_s & 0 & L_m & 0 & 0 \\ L_m & 0 & L_r' & 0 & 0 & 0 \\ 0 & L_m & 0 & L_r' & 0 & 0 \\ 0 & 0 & 0 & 0 & C & 0 \\ 0 & 0 & 0 & 0 & 0 & L \end{bmatrix}^{-1} \quad \text{and}$$

$$A_2 = -B_2 \begin{bmatrix} r_2 & 0 & 0 & 0 & 0 & 0 \\ -(\sqrt{3}/2)r_c & r_s + r_c/2 & 0 & 0 & -1/\sqrt{3} & -r_c/\sqrt{3} \\ 0 & -\omega_r L_m & r_r' & -\omega_r L_r' & 0 & 0 \\ \omega_r L_m & 0 & \omega_r L_r' & r_r' & 0 & 0 \\ -1/2 & \sqrt{3}/2 & 0 & 0 & 0 & -1 \\ 1/2 r_c & -(\sqrt{3}/2)r_c & 0 & 0 & 1 & r_L + r_c \end{bmatrix}$$

The solution to the state variable equation (2.47) during the period of $\frac{\varphi}{\omega_e} < t < \frac{\pi}{(3\omega_e)}$ is as follows:

$$\underline{x}(t) = e^{A_2 (t - \varphi/\omega_e)} \underline{x} (\varphi/\omega_e) + \lambda_2 (t, \varphi/\omega_e) B_2 \underline{u} \qquad (2.48)$$

where

$$\lambda_2(t, \varphi/\omega_e) = \int_{\varphi/\omega_e}^{t} e^{A_2 (t-\tau)} d\tau = A_2^{-1} (e^{A_2(t - \varphi/\omega_e)} - 1)$$

Similarly, the solution of state space equation (2.43) during the period $\frac{\pi}{(3\omega_e)} < t < \frac{\varphi}{\omega_e} + \frac{\pi}{(3\omega_e)}$ is given by:

$$\underline{x}(t) = e^{A_1 (t - \pi/3\omega_e)} \underline{x} (\pi/3\omega_e) + \lambda_1 (t, \pi/3\omega_e) B_1 \underline{u} \qquad (2.49)$$

where

$$\lambda_1(t, \frac{\pi}{3\omega_e}) = A_1^{-1} (e^{A_1(t - \pi/3\omega_e)} - 1)$$

The substitution with $t = \frac{\pi}{3\omega_e}$ into equation (2.48), and with $t = \frac{\varphi}{\omega_e} + \frac{\pi}{3\omega_e}$ into (2.49) establishes a relationship, which, after simplification, provides a further relationship between $\underline{x} (\frac{\varphi}{\omega_e})$ and $\underline{x} (\frac{3\varphi + \pi}{3\omega_e})$, as given below:

$$\underline{x} (\varphi/\omega_e + \pi/3\omega_e) = e^{A_1 \varphi/\omega_e} e^{A_2 (\pi/3\omega_e - \varphi/\omega_e)} \underline{x} (\varphi/\omega_e)$$
$$+ [e^{A_1 \varphi/\omega_e} \lambda_2 (\pi/3\omega_e, \varphi/\omega_e) B_2 + \lambda_1(\pi/\omega_e + \pi/3\omega_e , \pi/3\omega_e) B_1] \underline{u} \qquad (2.50)$$

Due to three-phase and half-wave symmetry of the waveforms in steady state, as illustrated in Fig.2.11, only one-sixth of a complete cycle needs to be solved in detail, and a solution over this interval completely defines all variables over an entire cycle. If S is the connection matrix responsible for the symmetry, it follows that [Luk,1991a]:

$$\underline{x}(\varphi/\omega_e + \pi/3\omega_e) = S\,\underline{x}(\varphi/\omega_e). \tag{2.51}$$

The unknown initial condition vector $\underline{x}(\varphi/\omega_e)$ is now found by combining (2.50) and (2.51) as follows:

$$\underline{x}(\varphi/\omega_e) = [S - e^{A_1\varphi/\omega_e}\, e^{A_2(\pi/3\omega_e - \varphi/\omega_e)}]^{-1}\; W \tag{2.52}$$

where $W = [e^{A_1\pi/\omega_e}\,\lambda_2(\varphi/3\omega_e, \varphi/\omega_e)\,B_2 + \lambda_1(\varphi/\omega_e + \pi/3\omega_e, \pi/3\omega_e)\,B_1]\,\underline{u}$

The only unknown quantity on the right side of equation (2.52) at a given steady-state rotor speed is φ, the extinction angle shown in Fig.2.11, which is known to lie in the range $0< \varphi <\pi/3$ for 120^0 inverter gating. The fact that $i_{qs}^s(\varphi/\omega_e)$ must be zero on the left side of equation (2.52) provides a criterion for iterating to the proper value of φ. Once $x(\varphi/\omega_e)$ is known, equations (2.48) and (2.49) may be used to form the complete waveforms during any period.

2.4.5.3 Simulation of Inverter Drive System Using State Variable Method

The state space expressions derived in the previous section for the waveforms of the 3-phase induction motor drive were translated into program codes. The VAX-based simulation software package used, called 'Control-C', was conveniently available and was adopted in the simulation. 'Control-C' was initially intended for the analysis of control systems but proved particularly adaptable to the investigation described in this thesis. The distinct facility of the software package in handling data exclusively in matrix forms greatly reduced both the programming time and debugging time because machine equations in state space form could be entered directly, and wrongly entered data could easily be identified.

To test the digital model written in 'Control-C', the same machine data from [Lipo,1975] was used, so that the simulation result could be compared with the verified results from Lipo's work. Typical simulated results for current, line voltage, phase voltage, and the torque over one electrical cycle are shown in Fig.2.13 to Fig.2.16. These results compare favourably with those obtained by Lipo. It is, however, not

possible to compare the present model in 'Control-C' with that of Lipo's in terms of processing power required for the solution, since adequate information regarding computational time is not available from the latter. It is strongly believed that the elimination of complex algebraic equations in formulating the present system drive model has resulted in a major reduction in computational power required for the solution. Another important result in such a simplification in modelling a drive system is that it enables motor drive design engineers to develop the machine model with a minimum of mathematics involved.

2.4.6 Other Methods

There are some other analysis methods which are not as common as those discussed in the previous sections but prove particular useful in some applications. The switching function analysis, for example, is based on the repetition in which the switching devices in a power converter are switched between conducting and blocking states. Such an example is clearly presented in reference [Novotny,1975], where 36 equations describing the voltage and current transfer relationships of a six-step inverter are reduced to 2 or 3 equations by taking advantage of the symmetry of the waveform and redundancy that is present in the 36 equations. The method, however, has not been fully explored according to the literature available. A further alternative method [Dewan,1970] makes use of 'Boolean Algebra' to model the switching device as a binary device, where a binary '1' represents a conduction state, and a binary '0' represents a blocking state. A number of other methods have also been presented in the literature [Jardan,1969;Jacovides,1973b] but have not proved useful for wider applications.

2.5 INTERIM CONCLUSION

This chapter has served to provide the fundamental basis for the analysis, simulation and experimental techniques used in the investigation. These simulation and experimental techniques will be further elaborated upon during later chapters of this

thesis. The simulation technique adopted was applied to an electrical drive system initially presented by Lipo and the results obtained were found to be in good agreement with the results presented in the reference [Lipo,1975]. The good agreement between these results suggested that the system to be investigated and reported upon in the thesis could also be simulated by the technique described.

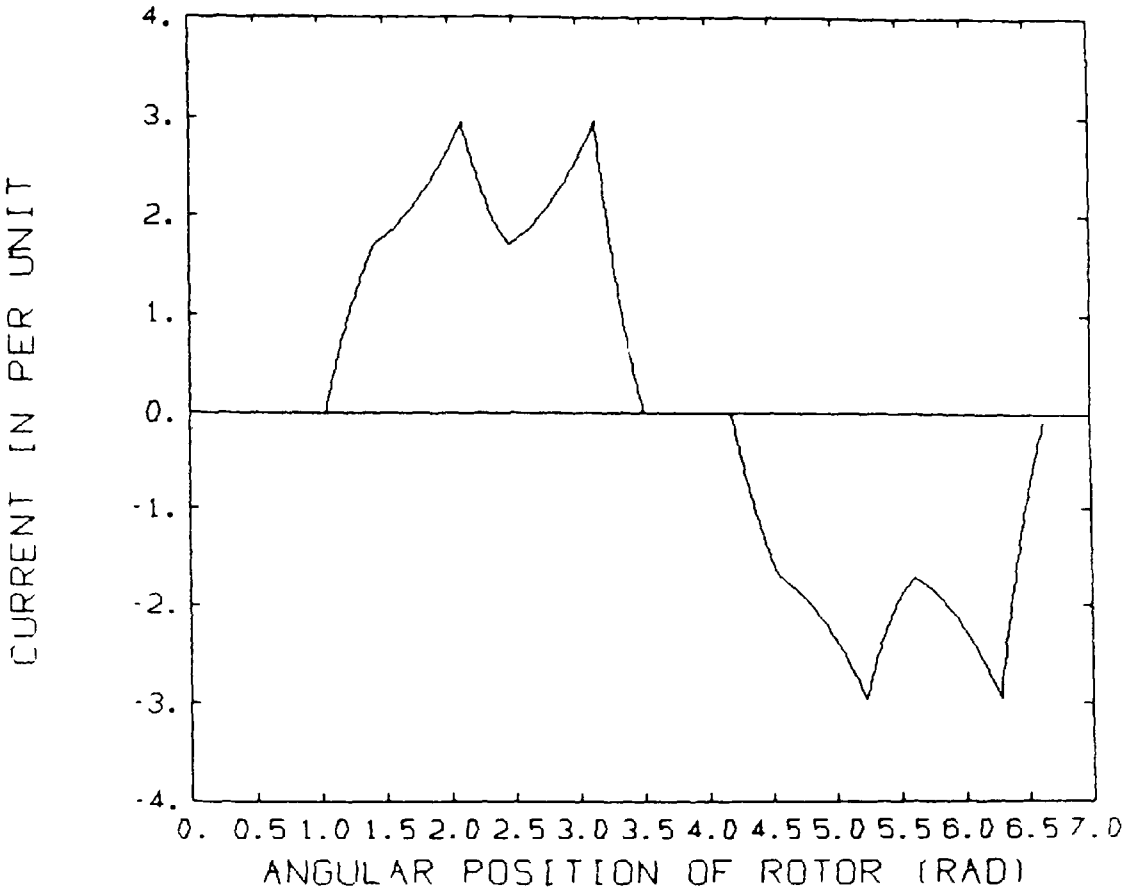


Fig.2.13 Motor line current

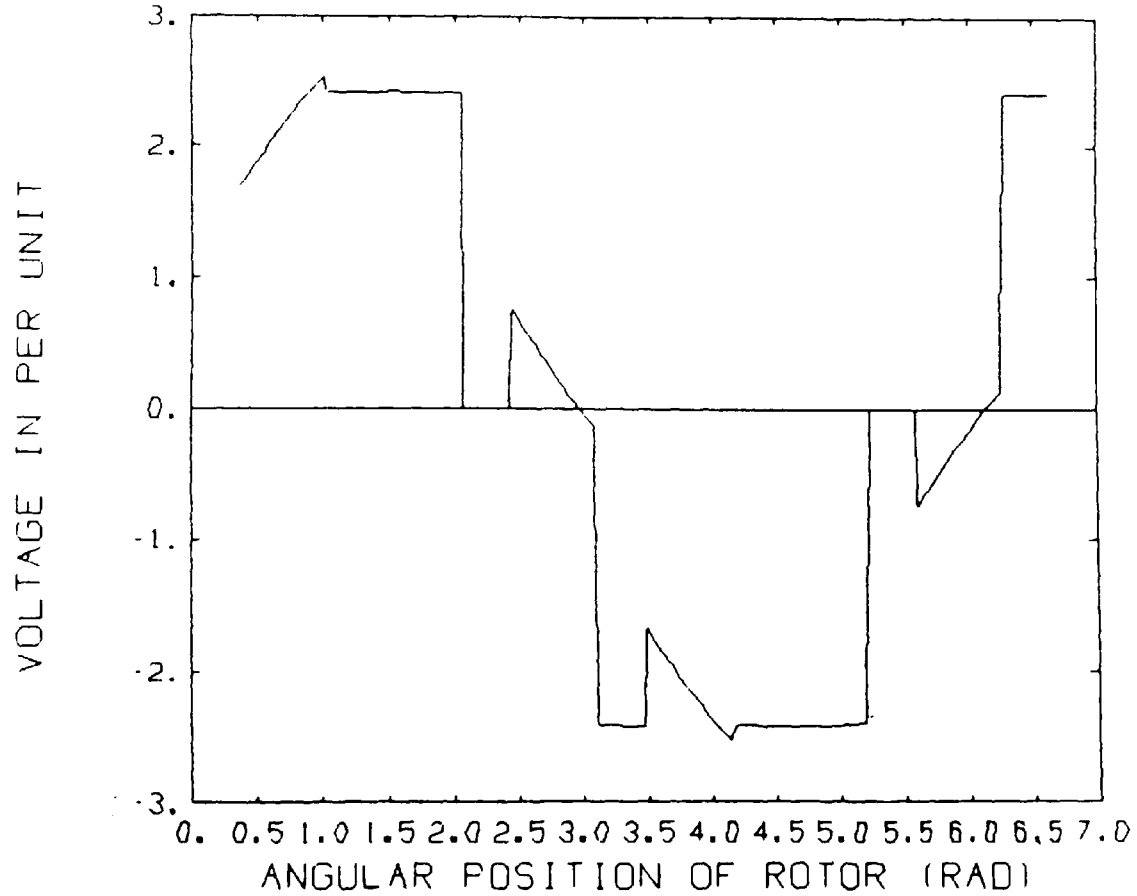


Fig.2.14 Motor line voltage

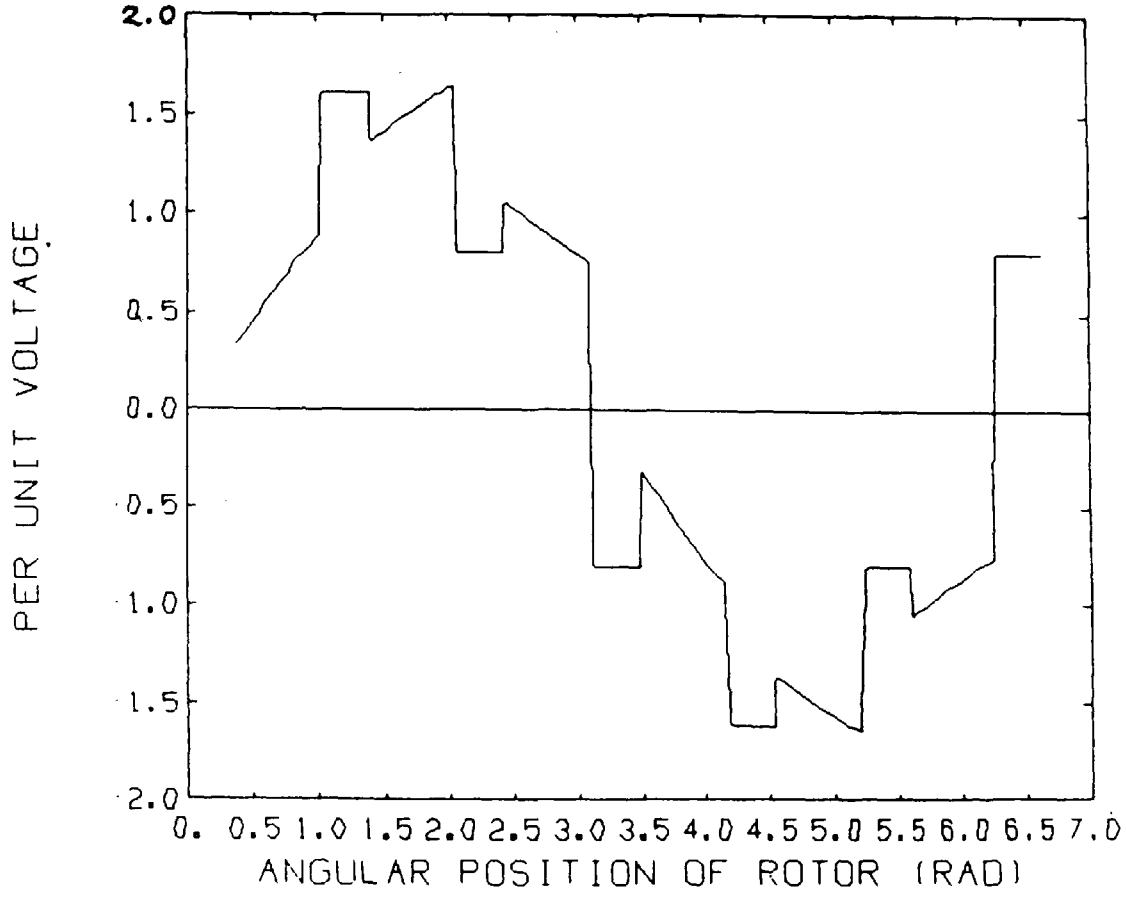


Fig.2.15 Motor phase voltage

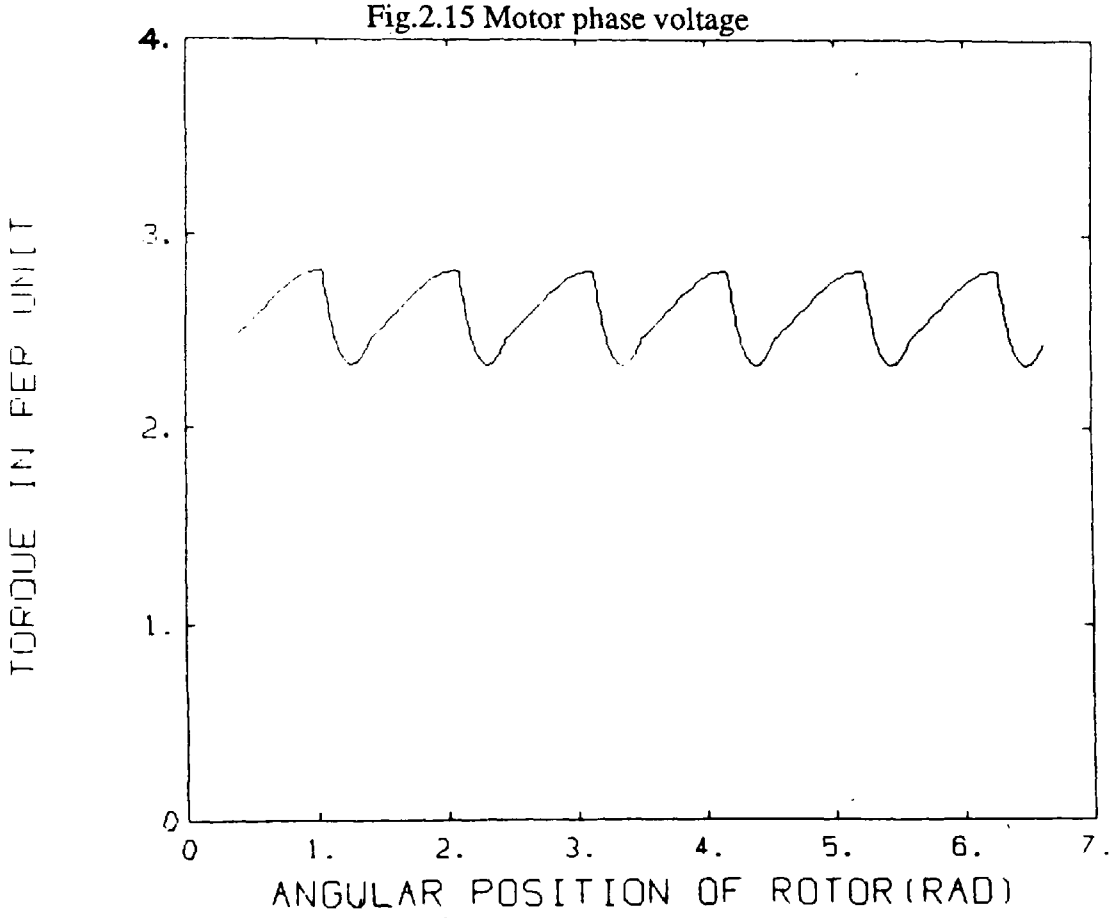


Fig.2.16 Electromagnetic torque

Chapter 3 : Principles of Variable Speed Induction Motor Control System

3.1 INTRODUCTION

The three-phase squirrel cage induction machine is the most common form of AC electric motor used in industry because of its low cost and high reliability. The recent advances in microelectronics and power electronics technology have enabled this traditionally constant speed machine to be converted, cost-effectively, into a variable speed actuator or drive. The ever-increasing demand for high performance electrical drives has resulted in the application of sophisticated control strategies such as vector control, adaptive control and sliding mode control techniques. The scope of the study of variable speed induction motor drive, as a result, has now extended from such traditional methods as 'inserted rotor resistance', 'voltage and frequency control' by means of static converters, to more complex methods including flux acquisition, transformation of reference frame and other functions.

The range of electrical variable speed drives available is quite extensive and cannot be commented upon in this thesis. A thorough review, however, was carried out by Jones and Brown [Jones,1984]. This chapter focuses on the requirements of variable speed drive systems. It is shown that these requirements can be partly, and sometimes wholly, satisfied by such control technique as pulse-width modulation (PWM) employing natural sampling and regular asymmetric sampling. Where dynamic response of the electrical drive has become a significant requirement, it is shown that 'vector control' can be usefully applied. The high computing speed requirement that can be imposed by the method of real-time PWM generation considered is greatly enlarged upon, and is shown that the transputer can satisfy such imposition. The derivation of the vector control strategy by the end of this chapter, which is mainly based on the work of Harashima [Harashima,1985], forms the theoretical basis of the practical implementation of the strategy described later in chapter 7.

3.2. PRINCIPLES OF VARIABLE SPEED INDUCTION MOTOR DRIVES

3.2.1 Torque-speed Characteristic of an Induction Motor

The equivalent circuit of an induction motor offers a good starting point for establishing the theory of a variable speed drive, and it is illustrated in Fig.3.1(a). A modified equivalent circuit [Steven,1983], shown in Fig.3.1(b), proves to be useful in the analysis by further simplification of the circuit, in which R_m is removed and grouped with the motor friction and windage losses, and the magnetising reactance is being included in the

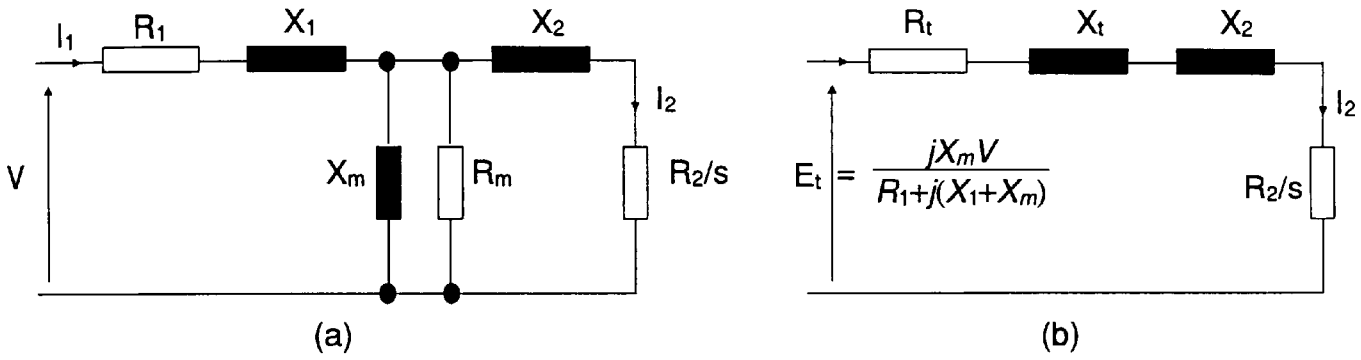


Fig.3.1 Equivalent circuit of an inductance motor
(a) Normal (b) Transformed

Thevenin impedance denoted by subscript *t*. The equivalent circuit may be used to develop the following torque equation for the motor:

$$T = \frac{mE_t^2 R_2}{(\omega_o - \omega_r)\left[\left(R_t + \frac{\omega_o R_2}{\omega_o - \omega_r}\right)^2 + (X_t + X_2)^2\right]} \qquad (3.1)$$

where m is the number of phases, ω_o is the synchronous speed, ω_r is the rotor speed, and E_t is the Thevenin's equivalent source voltage shown in Fig.3.1(b).

Since the operation of a variable speed induction motor drive may be considered in terms of controlling the motor so that it delivers a given torque at a particular speed, equation (3.1) may be rearranged such that it becomes a quadratic equation of ω_r when other quantities are assumed constant. The speed(s) at a given torque is found by solving this quadratic equation as follows:

$$\omega_r = \omega_o - \left(\frac{R_2}{2T(R_t^2 + X_t + X_2)^2}\right)(mE_t^2 - 2R_t\omega_o T \pm\sqrt{m^2E_t^4 - 2\omega_o T(R_t mE_t^2 + \omega_o T(X_t + X_2)^2)}) \quad (3.2)$$

From equation (3.2), it is apparent that the following methods of altering the speed of the motor may be applied.

(a) Altering the supply voltage. This results in a change in E_t and therefore a change in speed.

(b) Connecting impedance in series with the supply and the stator winding. Such an impedance is therefore in series with the stator winding R_1 and X_1 and the resulting R_t and X_t is the transformed equivalent circuit. Consequently, from Fig.3.1(b), altering this series impedance will affect E_t, and the motor speed will be altered according to equation (3.2).

(c) Altering the rotor resistance. Such a method results in a change in speed according to equation (3.2) but only available to wound-rotor machines.

(d) Altering the synchronous speed of the motor. This may be done by changing the number of pole pairs, np, or by changing the supply frequency ω_o.

The insertion of impedance of method (b) is, in effect, similar to reducing voltage supply. However, the resulting reduction in supply voltage is current dependent and therefore load-dependent. This complicates the control of the drive and introduces ohmic power loss in the inserted resistance. Method (c) is not applicable to squirrel-cage rotor induction motor and is therefore not further discussed. Although the method of changing the number of pole pairs results in good efficiency and power factor, it can only produce several discrete changes in speed. Thus, it is only the methods of altering the magnitude and frequency of the applied stator voltage that will be considered further in this thesis.

3.2.2 Variable Voltage, Variable Frequency (VVVF) Power Source

It is important to note that reducing the amplitude of supply voltage without a proportional reduction in supply frequency can lead to magnetic saturation. Allowing for

the voltage drop across the stator winding, the relationship between the flux Φ, the applied voltage V and frequency ω is given approximately by the relationship:

$$\Phi = constant * \frac{V}{\omega}$$

This relationship shows that the motor flux Φ increases as the motor speed ω decreases and in such circumstances the supply voltage is reduced to avoid magnetic saturation. Consequently most schemes which control induction motor speed by altering the supply frequency also control the amplitude of the supply voltage to avoid magnetic saturation. The supply to the induction motor is therefore a variable voltage, variable frequency (VVVF) supply. The different types of VVVF supply that have been used are:

(a) Motor-Generators (M-G) set
(b) Cycloconverters
(c) DC link inverters

In the early years, variable speed motors were used to drive AC generators to produce a VVVF supply suitable for induction motor speed control, as shown in Fig.3.2. Such motor-generator sets tended to be noisy, bulky, and required frequent maintenance. As a result of the introduction of the thyristor and other switching devices, the M-G set became redundant so far as producing VVVF supplies is concerned.

The cycloconverter can achieve a direct AC to AC conversion without the necessity of an intermediate DC link. In practice, a three-phase to six-phase transformer is required in order that the ripple amplitude on the output waveforms is reduced to a reasonable level. This requires the use of thirty-six power semiconductor devices, as compared to twelve in other bridge units, in the power conversion circuit. Due to the harmonic distortion at higher output frequencies, the cycloconverter is limited to producing frequency in the range of 1 Hz to one third of the supply frequency. The commutation, however, can be achieved naturally by the polarity reversal of the input AC waveform. A fully regenerative operation is also possible. Due to the high initial cost and low frequency limitations, the main applications of the cycloconverter are found in controlling very large motors or in controlling multi-motor stations as required in steel works.

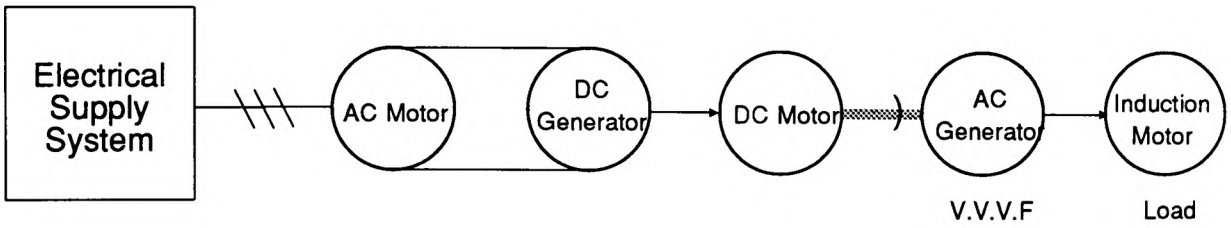


Fig.3.2 Motor-Generator (M-G) set

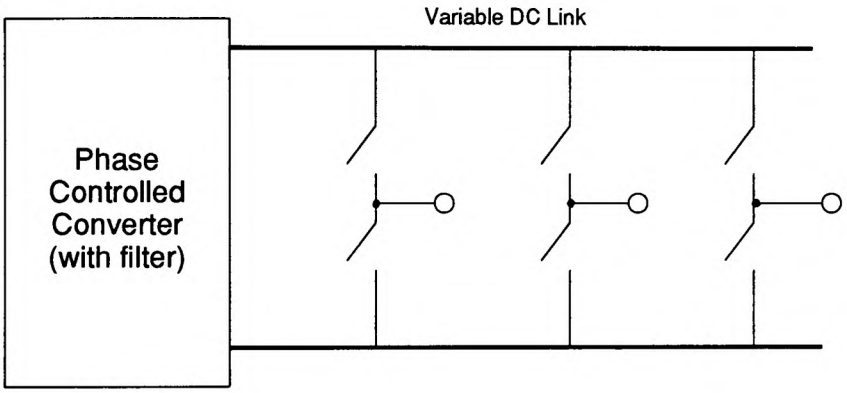


Fig.3.3 Six-Step (Quasi-Square) wave inverter

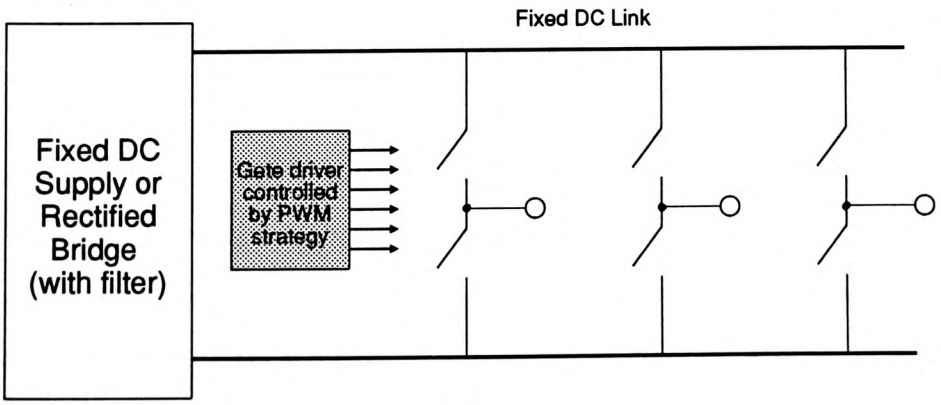


Fig.3.4 PWM inverter

The DC link inverter may be divided into two types by way of the voltage waveforms produced. The most simple type is known as six- step or quasi-square wave, and is shown in Fig.3.3. The top and bottom devices of each leg are switched on alternatively for 180 or 120 degrees of the cycle, and the three legs are switched with 120 degree displacement between them. The DC link voltage is usually controlled by a phase-controlled thyristor converter with an LC filter on the output. The filter introduces a time lag that degrades the transient response of the system. Alternatively, the DC voltage can be regulated by means of a DC chopper circuit.

The other type is the pulse width modulation (PWM) inverters, and is shown in Fig.3.4. The devices in a leg switch on alternatively at a frequency usually much higher than the frequency of the motor. The on and off periods of the devices for a leg are varied in a pattern determined by the PWM strategy adopted. This modulation is repeated in the other two legs with a 120 degree displacement. Due to a wide variety of PWM strategies available for the inverter and their application in the majority of present day DC link inverters, the PWM strategies are discussed in more detail in the following section.

3.3 PWM SWITCHING STRATEGIES

3.3.1 Overview of the Development of Switching Strategies

With the increasing availability of high speed power devices since the last two decades, different switching strategies were considered for implementation of a VVVF power source. One of the strategies that has since proved successful is the pulse width modulation (PWM). The main advantage of PWM over other techniques, such as pulse amplitude modulation (PAM), is the output voltage and frequency of the power converted can be realized in one modulation process. PWM also offers the advantage of controlling the harmonic spectrum of the output voltage waveform. This additional advantage can be realized by using two or three-level PWM, pulse number, and in certain cases, the sampling technique. Additional techniques, although not considered pure PWM, have also proved attractive for harmonic spectrum control. For example, a technique known as

'notch control' has gained much favour over the last two decades for controlling harmonic spectrum of the output waveform.

In the early 70's, initially well established modulation techniques of natural sampling, suitable for analogue implementation, were used [Jayne,1976]. As the technology changed towards digital, look-up table methods [Dwyer,1979] in which pre-determined switching angles were stored in memory were then developed and used in digital systems. With the advances in microprocessor/microcontroller technology in the mid 70's PWM strategies such as regular sampling, suitable for digital implementation, were developed. The inadequate processing speed of the microprocessors had precluded any contemplation of generating advanced PWM waveforms in real-time. On the other hand, as increasingly memory is needed to generate a wide range of speed by the memory-mapping method, the cost of the system increases dramatically. The rapid development in recent years in microelectronics has meant that the cost ratio of processor to memory devices is dropping steadily. Therefore, the applications of multi-microprocessors [Harashima,1985] and the more recently introduced signal processors [Buja,1985] to further improve the control of PWM inverters have been reported. As switching angles are computed on line in these systems, steady state as well as transient operating conditions are satisfactorily dealt with.

In the early 70's, the majority of PWM switching strategies that applied to DC link inverter systems were realized by analogue means. It was shown that PWM process employed in these techniques contained natural sampling, that is to say, the instantaneous intersection of the carrier wave and modulating wave were used as the switching instant for the power devices. An additional sampling technique known as regular sampling, was also applied in analogue PWM systems [Jayne,1976]. This additional technique, which consisted of taking regular samples of the modulating waveform, proved quite successful mainly because: (a) it reduced the harmonic distortion in the output for low value of frequency ratio or pulse number when compared with the natural sampling method, and (b) it was easier to implement in digital PWM control systems employing microprocessors. However, at that time the main limiting factor was the slow processing speed of the microprocessor chips. The real-time implementation of PWM strategy required much faster digital processing chips than was available at that time.

One important result of the development of the power devices is the increase of switching speed. The usual timing-related limitations imposed by the power devices, such as the maximum and minimum pulse widths and modulation frequency, as a result, have been greatly relaxed. Ultrasonic PWM inverters using switching frequencies of 20 kHz or above have been reported in special cases [Grant,1983]. At present, a variety of sampling methods supplemented by methods for selected harmonic elimination and a host of suboptimal methods are available. These methods, together with some emerging non-conventional approaches, will be discussed in the following subsections.

3.3.2 Natural sampled PWM

This method is a well-established modulation method used in communication engineering. Early developments in PWM power inverter technique centered on this method of sampling because it is intrinsically analogue and can be easily implemented by the use of analogue comparators. The generation of natural sampled PWM is illustrated in Fig.3.5. The switching edges of the width-modulated pulses, which correspond with the sampling instants, may be expressed by the transcendental equation:

$$t_1 - t_2 = \frac{T_c}{2}\left[1 + \frac{M}{2}(\sin\omega t_1 + \sin\omega t_2)\right] \qquad (3.3)$$

where T_c is the carrier period, M is the modulating index, and ω is the angular frequency of the modulating wave.

The solving of equation (3.3) by means of a digital computer is usually unsuitable because it involves a number of iterations and therefore makes this method of PWM sampling unrealistic to implement in real-time. Where this method is used, the switching angles of the PWM waveform for different fundamental output voltages are computed 'off-line' and stored in a look-up table by a computer. Alternatively, the carrier and modulating waves can be synthesed by a microprocessor by means of digital-to-analog (D-A) converters, and the comparison of the two generated waveforms is performed by means of an analogue comparator. A thorough description of the implementation of the scheme by means of microprocessor techniques is found in the reference [Bowes,1981]. More recently,

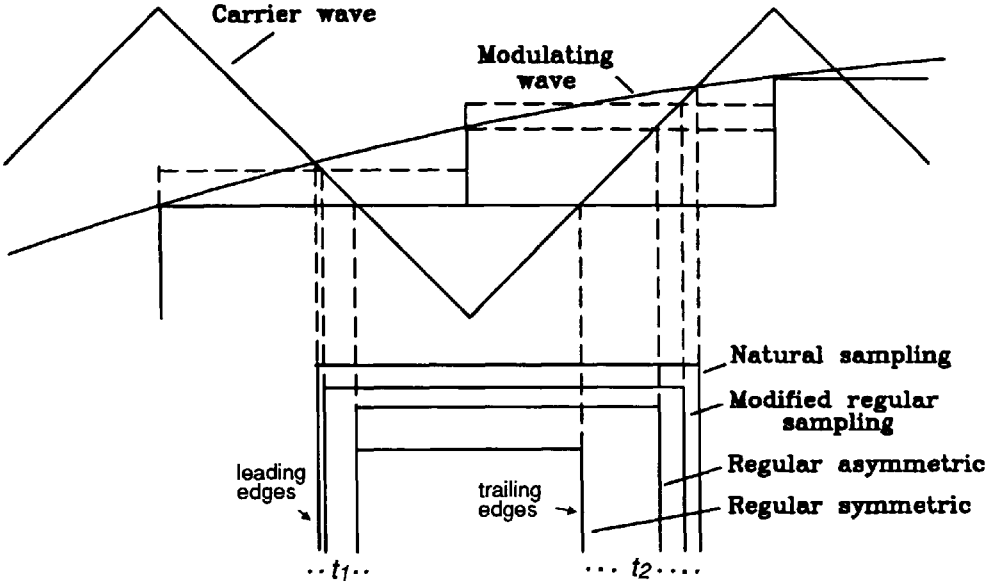


Fig.3.5 Common PWM sampling strategies

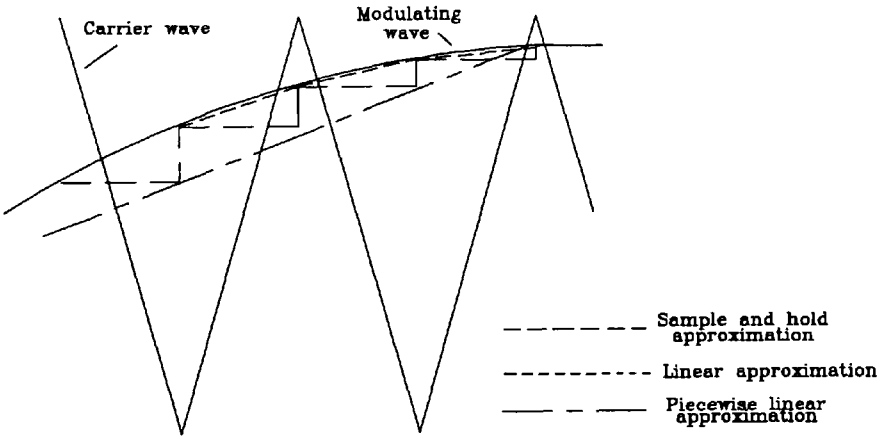


Fig.3.6 Sinewave approximation sampling methods

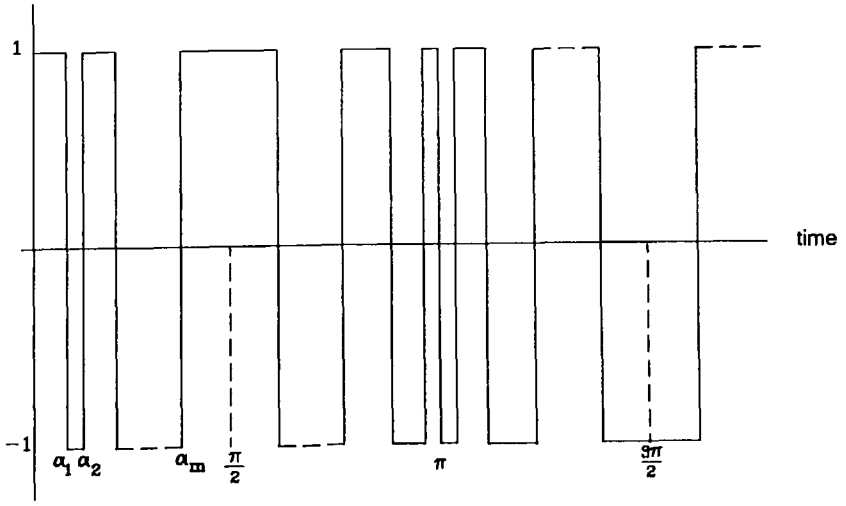


Fig.3.7 Generalized quarter-wave symmetric PWM waveform

however, it has been reported [Khanniche,1988] that the method can be implemented 'on line' by an algorithm that was based on piecewise linearisation of the switching strategy.

3.3.3 Regular Sampled Asymmetric PWM

This method of modulation has also been well-known in the communication industry before being applied to PWM inverters. The generation of the modulated pulses is again illustrated in Fig.3.5. Two types of regular sampling PWM strategies have emerged, the symmetric and asymmetric. The asymmetric sampled strategy is spectrally superior as the number of samples is twice that of the symmetric sampled strategies. Unlike natural sampling, regular sampling is inherently a digital method in which the magnitude of regularly spaced samples of the modulating wave determines the widths of the modulation pulses. The width of the n^{th} pulse of a regular sampled asymmetric PWM waveform is defined by the following analytical expression:

$$T_n = \frac{T_c}{2}[\,1 + (-1)^{n+1}\frac{M}{2}(\sin A_n + \sin A_{n+1})] \qquad (3.4)$$

where T_c is the carrier period, $A_n = \frac{n\pi}{R}$, and R is the frequency ratio of the carrier wave to modulating wave. This analytical expression for the pulse-widths can be readily implemented in real-time using a digital control system.

3.3.4 Modified PWM Strategies

A number of modified schemes based on the conventional regular sampled asymmetric strategy have been developed. These schemes aim at extending the output frequency range and improving the linearity of output voltage with increase in modulation depth, whilst at the time maintaining lower order harmonics low without affecting the fundamental component. On the other hand, the endeavour of implementing natural sampled PWM waveform in real-time has also resulted in the development of modified PWM methods based on natural sampling strategy. Some of such schemes are selected for discussion as follows.

A modified sampling technique, first introduced by Acharya *et al* [Acharya,1986], is again illustrated in Fig.3.5. In this method, the average of the samples at points corresponding to the two successive regularly sampled points at positive and negative apexes of the triangular carrier wave is used as the modified modulating signal. The implementation of this method entails the use of sample and hold (S/H) devices to store the amplitude of the sine (modulating) wave at the instants corresponding to the apexes of the triangular wave. The sine values of the two successive angles are averaged to give new S/H modulating signal as shown. Using this method resulted in an increase of two percent in the fundamental harmonic of the output voltage and a decrease of 20 to 30 percent of total harmonics content. The method was implemented by means of an 16-bit microprocessor and the PWM switching instants were calculated in real-time without incurring any additional memory requirement.

The injection of a third harmonic component into the sinusoidal modulating wave to reduce motor losses was proposed by several authors [Houldsworth,1984; Bowes,1985; Boys,1985]. The third harmonic component causes no current to flow in a three-wire system but shifts the harmonic energy to the less critical triple integer and higher order frequency components. A reduction of up to 30% harmonic I^2R loss as a result of injecting a 28% of third harmonic component to the modulating sinewave was claimed by Boys [Boys,1985]. It is also noted that the output voltage range is also increased by this method.

In another modified scheme [Khanniche,1988], a linear approximation of the modulating sine wave was proposed. The sine wave is represented by a series of linear segments, which lengths are determined by the carrier frequency. The approximated sine wave is assumed to be varying linearly with time over the sampling period, and changing slope at every peak of carrier wave as shown in Fig.3.6. The slopes and vertices of these segments that make up the sine wave are determined by sampling the continuous sinewave at every peak of carrier waveform. Alternatively, the sine wave may be simply approximated by a trapezoidal waveform as shown in the same figure. This method, called piecewise linear approximation, was proposed in the reference [Varonvitsky,1983]. It is, however, apparent from Fig.3.6 that the linear approximation method results in a modulating

synthesized sine wave that is closer to an ideal sinewave than other sampling method. This method therefore provides a good approximated natural sampled PWM waveform and yet avoids the time penalty associated with the solution of the transcendental equation of the pulse-widths of the natural sampled PWM.

3.3.5 Optimised PWM Strategies

A number of optimised PWM schemes have been developed to optimise PWM parameters such as the total harmonic distortion factor (THD) [Bowes,1985], efficiency and loss [Boys,1985;Takahashi,1985], acoustic noise [Takahashi,1986], speed and torque ripples [Zach,1985]. It is noted from these publications that each optimisation criterion requires its own optimisation procedures. However, a generalised optimisation method consisting of minimization of a cost function of operation of an inverter drive was developed by Buja [Buja,1980]. It should be noted that [Enjeti,1990] all processes constitute the minimization of unwanted effect due to the harmonics present, and therefore converge towards each other when significant numbers of low-order harmonics are eliminated.

Unlike the natural or regular sampled PWM processes that are based on well-defined modulation processes, such optimised PWM strategies do not have identifiable modulation processes. This precludes any contemplation of on-line generation of the optimised PWM waveforms. Moreover, such optimisation procedures usually result in non-linear relationships between the switching angles and the fundamental harmonic component of the PWM voltage. Therefore, a large number of look-up tables are usually required, each corresponding to a discrete fundamental voltage level.

In the harmonic elimination scheme, for example, any unwanted harmonics can be eliminated by equating the unwanted harmonics to zero and assigning a value to the amplitude of the fundamental component. A generalized quarter-wave symmetric PWM waveform is shown in Fig.3.7, where α_1, α_2, ... define the switching angles. The Fourier coefficients of a quarter-wave waveform are given by equation (3.4):

$$A_n = \frac{4}{m\pi}\left[1 + 2\sum_{k=1}^{m}(-1)^k \cos n\alpha_k\right] \quad (3.4a)$$

$$B_n = 0 \quad (3.4b)$$

As these equations are non-linear as well as being transcendental in nature, iterative techniques such as the Newton Raphson method may be employed. The solving of these equations is usually a very computational extensive process, which is usually done off-line on main-frame computers.

Recently, near optimised, or suboptimum PWM strategies [Bowes,1985] have been developed, where the aim has been to realise these optimised PWM in real-time. These strategies are based on the need to maintain a well-defined modulation process that can be simply and efficiently implemented in software and therefore on-line, and yet still retain the desirable characteristics of the optimised PWM.

3.3.6 Space Vector Modulation

The space vector modulation is a new method of AC waveform generation, which has drawn much interest quite recently [Boys,1990; Moynihan,1991]. A space vector modulator switches between the available, spatially fixed, vectors to produce an average vector of the required angular velocity, and magnitude. The period over which the vectors are averaged defines the switching frequency and determines the ripple currents flowing in the motor. By selecting the correct order of switching the power devices, it is possible to minimize the switching rate of the devices and hence improve the efficiency of the drive. Space vector modulation offers the advantage of direct manipulation of flux vectors that allows the unwanted current and torque ripples to be tightly and effectively contained, and readily lends itself to implementation in a digital signal processor. In contrast, traditional modulation strategies such as the regular sampled PWM, allow only indirect control of the flux vector.

3.3.7 Initial Implementation and Evaluation of the Natural Sampling and Regular Asymmetric Sampling Using the Transputer

3.3.7.1 The Transputer PWM Generator

A prototype transputer-based PWM generator was designed to generate single-phase natural and regular sampled asymmetric PWM waveforms. The transputer PWM generator mainly comprised of an INMOS B004 board (hosted by an IBM-AT compatible), a multi-channel 12-bit D-to-A converter and an 8-bit timer, as showed in Fig.3.8. A standard interface device of the transputer, called 'link adaptor', (see Appendix-A) was used as an interface for each link of the transputer to another external device. The handshaking of data communication was achieved by simple hardware TTL gates.

The upper shaded block of Fig.3.8 comprises the natural sampling method. To output the modulating signal, the D-to-A converter receives 2 bytes of data in succession from the transputer, first the least significant bit (LSB), then the most significant bit (MSB). The first 4 bits of the MSB are ignored. This is then followed by a byte word that selects the channel on which the previous 12-bit data is output. The procedure is repeated for outputting the carrier signal. The signals are then compared by means of an analogue comparator. The modulating and carrier signals are not output to the comparator simultaneously, but are separated in time by the total time required for conversion and output of each signal, which is about 20 μs. For a 50 Hz modulating wave, the phase shift created is thus 0.1 %, and can therefore be neglected. The lower shaded block of Fig.3.8, which mainly consists of an 8-bit timer, is for the generation of the regular sampled asymmetric PWM. Two link adaptors are used for communication with the timer — one for data and the other for control signals. The timer outputs the PWM waveform by timing out its loaded value input from the transputer. The description of generation of regular sampled asymmetric PWM by the transputer is expounded in chapter 6.

3.3.7.2 Experimental Results

A series of experiments were carried out on the prototype transputer based PWM generator where PWM waveforms over a wide range (2 Hz - 100 Hz) of output frequency by both sampling methods were generated. A typical PWM waveform of modulation

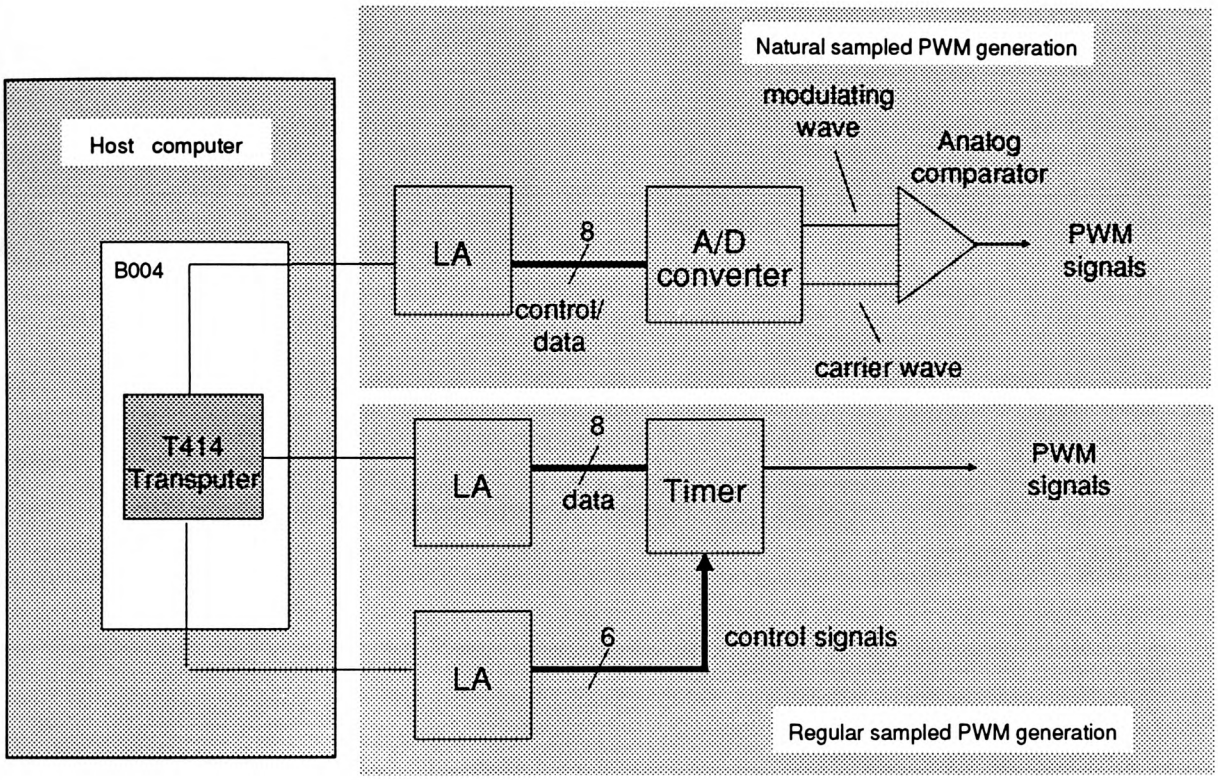
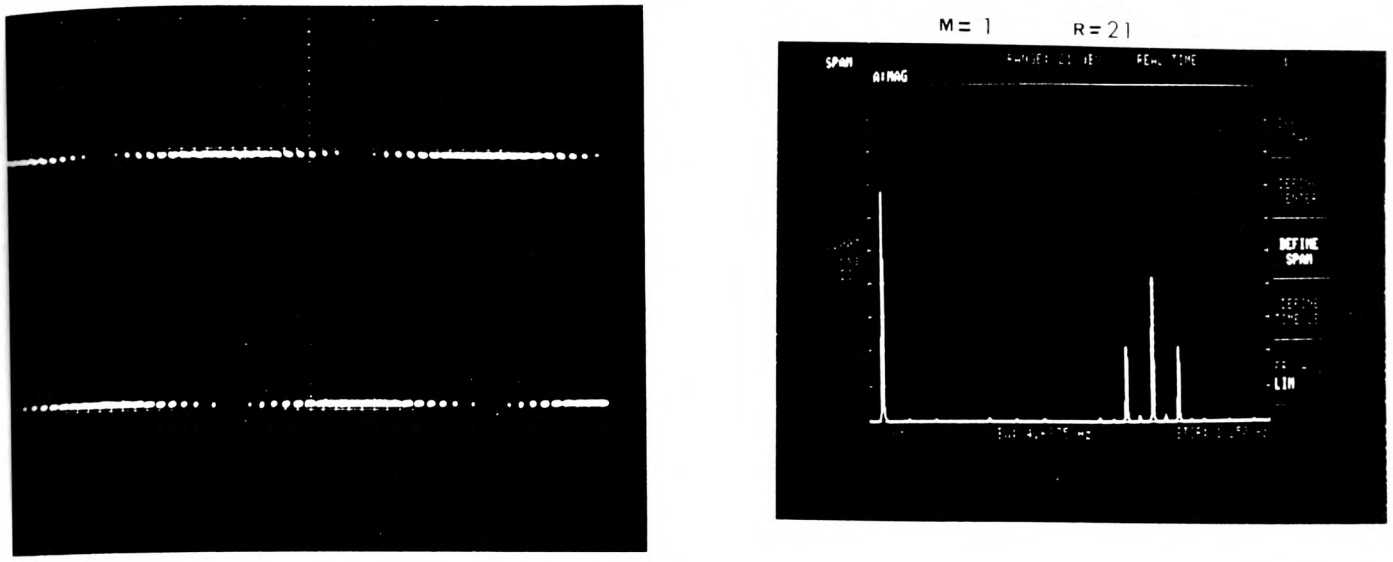
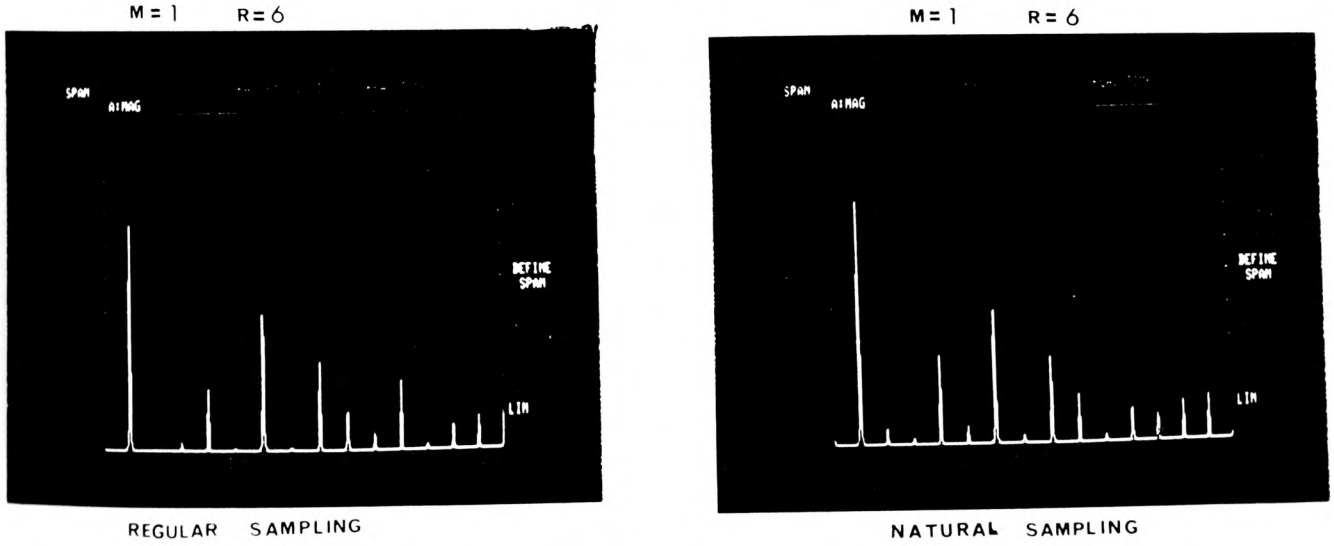


Fig.3.8 Block diagram of the prototype transputer PWM generator



(a) (b)

Fig.3.9 Generated regular sampled asymmetric PWM (M=1,R=21)
(a) waveform (b) harmonic spectrum



(a) (b)

Fig.3.10 Comparison of harmonic content (M=1,R=6,f=50Hz)
(a) regular asymmetric sampling (b) natural sampling

frequency of 33 Hz and a frequency ratio of 21 employing regular sampled asymmetric method is shown in Fig.3.9(a). The corresponding harmonic spectrum is shown in Fig.3.9(b). To compare the results of the two sampling methods, the harmonic spectra of both PWM waveforms using the same modulation parameters were studied. The harmonic spectra of the PWM waveforms when modulating frequency is 50 Hz, modulation index is 1 and frequency ratio is 6, are shown in Fig.3.10. Since the output level of the analog comparator is different from that of the timer that has typical TTL logic outputs, the magnitudes of the fundamental output of the two PWM waveforms are quite different. However, if the relative magnitude of the harmonics to the respective fundamental component is compared, the regular asymmetric sampling method is found superior to the natural sampling method, when such low frequency ratio as 6 is used. This is consistent with verified results reported elsewhere [Jayne,1977].

3.3.7.3 Interim Remark

From the above experimental results, it is shown that real-time generation of the regular sampled asymmetric PWM in the interested range of frequency, can easily be handled by the T414 transputer. As with other microprocessor-based methods, the transputer-based implementation suited the regular sampling method rather than the natural sampling method. This result has led to the decision to use the regular asymmetric sampling method in the final drive system (chapter 6,7). This initial evaluation of the use of the transputer in the generation of PWM waveforms was a consequence of the preparation of a paper presented by the author in Belfast [Luk,1989b]. The B004 transputer board used in the prototype transputer PWM generator was later superseded by an upgraded B008 board. This initial method of generating a single phase PWM was, however, replaced by a superior method capable of generating a 3- phase PWM waveform at a later stage, as will be described in Chapter 6.

3.4 VECTOR CONTROL

3.4.1 Introduction

The concept of field-oriented control (FOC), or more commonly known as vector control, for AC machines, was commercially introduced by Siemens Company in the mid 70's, and was based on vector transformation of motor waveforms from a stationary to a rotating frame and vice versa. The aim of the scheme was to improve the dynamic response of the AC machines. The concept of transformation has been used extensively in the unified machine theory [Adkins,1975]. It is shown in the unified machine theory that AC or DC machines are generically similar, and that any machine can in fact be represented by a primitive machine by means of appropriate transformations. Because of the demand for low maintenance induction motor drive systems over the last decade, great attention has been given to the application of field orientation schemes to drive systems.

3.4.2 Review of Vector Control Strategy

Although research in the application of vector control strategies in induction motors has been carried out in many countries, there does not appear to be a comprehensive review of the relative advances made by researchers in different countries. It was therefore decided to make such a review, the result of which is presented in the following sections.

(a) Germany

The theory of vector control was first introduced by Blaschke and Hasse in Germany in the early 70's. In the early 80's, attention was already given to the parameter adaptive scheme of the rotor resistance to improve the reliability of vector control under unfavourable conditions [Garces,1980]. However, the mainstream of research in vector control in the last decade was mainly led by Leonhard *et al* [Leonhard,1975; Gabriel,1980], who are based at the same institute as Blaschke's — Technische Universitat Braunschweig. Leonhard and his researchers continue to make great progress in the application of vector field control, and his work is very well documented [Leonhard,1985]. Other researchers such as Holtz of University of Wuppertal, for example, have also published several papers on different application areas of vector control [Holtz,1983;85;88]. More recently, Naunin *et al* of Technical University of Berlin

[Naunin,1991], compared the use of microprocessors and transputers in the control of AC induction motors. Thus, the literature shows that the level of research activity in Germany has maintained its leading position in the application of vector control to variable speed AC drives.

(b) The United States

The application of vector control in the United States was first taken up by Bose and Lipo at the University of Wisconsin-Madison. Although Bose himself favours a fundamentally different scheme of control — the scalar control [Bose,1984], he was amongst the first to introduce the vector control outside Germany [Bose,1981]. The early research work by Lipo *et al* was involved in the establishment of a new rotor identification scheme and an on-line optimization procedure reported in the co-authored publications [Matsuo,1985;Kirschen,1985] respectively. More recently, Lorenz *et al* has published no less than five papers [Lorenz,1986;90] on both theoretical analysis and practical implementations of different vector control schemes on variable speed AC drives.

(c) Japan

In Japan, the application of vector control emerged on the industrial arena in the early the 80's. Akamatsu *et al* [Akamatsu,1982] of Fuji Electric Company, and Masachiro *et al* of Mitsubishi Electric Company [Masachiro,1982], were amongst the earliest Japanese authors who published their work on the use of vector control to improve the performance of the current fed inverter (CSI) induction motor drives. The work by Masachiro mainly entailed the incorporation of vector control together with a specialized scheme to generate PWM waveforms from zero to rated frequency for a CSI motor drive. The transient performance of the resulting drive system was comparable to that of the DC drive. Similarly, the work by Masahiro *et al* was also associated with vector control of large motors fed by CSI. However, a rotor and flux variation compensation scheme was introduced to achieve high performance even during unfavourable conditions such as prolonged running of the drive. A further industrial commitment to the application of vector control on variable speed drives was demonstrated by Toshiba Corpoartion [Miyazaki,1984], where a standardized vector control hardware unit for different drive systems was developed. The use of vector control was applied to other operation

conditions such as light load running at high speeds [Ito,1983]. It was suggested by Ito *et al* that field-weakening technique should be incorporated into vector control to stabilise the drive system at light load and high speed operations. In the mid 80's, many schemes were developed where the main aim was to identify [Matsuo,1985], and compensate the rotor parameters [Nagase,1984]. Due to the complexity of many of the schemes suggested, where real-time implementation was required, such schemes could only be realised with very powerful sequential processors or multi-processor systems. This led to the development of multi-processor vector control drive systems. Harashima *et al*, for example, developed a multi-processor control system which improved the response of induction motor drives [Harashima,1985]. The result of the work suggested two types of control referred as I-type and V-type methods. Harashima also suggested the method for compensation of change in rotor resistance. However, Harashima failed to address the complexity of the hardware requirement to implement the scheme proposed. The implementation difficulty experienced when using a multi-processor system was shortly expanded upon by Kubo *et al* [Kubo,1985], who also used a multi-microprocessor system for vector control of induction motor drives. Although the method used by Kubo adopted a slightly different approach, the problems associated with the communication among the processors were addressed. It was interesting to note, however, that a vector control system with rotor time constant identification scheme using only a single high-speed sequential 16-bit microprocessor was developed by Koyama *et al* [Koyama,1986]. Another method of overcoming the variation of motor parameters such as rotor resistance was suggested by Ohnishi [Ohnishi,1986], which mainly consisted of model reference adaptive scheme (MRAS). In this method, the ideal motor model ran in parallel with the actual motor and the output of the model was compared with the real output of the motor. The difference of these outputs, the error, was then dependent upon the parameter variation of the actual motor. An appropriate adaptive mechanism was employed to offset the error. The application of MRAS was further developed by Sugimoto *et al* [Sugimoto,1987], who suggested an on-line identification method under any load and speed conditions of operation. Sugimoto's scheme mainly consisted of the injection of a sinusoidal signal into the flux axis primary current, which allowed the rotor resistance to be identified under the operating condition. More recently, another alternative scheme [Haneda,1989] proposed the use of a globally stable and parameter insensitive

observer-linearizer, which was implemented by two processors in an optimal position servo application.

(d) The United Kingdom

Research work in the area of vector control to date in this country is relatively recent and not widespread. In 1987, Acarnley *et al* presented the prospect of field-orientation in AC drives in an IEE meeting in London [Acarnley,1987]. Acarnley *et al*, based at the Electric Drives and Machines Group of Newcastle University, are one of the few known research groups in the U.K. who are actively involved in research on vector control of induction motors. The development of the methods of parameter identification constitutes the main activity of this group [Acarnley,1991]. An initial implementation of the vector control by the use of the transputer was reported by Sumner and Asher of Nottingham University in 1986 [Sumner,1986]. This was followed by an extended implementation in 1990 [Asher,1990]. Recently investigation into the parameter sensitivity has been carried at Birmingham University [Du,1991].

(e) South Africa

For the last couple of years, research in vector control has been actively taken place in several South African universities. In the University of Natal, for example, a team of no less than six researchers have engaged in a program on developing a very comprehensive motion control development system (MCDS) [Webster,1991]. Although it is still in its development stage, the scope of this program is reflected by the number of papers (at least six) presented in conjunction with the work.

(f) Others

The remainder of the research in vector control is made up of a number of smaller groups that are scattered in different countries. Notably the eastern European research groups [Kazmierkowski,1991] have concentrated on the theoretical aspects. Much of the work done by these researchers to date involves the simulation of new methods. In the Far East, such as Taiwan [Liu,1989] and Hong Kong [Chan,1990], implementations of vector control with varying degree of success are also reported by a small number of researchers.

To summarize, vector control is now a well-established research area in the field of electrical drives, although initial investigations commenced some twenty years ago. In the first decade, the majority of the research were carried out in Germany by people such as Hasse and Blaschke. During this time, much of the investigation centred around the establishment of control theory and schemes. The implementation of these vector control schemes was limited by the lack of computer processor speed available at that time. The advent of high processing speed microprocessor chips during the last decade, however, resulted in a surge of research activities in the implementation of vector control schemes and other control techniques such as adaptive control. The application of vector control to industrial drive systems has been strongly supported by Japanese researchers, with the active participation and support from their industrial organisations. However, the German researchers have maintained a strong hold on development in vector control technology and have continued to publish the results of their work. So far as the U.S. is concerned, research in vector control technology still centres around the same University where vector control was first applied in the country. Similarly, Eastern European countries are still concentrating on the analysis and simulation of vector control but still do not appear to make much advancement on its practical application. The reviewing of the literature available on vector control on a worldwide basis shows that many of the terminologies used by different countries can be quite different. It should also be noted that a number of different frames of reference are also used, typical examples being rotor flux orientation, air-gap flux orientation, and stator flux orientation. Some interesting research work by Profumo [Profumo,1991] analyzed the effect of the different frames of orientation on vector control schemes and concluded that there was no overall advantage by one frame of orientation over the other.

3.4.3 Classifications of Field Orientation Schemes

3.4.3.1 Direct Vector Control

The control of magnetising flux or field in induction motors is central to the basic philosophy of field orientation. It is therefore natural to measure the airgap flux directly by means of Hall's probe or search coils. This method, first developed by Blaschke

[Blaschke,1972], is illustrated in the simplified block diagram of Fig.3.11. The controller, which may be an analog system or a computer, together with the power conditioning unit (inverter), determines the required flux and torque levels from the speed and torque input demands, and outputs the power to the motor. The flux of the motor is measured by two flux sensing devices placed orthogonally in the stator. The essential feature of the direct control scheme is the transformation of rotating to stationary and vice versa co-ordinate representation of the drive system by means of the flux signal detected by the flux sensing devices. The resulting stationary frame signals are then converted to phase current commands and used to control the inverter. Similarly, the measured airgap flux signal from the flux sensor is fed back to the controller for flux compensating, where the torque producing component of stator current is separated from the flux producing component.

The advantage of the direct vector control scheme is its insensitivity to changes in motor parameters. However, one major problem with this method is that difficulty may arise in using the Hall effect device because the measurement of flux density at a single point does not give a reliable indication of flux linkages. The signal can also be very 'noisy' as a result of slot ripples. Although the installation of a filter may reduce noise level, it introduces frequency-sensitive phase shift and therefore worsens the decoupling effect between the torque producing component and magnetising component of the motor flux. Hall effect devices are also temperature sensitive and the signal derived from them can lead to error when a wide range of temperature change is experienced. Alternatively, although search coils are more robust and less temperature sensitive, they have the inherent problem of not detecting flux change when the field is stationary. The requirement of flux detection that is fundamental to direct vector control has been a main problem in applying vector control to existing in-service induction motor drives. Therefore, variable speed vector control electrical drives are generally produced as a complete package rather than being applied to existing drive systems.

3.4.3.2 Indirect Flux Sensing

The indirect flux sensing method, pioneered by Hasse [Hasse,1969], is an attempt to avoid the use of flux sensors. This is achieved by measuring the quantities such as stator

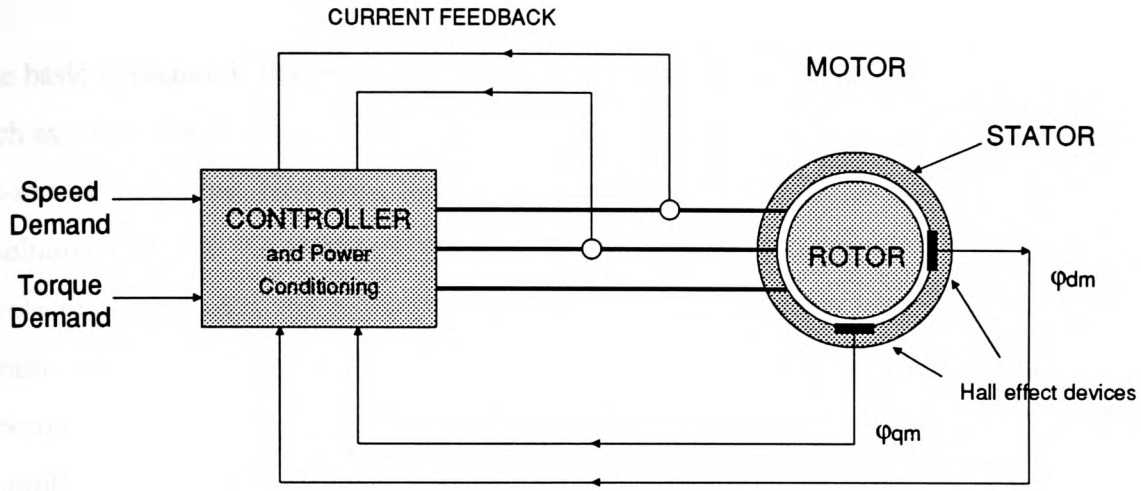


Fig.3.11 Simplified block diagram of direct vector control

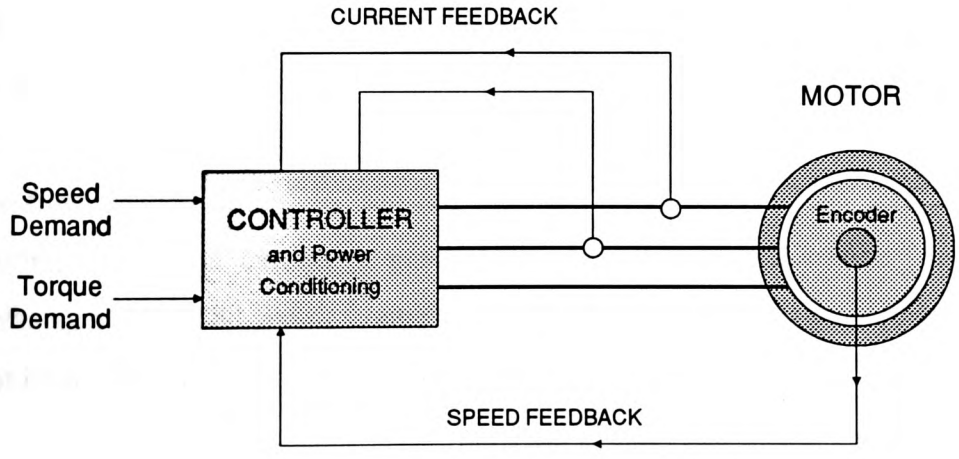


Fig.3.12 Simplified block diagram of indirect vector control

voltages, stator currents and rotor speed and calculating the flux from these measurements.

The basic operation is illustrated in Fig.3.12. The sensor used is a speed measuring device such as a tachometer. Such a device can easily be fitted onto the non-drive end shaft of an 'in-service' machine. However, the device is required to have high accuracy and good resolution such that the estimated value of the flux is close enough to its real value for a viable implementation. Moreover, this method is more sensitive to variation in motor parameters resulting from changes in operating temperature, current and frequency. Nevertheless, when being compared with the direct vector control, the indirect vector control is found to be much easier to implement and therefore becomes more acceptable for commercial exploitation in industrial products. The literature available suggests that the indirect flux method is preferred for industrial applications.

3.4.3.3 Vector Control Without Sensors

Although it is obvious that vector control without employing flux and speed sensors is preferable, it should be noted this method has been available only relatively recently [Kazmierkowski,1985]. In this method, a torque proportional quantity was calculated from stator voltage and current without the measurement or calculation of flux and the speed signal was replaced by the stator frequency signal. The disadvantage of this method is the decrease in dynamic response of the system due to the ripples in the frequency signal. In another method described by Davies [Davies,1991], the motor impedance and power flow are monitored by sensing the voltage at the machine terminals and by measuring DC busbar current or power. However, the method suffers from the same disadvantages as a DC motor controller in low speeds when the machine parameters cannot be measured with the same accuracy without introducing a compensation scheme.

3.4.4 Theory of Vector Control

3.4.4.1 Complex Vector Approach

The representation of vector quantities by complex algebra can be very advantageous when the motor is represented in its two- dimensional form, and end effects of the motor

are neglected. This provides a much more concise analysis when using vector control theory. This method, which was first used by Blaschke and Hasse, has been since adopted widely by researchers in electrical drives, particularly in Germany. The method of complex vector, when applied to a direct- and quadrature-axis representation of an induction motor may be developed as follows [Leonhard,1985].

With the usual simplifications such as neglecting the slot effects and spatial harmonics, assuming infinite permeability of iron and disregarding iron losses and eddy currents, the complex time-dependent current vector for the stator is given by equation (3.7a):

$$i_s(t) = i_{sa}(t) + i_{sb}(t)\, e^{j\gamma} + i_{sc}(t)\, e^{j2\gamma} \qquad (3.7a)$$

where $\gamma = \frac{2\pi}{3}$ and subscripts sa, sb and sc pertain to phase a, b and c of stator winding.

This equation defines the instantaneous magnitude and space position of the sinusoidal MMF wave. A similar expression exists for the rotor current vector:

$$i_r(t) = i_{ra}(t) + i_{rb}(t)\, e^{j\gamma} + i_{rc}(t)\, e^{j2\gamma} \qquad (3.7b)$$

Although these current vectors can be mathematically defined as in equations (3.7), they cannot be measured in a real machine, whereas the MMF is a measurable and real quantity.

The voltage equations for the stator and rotor windings can be defined as:

$$\underline{v}_s = R_s i_s + \frac{d\psi}{dt} = R_s i_s + L_s \frac{di_s}{dt} + L_m \frac{d}{dt}(i_r\, e^{j\varepsilon}) \qquad (3.8a)$$

$$\underline{v}_r = R_r i_r + \frac{d\psi}{dt} = R_r i_r + L_r \frac{di_r}{dt} + L_m \frac{d}{dt}(i_s\, e^{-j\varepsilon}) \qquad (3.8b)$$

The stator current, i_s, leads the rotor current, i_r, at an angle of ε as depicted in Fig.3.13. The stator and rotor inductances can be represented in terms of stator leakage factor, σ_s, mutual inductance L_m, and the rotor leakage factor, σ_r, as follows.

$$L_S = (1 + \sigma_S) * L_m, \quad L_r = (1 + \sigma_r) * L_m \tag{3.9}$$

The torque equation which describes the mechanical dynamic of the motor is given as:

$$J\frac{d\omega}{dt} = T_e - T_L = \frac{2}{3} L_m \,\mathrm{Im}\,[\, i_S\,(i_r e^{j\varepsilon})^*] - T_L \tag{3.10}$$

where J is the moment of inertia and T_L is the load torque.

The four equations of (3.7) and (3.8), together with the torque equation of (3.10), constitute a dynamic mathematical model of symmetrical induction motors, and have proved adequate for representation of the motor when simulating and analysing the performance of such drive systems incorporating vector control. It should be noted, however, that the model is not adequate to represent such effects as slot ripples and tooth saturation, as would be required in machine design.

The further introduction of the field coordinates, which was originated by Blaschke [Blaschke,1972], transforms the last equations into a form that is similar to that used for DC machines. Blaschke used a modified magnetizing current vector, $i_{mr}(t)$, given by equation (3.11), to represent the new flux reference:

$$i_{mr}(t) = i_S(t) + (1 + \sigma_r)\, i_r(t)\, e^{j\rho} \tag{3.11}$$

The instantaneous angular velocity of this vector is the stator frequency, and is given by:

$$\frac{d\rho}{dt} = \omega_o(t) \tag{3.12}$$

With the new reference vector $i_{mr}(t)$, as shown in Fig.3.13, the stator direct and quadrature components of current and voltage may be defined respectively, as follows:

$$i_S(t)e^{j\rho} = i_{sd} + j i_{sq} \tag{3.13a}$$

$$u_S(t)e^{j\rho} = u_{sd} + j u_{sq} \tag{3.13b}$$

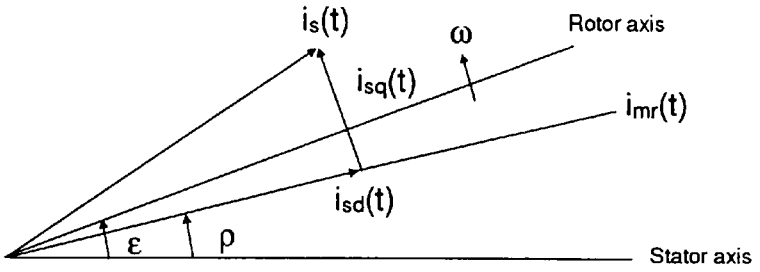


Fig.3.13 Phasor diagram of quantities in vector control by Leonhard's method

Using the expression defined by equations (3.8), (3.9), (3.12) and (3.13), the following equations can be determined for the transformed variables [Leonhard,1985].

$$T_r \frac{di_{mr}}{dt} + i_{mr} = i_{sd} \ , \quad T_r = \frac{L_r}{R_r} \tag{3.14}$$

$$\frac{d\rho}{dt} = \omega_o = \frac{i_{sq}}{T_r i_{mr}} + \omega_r \tag{3.15}$$

$$T_e = k\, i_{mr}\, i_{sq}; \quad k = \frac{2}{3}\frac{P}{2}\frac{L_m}{1+\sigma_r}, \quad P = \text{No. of pole pairs} \tag{3.16}$$

$$\frac{d\varepsilon}{dt} = \omega_r \tag{3.17}$$

The dynamic response of the rotor current in the direct axis given is by equation (3.14), where the direct axis component of the stator current, i_{sd}, is taken as the input variable. The angular or space position of the flux vector, however, may be determined from equation (3.15) where the stator quadrature component, i_{sq}, is taken as input variable, assuming that the modified component of rotor current, i_{mr}, is maintained constant. Similarly, the torque provided by the motor can be determined from equation (3.16), where the stator quadrature component of current, i_{sq}, is the input variable. The torque is conveniently controlled by i_{sq} according to equation (3.16). These equations are analogous to those for a DC shunt excited motor, where the speed and torque of the machine are independently controlled by the field and armature currents.

3.4.4.2 Bose's Approach

This approach, first presented in detail by Bose in his book [Bose,1986], introduces and treats the field coordinates without the use of complex vector. This approach is widely used in the U.S, the U.K., and the Far East. Assuming the induction motor is singly fed, the voltage equations for the rotor in the rotating *d-q* frame are given as follows:

$$\frac{d\psi_{qr}}{dt} + R_r * i_{qr} + (\omega_o - \omega_r) * \psi_{dr} = 0 \qquad (3.18a)$$

$$\frac{d\psi_{dr}}{dt} + R_r * i_{dr} + (\omega_o - \omega_r) * \psi_{qr} = 0 \qquad (3.18b)$$

where ω_o and ω_r are the synchronous and rotor speeds repectively; whereas ψ_{dr}, ψ_{qr} are the direct and quadrature components of rotor flux respectively.

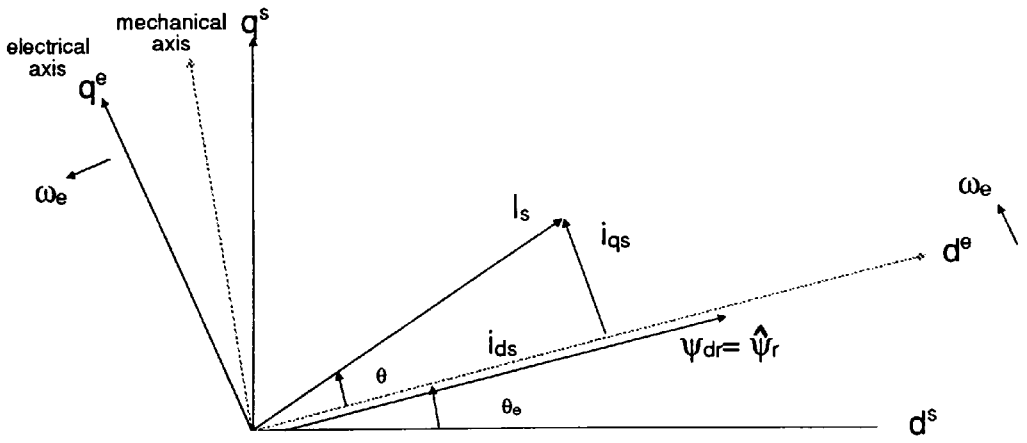


Fig.3.14 Phasor diagram of quantities in vector control in Bose' method

The quantities ψ_{dr} and ψ_{qr} are given by the equation (3.19) as follows:

$$\psi_{qr} = L_r * i_{qr} + L_m * i_{qs} \qquad (3.19a)$$

$$\psi_{dr} = L_r * i_{dr} + L_m * i_{ds} \qquad (3.19b)$$

Solving for i_{qr} and i_{dr} gives:

$$i_{qr} = \frac{1}{L_r} * \psi_{qr} - \frac{L_m}{L_r} * i_{qs} \tag{3.20a}$$

$$i_{dr} = \frac{1}{L_r} * \psi_{dr} - \frac{L_m}{L_r} * i_{ds} \tag{3.20b}$$

The rotor currents may be eliminated by substituting equation (3.20) into equation (3.18), the result of which gives the following equations:

$$\frac{d\psi_{qr}}{dt} + \frac{R_r}{L_r} * \psi_{qr} - \frac{L_m}{L_r} * R_r * i_{qs} + \omega_{sl} * \psi_{dr} = 0 \tag{3.21a}$$

$$\frac{d\psi_{dr}}{dt} + \frac{R_r}{L_r} * \psi_{dr} - \frac{L_m}{L_r} * R_r * i_{ds} + \omega_{sl} * \psi_{qr} = 0 \tag{3.21b}$$

In order to decouple the torque producing component of the rotor current from the flux producing component of the stator current, the rotor flux that revolves at the synchronous speed, must be aligned with the *d*-axis. Then the *q*-axis component of the rotor flux vanishes and the *d*-axis component becomes the total rotor flux. Therefore,

$$\psi_{qr} = \frac{d\psi_{qr}}{dt} = 0 \tag{3.22}$$

Analogous to the control of DC motors, it is the best policy to maintain ψ_{dr} constant, usually at its maximum level, limited either by iron saturation, or, at above base speed, the DC link voltage of the inverter. Thus,

$$\psi_{dr} = \hat{\psi}_r = constant \tag{3.23}$$

Therefore,

$$\frac{d\psi_{dr}}{dt} = 0 \tag{3.24}$$

Substituting equations (3.22-24) into equations (3.21a) and (3.21b) gives the following relationships for, ω_{sl}, the angular slip frequency:

$$\omega_{sl} = \omega_o - \omega_r = \frac{L_m}{\hat{\psi}_r} * \frac{R_r}{L_r} * i_{qs} \tag{3.25a}$$

$$\frac{L_r}{R_r}\frac{d\psi_{dr}}{dt} + \hat{\psi}_r = L_m i_{ds} \tag{3.25b}$$

Since $\hat{\psi}_r = L_m * i_{ds}$, and noting the condition given by equation (3.24), then when this relationship is substituted into equation (3.21b), the following relationship is obtained:

$$\omega_{sl} = \frac{R_r}{L_r} * \frac{i_{qs}}{i_{ds}} \tag{3.26}$$

The torque equation as a function of stator currents and stator flux is given by:

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)(i_{qs}\psi_{ds} - i_{ds}\psi_{qs}) \tag{3.27}$$

To eliminate the stator fluxes, the following flux leakage relations can be used:

$$i_{qr} = \frac{1}{L_r} * \psi_{qr} - \frac{L_m}{L_r} * i_{qs} \tag{3.28a}$$

$$i_{dr} = \frac{1}{L_r} * \psi_{dr} - \frac{L_m}{L_r} * i_{ds} \tag{3.28b}$$

Thus, the torque equation, after substituting equations (3.28) into equation (3.27) and simplifying, becomes:

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)\frac{L_m}{L_r}(i_{qs}\psi_{dr} - i_{ds}\psi_{qr}) \tag{3.29}$$

Since $\psi_{qr} = 0$ and $\psi_{dr} = \hat{\psi}_r$, the torque expression is reduced to:

$$T_e = \frac{3}{2}\left(\frac{P}{2}\right)\frac{L_m}{L_r} i_{qs}\hat{\psi}_r \tag{3.30}$$

Therefore, from equation (3.30), provided the inverter can deliver the required currents, i_{ds} and i_{qs}, the developed torque T_e will respond instantaneously with i_{qs} and has a delayed response due to i_{ds}, which is analogous to the separately excited DC machine.

3.4.4.3 The V-type Vector Control Method

In the derivation of the conventional vector control methods as described in the previous sections, it is assumed that the induction motor is fed by a controlled current source that can provide the required impressed current instantaneously. In practice, however, a voltage source such as a voltage fed inverter is more commonly used especially in low and medium power drives. In many industrial applications, a voltage controlled PWM inverter with local current feedback is used for induction motor drive systems. To achieve the performance of a controlled current source, the voltage controlled PWM inverter usually includes a very high gain in the local current loop. The 'bang-bang' control used in a PWM inverter, for example, is essentially a voltage fed inverter with a high current gain. In this method, the current is forced to follow the referenced current by means of a voltage source.

It should be noted that the stator circuit in such cases is omitted. However, if a controlled voltage source is used, the 'electrical dynamics' due to stator circuit should be monitored in order to achieve an entirely decoupled model of the drive system. A control strategy based on the exact model of a voltage fed inverter induction motor drive was developed by Harashima [Harashima,1985]. In this scheme, the normal control laws to decouple the rotor circuit in the conventional, current-controlled vector control were performed. The coupling terms in the control model of an induction motor, which are responsible for the sluggish response of the motor, has been shown in Fig.2.6. However, it can be shown how these terms in a voltage fed inverter drive may be compensated for by means of control algorithms as follows.

Suppose the direct and quadrature components of the stator voltage, V_{ds} and V_{qs} respectively, are controlled according to the following equations.

$$V_{ds} = R_s * i_{ds} - \omega_o * L_\sigma * i_{qs} \qquad (3.31)$$

$$V_{qs} = R_s * i_{qs} + \omega_o * \frac{(L_s * L_r - L_m^2) * p + L_s * R_r}{L_r * p + R_r} * i_{ds} \qquad (3.32)$$

Since $i_{ds} = constant$, then

$$V_{qs} = R_s * i_{qs} + \omega_o * L_s * i_{ds} \qquad (3.33)$$

It can be seen that if the above control laws are implemented, the interacting terms of the hatched block part in Fig.3.15, which includes the power conditioning and the controller to the motor model of Fig.2.6, can be compensated. The 'dynamics' of the stator current i_{ds} and i_{qs} can be decoupled. Consequently, when the equations of the control laws are upheld, the system can be considered as a linearised model, as envisaged in Fig.3.16. Thus, the model demonstrates that a quick torque response to a stator voltage change can be achieved. This control strategy is referred to as 'V-type' control in the literature.

3.5 INTERIM CONCLUSION

From the discussion of the vector control strategies in this chapter, it is evident that the analysis of such vector control drive systems and their implementation are very involved, and also demanding in mathematics knowledge such as the complex vector analysis. However, it will be shown in the following chapters how vector control can be usefully and efficiently adapted to induction motor drives by means of the transputer.

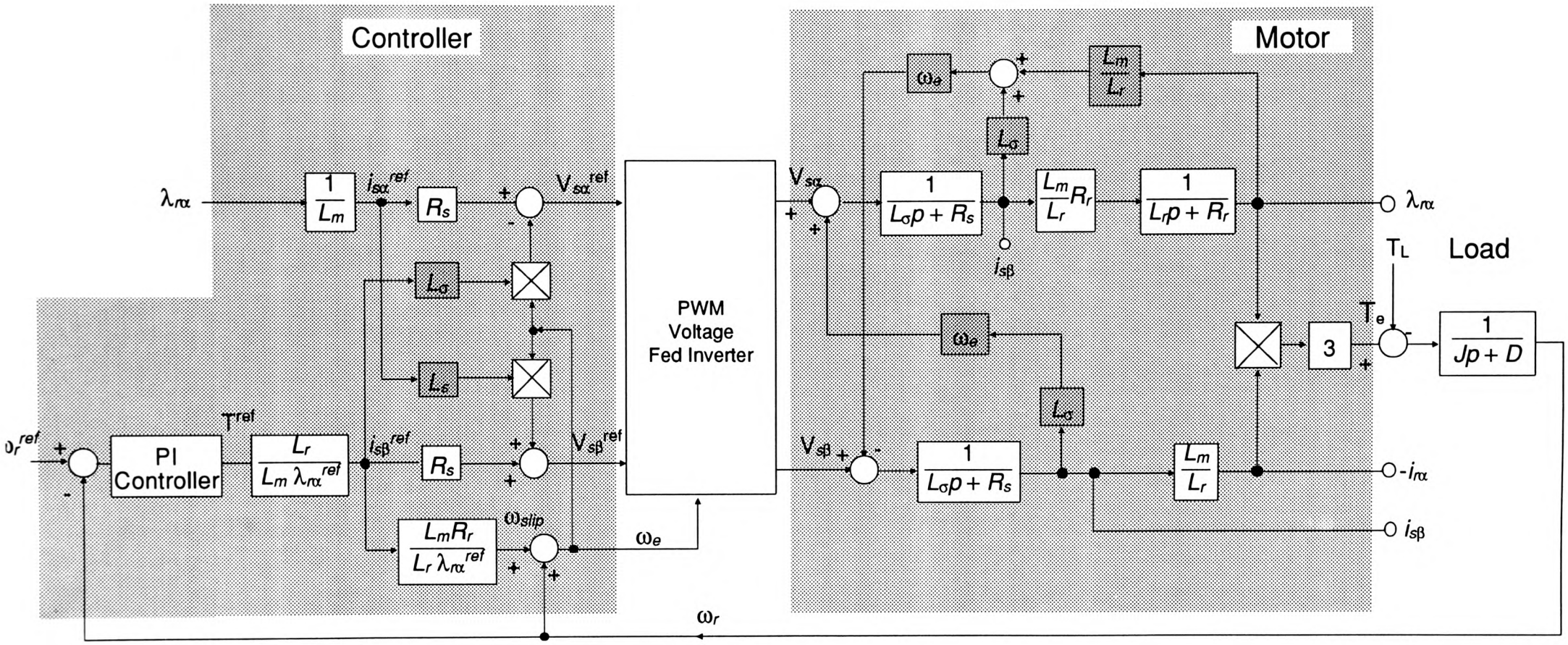


Fig.3.15 Block diagram of vector control in voltage fed induction motor drive

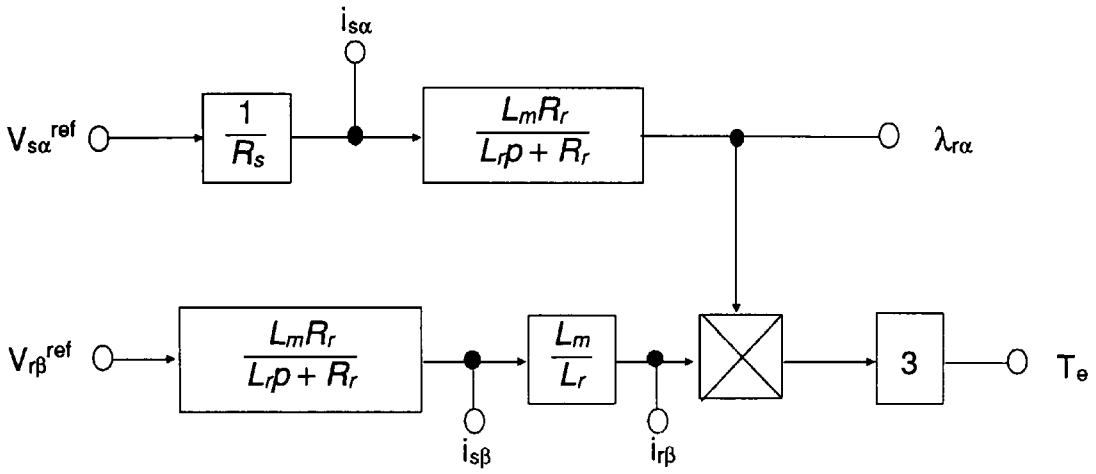


Fig.3.16 Linearised (decoupled) model for V-type vector control

Chapter 4 : Digital Simulation of Pulse Width Modulated (PWM) Inverter Drives

4.1 INTRODUCTION

Whilst the use of semiconductor switching devices has widened the application of the induction motor in variable voltage, variable frequency drives, it may also add extra complexity to the simulation of such drive systems. For example, if an accurate model of an induction motor drive system consisting of a transistorised inverter bridge is required, it is virtually impossible to obtain a general analytical solution. The use of analogue computers in modelling and simulating complex systems has proved particularly popular because a faster and less expensive solution can be obtained when compared to a digital system. The modelling of induction motors with saturable leakage reactances illustrated in reference [Lipo,1984], for example, demonstrates how an analogue simulation method may be efficiently applied to a complex system. However, the analogue technique suffers the well-known problems associated with analogue systems such as DC drift, offset error and inflexibility. With the ever increasing processing power of present day digital computers, and their price falling steadily, digital modelling of induction motor drives is becoming commonplace [Lipo,1975; Bowes,1983; Ho,1986]. Such digital models of the motor drive system are usually flexible enough to accommodate a change of system parameters with little change required for the software program describing the drive model. The simulation time on the digital model is also being reduced considerably by the powerful digital computer and by such specialised simulation software packages as 'Control-C'. Indeed, the aspiration of simulating induction motor drives in real-time by means of a multi-processor system is demonstrated in the reference [Tait,1984]. The advantages of using simulation methods that combine digital and analogue (hybrid) techniques have also been described by some authors [Kassakian,1979]. These simulation techniques provide a fast way of developing a model for complex drive systems and allow the control system under development to be debugged without subjecting to a real environment.

It should be noted, however, that digital simulation will become more widely accepted due to the future advancement in the computing power of digital computers and availability of specialised simulation software packages. It has also been suggested

[Bose,1989] that computer-aided design (CAD) would be used in the development of 'integrated drives' in the future. In developing such 'integrated drives', the design parameters of the motor and the power converter will be optimised to achieve better utilization of power and performance of the drive system. A preliminary example of using CAD to develop a comprehensive PWM drive system is found in the reference [Bowes,1988]. It is shown by Bowes how digital models of the PWM inverter and the motor, which are developed on a dedicated mini-computer, can be combined and used in the analysis of the drive system. Such a modular approach used in the development of the drive system allows individual modules such as the waveform generator and motor to be developed, tested and used independently. These individual modules can then be used as building blocks for further development of a complete drive system.

The development of the digital model of the drive system reported upon in this thesis also adopted a modular approach that is different from the approach used by Bowes. In the design of a digital model for the drive system, four main modules could be identified, as depicted in Fig.4.1. Such modules would be developed into program codes under the software environment. The '**PWM**' module embraces different sampling strategies for the generation of the PWM waveforms. This module inputs the values for different PWM parameters such as carrier frequency, modulation index and frequency ratio or pulse number. The output is the PWM signal waveform. The '**Power**' module represents a set of logic checking rules for ON and OFF states of different types of power semiconductor switches which may be used. This module inputs the PWM waveform signals generated by the '**PWM**' module. After the logic checking for the conduction of the power semiconductor switches has been completed, the '**Power**' module then outputs the PWM power signals. The '**Motor**' module comprises the motor equations in *d-q* or *A-B-C* frame representation. The module inputs the PWM power signals and outputs the machine flux, currents, voltages and generated torque. Within the '**Load**' module is a set of equations representing different load characteristics that are usually constant and/or a function of motor speed. The '**Load**' module inputs the generated torque from the motor and outputs the load speed and angular position.

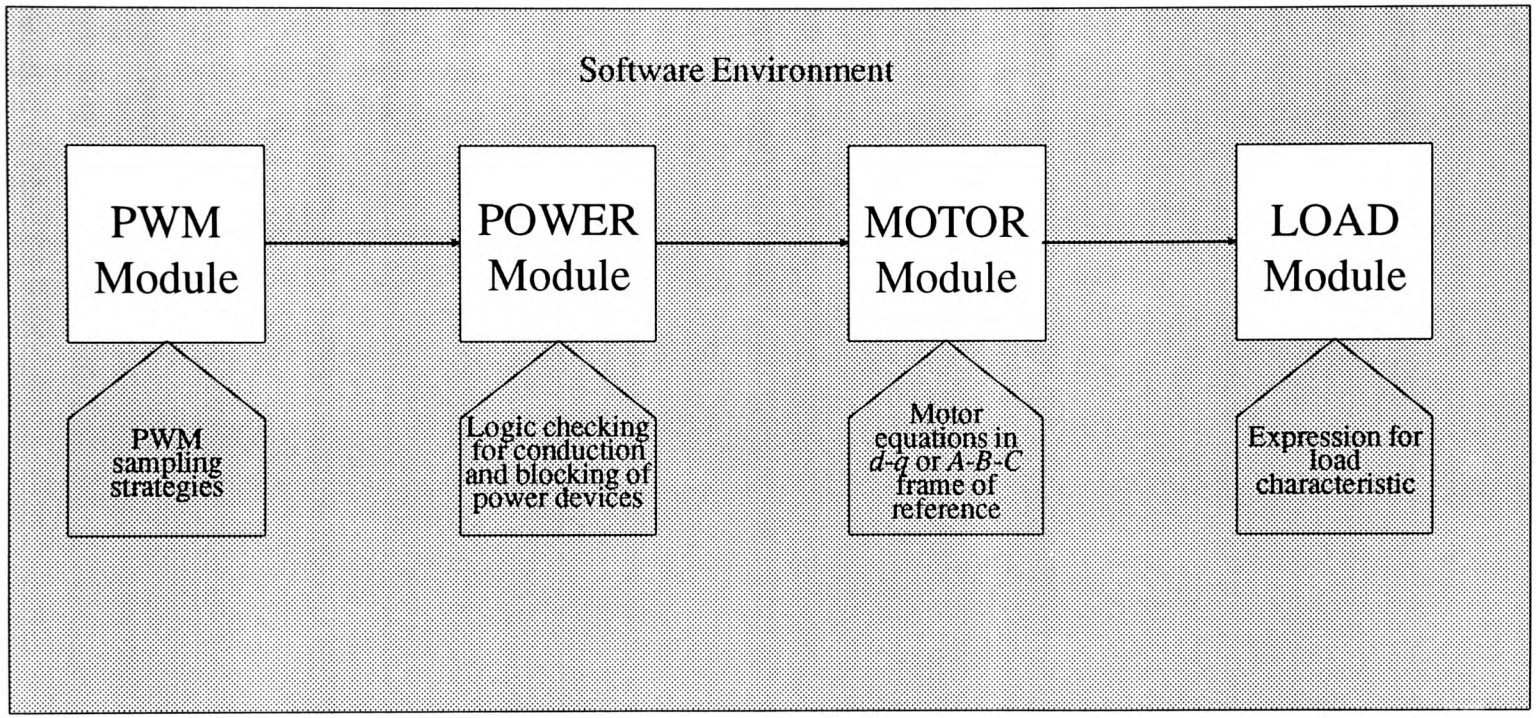


Fig.4.1 A digital model of a typical inverter drives system

4.2 SOFTWARE SIMULATION TOOLS

4.2.1 Review of Simulation Software Packages

A number of general circuit simulation programs such as ECAP, CSMP and SPICE are available, but do not lend themselves for motor drives simulation and analysis since they are intended for the application of a large and general class of circuits, requiring minimum programming effort from the user. More recently, commercial simulation software packages designed specifically for the converter power drives have been available, one example being the '*Computer Electronics Laboratory*', marketed by Orange Enterprises. These packages, however, can only be used for steady-state simulation of such drive systems. Moreover, most of these packages, such as those from Orange Enterprises, are not intended to be modified by the user and are therefore rather inflexible.

A VAX-based programming language called 'SIMNON', developed by the Lund Institute of Technology of Sweden, has been developed solely for a complete dynamic model of a drive system including the controller. 'SIMNON' is considered to be a very user-friendly language [Bose,1985] and has a modular structure. It is intended to work with systems represented in state space (variable) form.

4.2.2 The MATLAB Software Package

4.2.2.1 Overview

MATLAB, which stands for "Matrix Laboratory", is an interactive, matrix-oriented software package that is ideally suited to analysis of linear circuits and systems, control theory, numerical analysis and other engineering applications. As the abbreviated name implies, the software handles data exclusively in matrix forms. It is an interactive system whose basic data element is a matrix that does not require dimensioning. The second-generation of MATLAB that runs on IBM compatible personal computers (PC) is called PC-MATLAB. Other versions of MATLAB that run on more powerful machines such as Sun Workstations are also available. The later versions of MATLAB have evolved beyond the matrix laboratory to become a versatile scientific "spreadsheet" for numerical calculation. These new versions of MATLAB are rapidly becoming a standard software

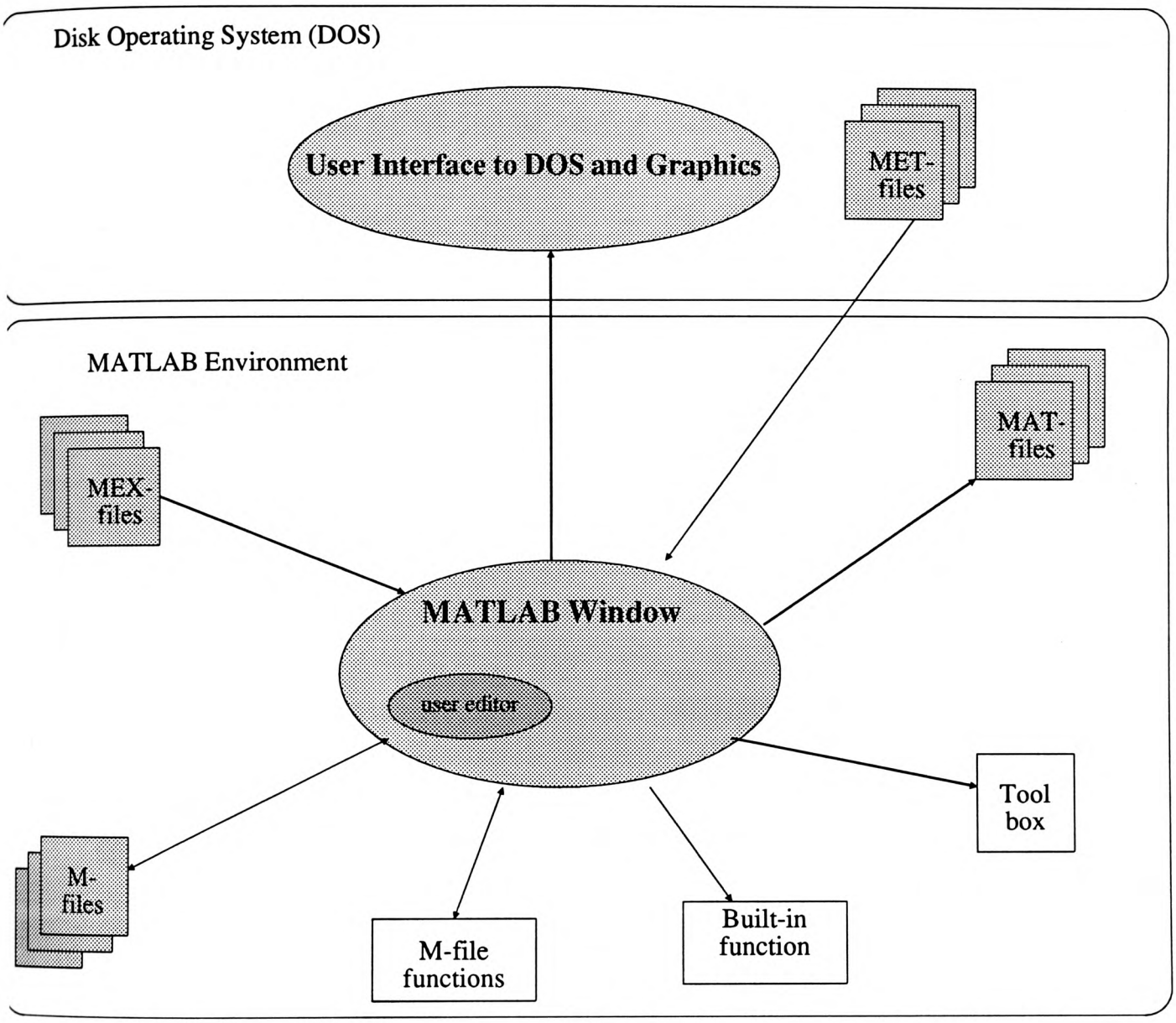


Fig.4.2 Operational environment of PC-MATLAB

package for the simulation of control systems for academic institutions and engineering companies.

The operating environment of MATLAB, depicted in Fig.4.2, allows the user to integrate matrix computation, numerical analysis, signal processing, and graphics in an easy-to-use form where problems and solutions are expressed in a form virtually as they are written mathematically, without the need of traditional programming technique. Basically there are four types of files — M-files, MAT-files, MET-files and MEX-files. Since these files are used in MS-DOS operating system and therefore their name have the format of `filename.ext`, where `filename` is the name of the file and the extension specifier, `ext`, usually denotes the type of file. MATLAB uses the name of the type of the file as the extension specifier. For example, an M-file may be called `text.m`, and an MET-file may be called `graph.met`. The M-file, which is in ASCII format, is run in MATLAB environment, and constitutes the MATLAB program and function files. It is with the M-file that a user may create his own customized library of functions. The MAT-file is used for storing all or part of the values generated for the variables during the running of MATLAB programs. This is necessary when there is not enough memory and some data has to be cleared. The data can be retrieved for later use. The MET-file is for graphic output of data. This file can be called upon by a command called 'GPP' (graphics post processor) in the DOS environment for the generation of graphics files of different formats. The MEX-file is a compiled, executable file in *C* or *Fortran*, which like the M-file, runs in the MATLAB environment. Additionally, there are built-in functions, standard matrix and associated functions in the M-file format to support the fundamental operations of MATLAB. The 'User Interface' to DOS can be invoked within the MATLAB environment by the exclamation command (!). The graphic output facility contains standard output formats such as Hewlett Packard Graphic Language (HPGL) and Encapulated Postscript (EPS) so that output data can be transferred easily.

The following sub-sections of this chapter are mainly devoted to feature parts of the MATLAB software package that are particularly relevant to the programs developed (see Appendix-B) for the generation of the simulation results presented later in this chapter. A complete description of MATLAB is found in the MATLAB user guide [MATLAB,1990].

4.2.2.2 Editing Environment

By design, MATLAB does not provide a built-in editor, the choice of editors being left to the user's personal preference. Depending on the size of memory available and the memory required to run the word-processor, it is found more convenient to run the editor without leaving the MATLAB environment by use of the exclamation command (!). Small processing editors work well with this method.

4.2.2.3 Matrix operations

Since MATLAB handles data exclusively in matrix form, an effective means is provided to express the data in matrix form. A matrix consists of a number of rows, where each row is expressed as a series of elements separated by a space, and each row by a semicolon (;) symbol.

For example, a 3x3 matrix $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$,

can be represented easily by the following expression:

```
A = [1 2 3; 4 5 6; 7 8 9]
```

The usual matrix operators bear the usual symbols found in the mathematical literature, and are tabulated in Table 4.1 as follows.

Matrix operator		Relational and logical operators	
+	addition	<	less than
-	subtraction	<=	less than or equal to
*	multiplication	>	greater than
/	right division	>=	greater than or equal to
\	left division	==	equal
^	power	~=	not equal
'	conjugate transpose	&	AND
		|	OR
		~	NOT

Table 4.1 Common Matrix operations of MATLAB

The conventional control loops in common programming languages such as the 'if-else-end', or 'for-end' are also found in MATLAB. However, these control loops are relatively inefficient when compared with the matrix operations [Schang,1991]. Although it may be easy to replace simple control loops with a matrix oriented expression, it may not be possible and/or efficient in terms of memory space required, when the control loop becomes more complex. The simulation programs written to generate the simulation results of the drive system investigated for the research reported in this thesis are evident examples of this. (see Appendix-B).

4.2.2.4 Modularity

One very important feature of MATLAB is the facility provided for the user to build functions and subroutines very efficiently by means of the M-files. As the future trend for the design of motor drives points towards integrating different parts of the drive system under a single development environment, it is strongly felt that the modularity of a development system will become a crucial feature.

4.2.2.5 Method of Spectral Analysis

Two Fourier transform commands are provided by MATLAB — the discrete Fourier Transform (dft) and the fast Fourier Transform (fft). Quantities represented in the time domain that are stored in a row or column matrix, T, can be transformed to the frequency domain and stored in another matrix, F. Thus,

F=fft(T) or F=dft(T)

The number of samples to be used in the 'fft' should be equal to 2^n(where n is an integer value), otherwise zeros are added to the data in order to make the total number of data equal to the next nearest power of 2 before the transform is performed. In the latter case, inaccuracy caused by the added zeros that do not represent the actual data, can be expected. Therefore, care has to be taken to ensure the number of input data samples is equal to a 2^n number. If the number chosen is restricted by the application, the 'dft' may

be used at a penalty in terms of computation time (a factor of about 100 for 1K data was recorded when run on an IBM 286- machine).

4.2.2.6 Toolbox Software

The Control System Toolbox, which is an additional software package to the main MATLAB package, uses MATLAB matrix functions, and builds upon the basic MATLAB environment to provide functions specialising in control engineering. The Toolbox is a collection of algorithms expressed in M-files, which implement common control system designs, analysis and modelling techniques. Only the model command function 'c2d' – continuous to discrete-time conversion, was used in the simulation programs, and the results of which are described later in this chapter.

The synopsis of the 'c2d' is: `[Phi, Gamma] = c2d (a, b, dt)`. It converts the continuous- time state-space system, in its standard form:

$$\dot{x} = Ax + Bu,$$

to its discrete-time form:

$$x(n+1) = \Phi\, x(n) + \Gamma\, u(n),$$

where the matrices Φ and Γ are exponential functions of A and B. It is also assumed that the control inputs are piecewise constant over the sample time `dt`.

4.3 DEVELOPMENT OF PWM DRIVE MODEL

4.3.1 Overview of Program Development

As the programs developed (full listings in Appendix-B) may be better understood by summarizing the main steps of the simulation, a generalized flowchart of the simulation programs is given in Fig.4.3. The program first inputs the initial data, such as the machine

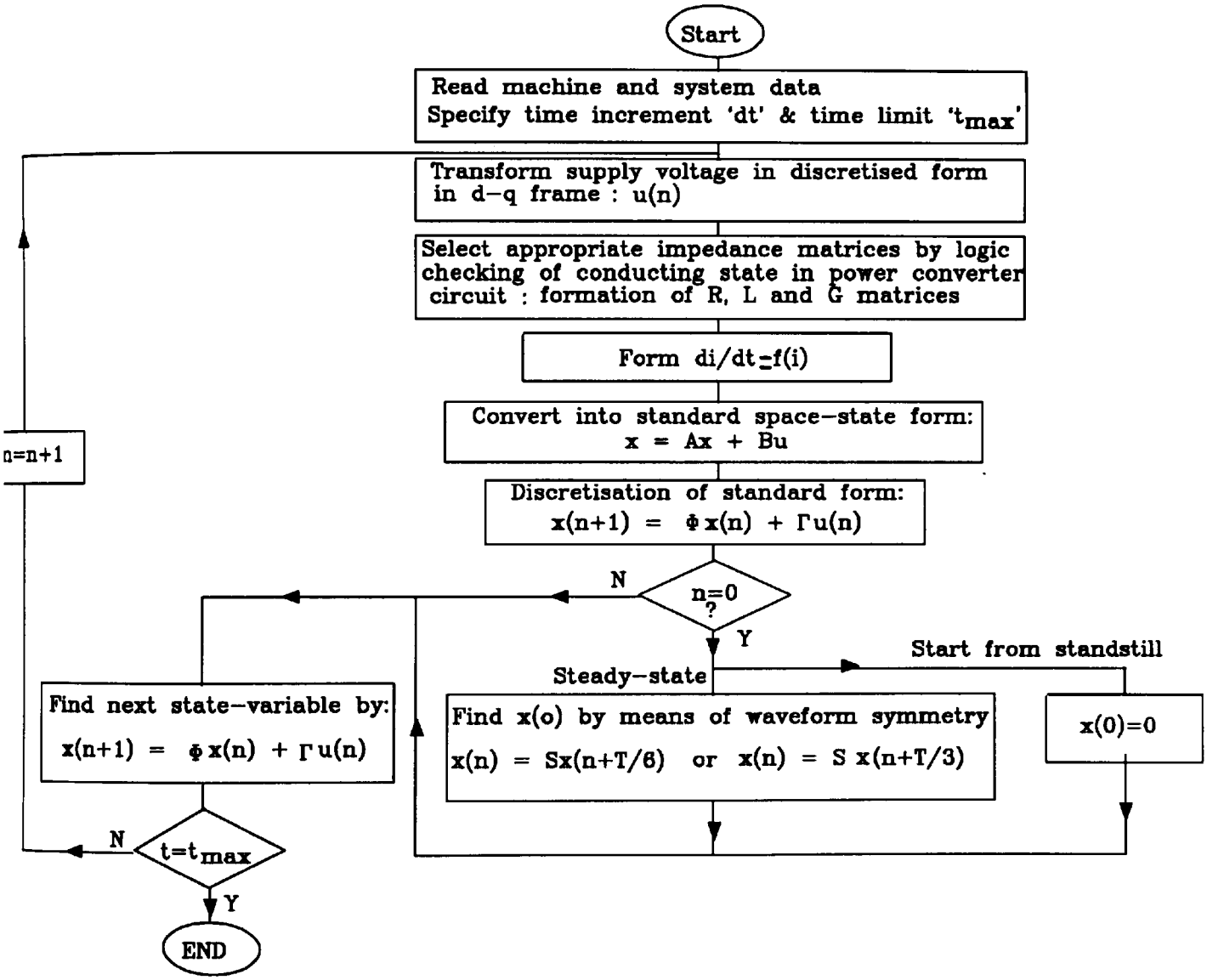


Fig.4.3 Generalized flowchart for simulation of motor drive by state-space method

parameters and the simulation period. The supply source, which may be sinusoidal or PWM, is transformed to discrete form as the input matrix $u(n)$, which may be represented in *d-q* or *A-B-C* frame. Logic checking on the conduction state of the semiconductor switching devices is then performed, which depends upon the type of the power device chosen and the power converter topology employed. The impedance matrices R,L and G are then determined. The motor equations, in matrix form involving R, L and G, are then transformed into the form $\frac{di}{dt} = f(i)$, where the derivative of the motor currents is a function of the motor currents. The motor flux may also be used in place of motor currents, and in such case a corresponding motor flux equation is resulted. The motor equation, which may be in currents or fluxes, is then transformed into the standard space-state form. The discretization of the standard continuous space-state form can be done by the MATLAB command 'c2d'. Two forms of initial conditions may be possible — firstly, when the motor is at standstill, and secondly when the motor is running at steady state. For standstill, the state variable $x(0)$, which represents the motor stator and rotor currents or fluxes, assumes zero values. When the motor is running at steady state, $x(0)$ can usually be found from voltage and current waveforms symmetry existing in the system. Once the initial value $x(0)$ is found, the next value can be determined from the discrete standard space-state equation. The process is repeated for the successive samples until t_{max}, the period over which simulation is to be made, is reached. The development of the digital model of the drive system has been greatly assisted by the modularity of MATLAB, which allows small modules to be created and tested independently. Larger systems can be built up by means of these small modules. The following sections describe the individual modules that make up the complete system of the model.

4.3.2 PWM Module

The PWM module consists of the switching algorithms employed to generate the signal PWM waveforms. To analyse the harmonic content of PWM waves by means of Fourier Transform and to solve the machine equations by the discrete space state method as described in the next section, it is necessary to obtain a digitized version of the continuous PWM waveform. The sampling process of the PWM waveform is depicted in Fig.4.4. The sampling period, T_S, of the sampling pulse chain should be small compared with the time

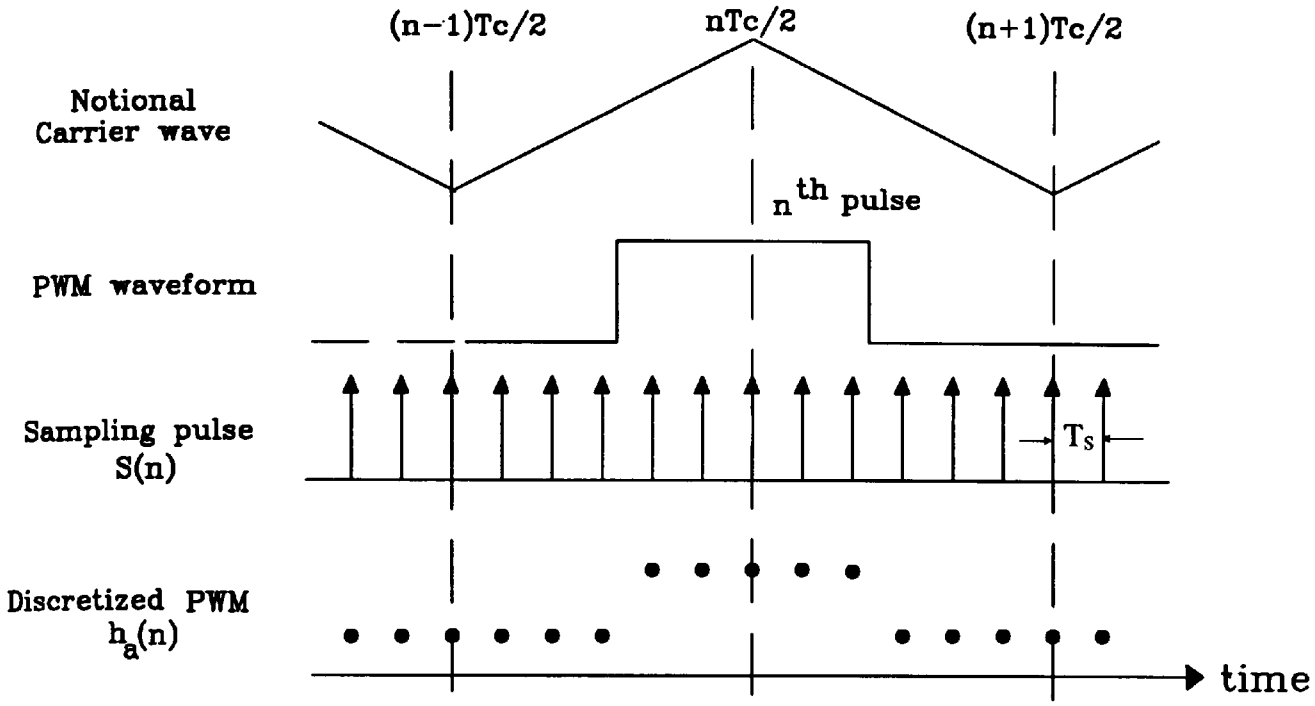


Fig.4.4 Discretizing process of PWM waveform

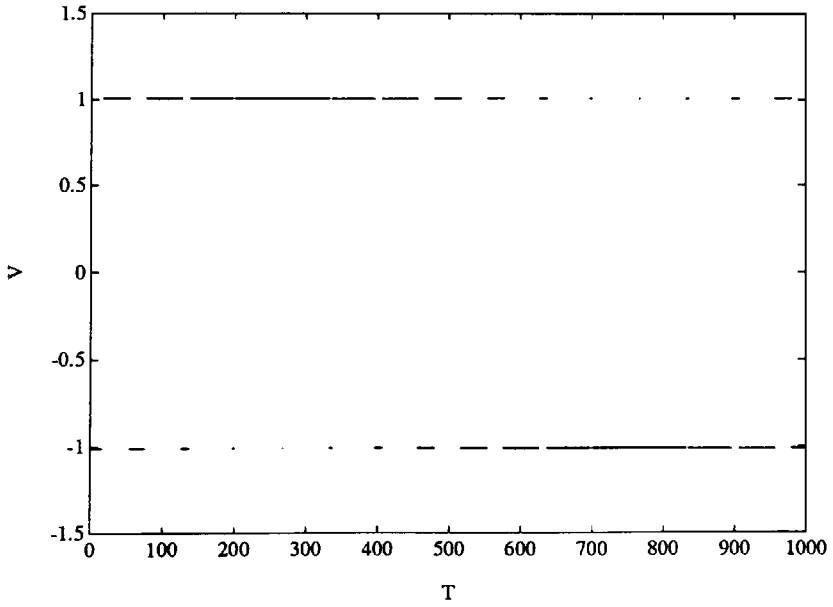


Fig.4.5 A discrete PWM waveform represented by MATLAB
(M=0.99, R=12, No. of sample/cycle=1000)

constants of the system, an order of 10 or higher is usually used. It can be seen from the Fig.4.4 that if the pulse-width is less than T_S then that particular pulse may not be sampled. For the regular sampled asymmetric modulated PWM waveform, and in fact for most of the PWM schemes, there is no switching at the instants corresponding to the apexes of the carrier waveform if the modulating index is less than unity. Thus, by synchronizing the sampling pulse chain with the apexes of the carrier wave, pulse-widths of less than T_S will be sampled. In MATLAB, switching times represented by a row vector, T, is of the form:

$$T = [T_S \ 2T_S \ 3T_S \ldots]$$

Similarly, the switching levels at the sampling points are represented by a row vector, V, of the form:

$$V = [1 \ 1 \ -1 \ \ldots]$$

The plotting of matrix V against matrix T then generates the digitized PWM waveform shown in Fig.4.5, where a frequency ratio R of 15, a modulation index M of 0.99, and a sampling frequency of 1000 samples/cycle are used. It is noted that all pulses of duration widths less than T_S are sampled with this method.

4.3.3 Power Module

If the effect of power semiconductors are included in the simulation, then it will require a logic check, at each time increment, to find the conducting state of every circuit branch including such switching devices. The switching behaviour of the devices, as mentioned in chapter 2, introduces complexity in the drive. However, a general approach, summarized below, can be adopted [Hindmarsh,1982].

(i) A diode will conduct if the forward voltage drop across it is equal to or greater than 1 V.

(ii) For a GTO or power transistor, a firing pulse must be present for conduction, and the forward voltage drop must be greater than 1 V.

(iii) For a thyristor in a naturally-commutated system, conduction will continue if it was conducting in the previous conducting time increment, with a current greater than the holding value - 100 mA; or it will conduct if the forward voltage drop is greater than 1 V, and a firing pulse is present.

(iv) For a thyristor in a forced-commutated system, a GTO or a power transistor, its switch-off time can be specified. A MOSFET turns fully ON when the gate voltage is about 10V, and turns OFF when it drops below the threshold voltage, typically 3V.

If no conduction is detected, it may be necessary to calculate the voltage across such an element arising from the rest of the circuit and applied to this element in the simulation, so that its current will calculate zero. It may be possible, with certain types of simulation to set this semiconductor's impedance to several thousand ohms, and solve the circuit without mathematical instabilities.

In some cases, if the switching devices behave close to an ideal switch, the logic checking stage may be omitted without causing appreciable errors. For example, if the MOSFET is driven by a gate driver with a high capability of sinking or sourcing the gate current, and is operating at a low frequency, then the PWM signal waveform will be very similar to the PWM power waveform. In such a case, it is justified and instructive to neglect the logic checking stage.

4.3.4 Motor Module

4.3.4.1 Motor Equation in State Variable Form

From the previous chapter the machine equation can be written in standard state-variable form as follows:

$$\dot{x} = Ax + Bu \tag{4.1}$$

where x is the state variable of the transformed current vector and u is the PWM input vector to the d-q frame; and A, B are as defined in chapter 2. It should be noted that the values of the matrices A, B of the motor can be determined from the motor parameters. The manufacture's data of the motor under investigation is found in Appendix-F.

The discrete form of equation (4.1) is given by,

$$x(n+1) = \Phi x(n) + \Gamma u(n) \tag{4.2}$$

where, $\Phi = e^{AT_S} = I + A\,T_S + \frac{1}{2!}A^2 T_S{}^2 + \ldots$, I = *Identity matrix*

$$\Gamma = \int_0^{T_S} e^{A\tau} d\tau B$$

Since the PWM input $u(nT_S)$ over the period, $nT_S < \tau < (n+1)T_S$, is constant, then,

$$\Gamma = [I\,T_S + (\frac{1}{2!})A\,T_S^2 + \ldots]\,B$$

The transformation from equation (4.1) to (4.2) is achieved by the command '`c2d`' (see section 4.2.2.6) provided by MATLAB. The sampling time T_S is chosen to be the same as that used in the discretization of the PWM waveform.

4.3.4.2 Initial Condition

For a complete solution of equation (4.2), it is necessary to find the initial value of the state variable, $x(0)$, which represents the motor currents or fluxes at $n=0$. Since the residual flux of the machine is usually small, and therefore, for the purpose of simulating the dynamics of the drive system, it can be neglected without causing observable effects. If the simulated drive system is started from a standstill condition, the initial state variable, $x(0)$, is then simply equal to zero. If only the steady state operation of the drive system is simulated, $x(0)$ is no longer equal to zero and must be determined. Assuming a three-wire balanced source supplying the motor drive, the value of $x(0)$ can be found by the inherent symmetry of the resulted motor voltage and current waveforms. Depending on the symmetry of the waveform, one of the following relations may be used [Bowes,1983]:

For waveforms with no half-wave symmetry,

$$x(t + T/3) = S_1 x(t) \tag{4.3a}$$

And for waveforms with half-wave symmetry,

$$x(t+T/6) = S_2 x(t) \tag{4.3b}$$

where T is the period of the waveform, and S_1 and S_2 are as defined in the reference [Luk,1991a].

Thus, for a waveform with half-wave symmetry, if the p^{th} sample is at $T/6$, then from equation (4.2), the following equations may be obtained:

$$x(1) = \Phi x(0) + \Gamma u(0) \tag{4.4a}$$

$$\vdots$$

$$x(i) = \Phi x(i-1) + \Gamma u(i-1) \tag{4.4b}$$

$$\vdots$$

$$x(p) = \Phi x(p-1) + \Gamma u(p-1) \tag{4.4c}$$

By multiplying equations (4.4a), ...(4.4b), ...(4.4c), with Φ^{p-1},... Φ^{i},...,1 respectively, and through the process of summation and elimination the following equation for $x(0)$ can be obtained.

$$x(0) = [S_2 - \Phi^p]^{-1} \sum_{i=0}^{p-1} \Gamma \Phi^i u(p-1-i) \tag{4.5a}$$

A similar expression exists for a waveform with no half-wave symmetry:

$$x(0) = [S_1 - \Phi^p]^{-1} \sum_{i=0}^{p-1} \Gamma \Phi^i u(p-1-i) \tag{4.5b}$$

Once the initial vector $x(0)$ is obtained, the complete solution for the state equation of the motor currents can then be found as a matter of routine by means of equation (4.2). A typical PWM voltage waveform that is assumed to have half-wave symmetry over one-sixth of its cycle is shown in Fig.4.6. The discrete sampled currents in the d-q frame are state variables whose values can be determined by the discrete state variable equation

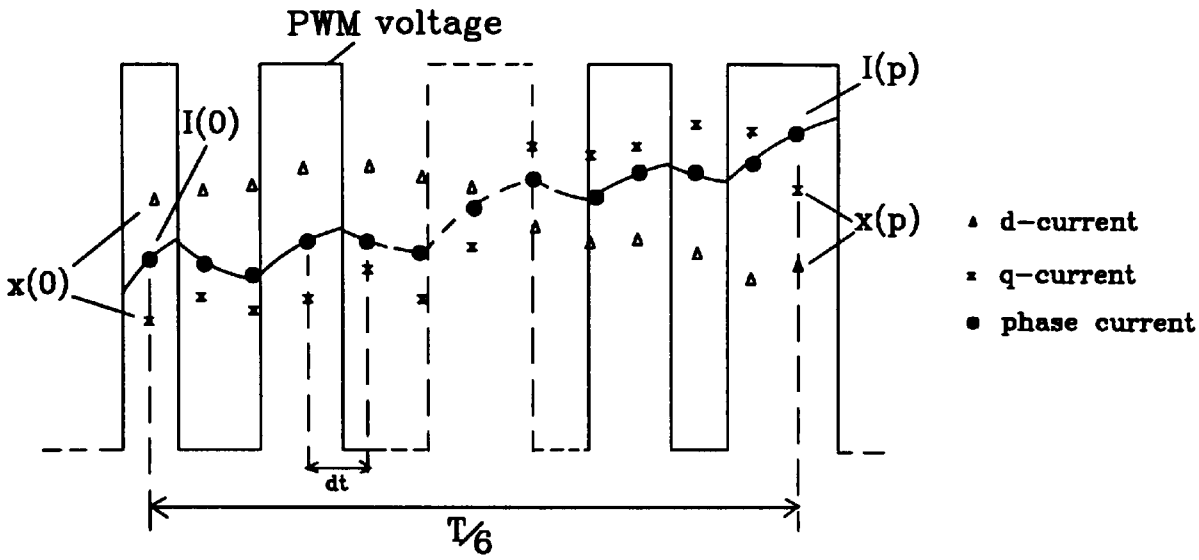


Fig.4.6 State variables and phase current for a PWM waveform with half-wave symmetry

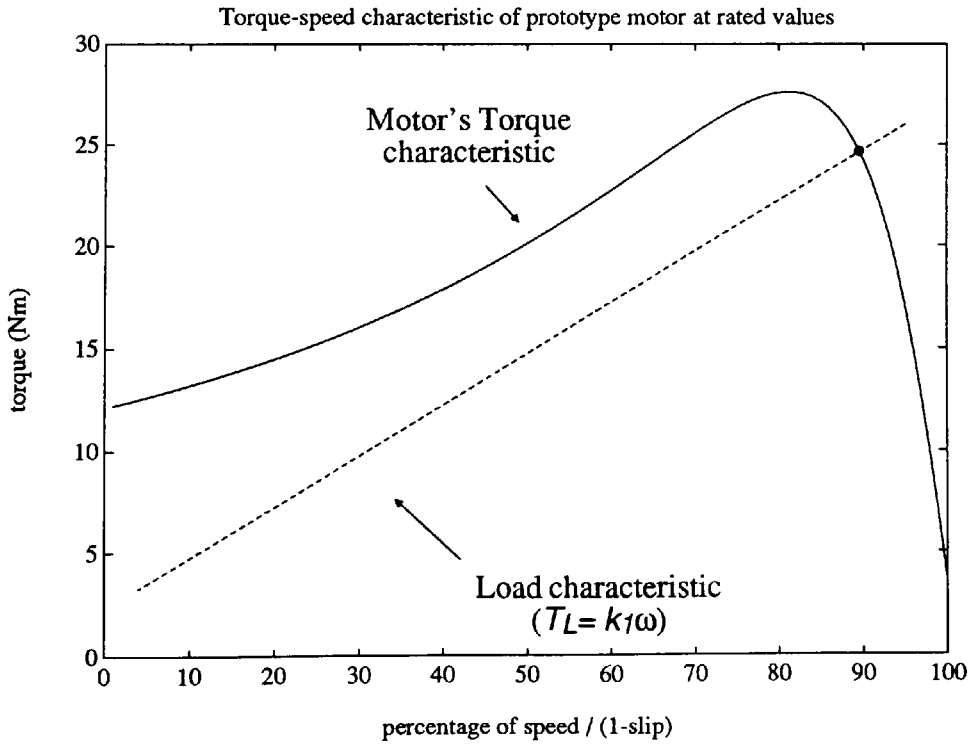


Fig.4.7 Torque-speed characteristic of prototype induction motor (2.2 kW, 4-pole, 3-phase caged-rotor type)

(4.2). The discrete phase currents of the motor $I(0)..I(i)..I(p)$ can be found by the inverse transformation of the discrete state variables $x(0),..x(i),..x(p)$.

4.3.5 Load Module

The **Load** module consists of a set of mechanical load characteristic curves. The analysis of load torque can be rather involved because most loads comprise torques of different origins. The coupling devices, for example, are characterized by nonlinear elasticity and backlash, whereas the viscous torque in many applications is an approximately quadratic function of the speed. Most industrial loads, however, can be represented by the following expression with acceptable accuracy:

$$T_L = k_0 + k_1\omega + k_2\omega^2 + ... \qquad (4.11)$$

where T_L is the load torque, ω is the load speed, k_0 is the coefficient of static friction and k_1, k_2 ... are positive constants. The steady state operating point is determined by the intersection of the motor torque characteristic and the load curve, as given in Fig.4.7.

4.4 RESULTS OF SIMULATION OF DRIVE SYSTEM BY MATLAB

4.4.1 Direct On Line Start-up with Sinusoidal (mains) Supply by *A-B-C* Model

In the *A-B-C* model, the mutual inductance of the motor varied with the rotor position. Thus a new impedance matrix has be to computed at the next sampling point when the motor rotates to a new position. The program was developed from a model described in [Hindmarsh,1982]. The inspection of the program (see Appendix-B), shows that a number of $6x6$ matrices have to be computed for every time increment. Each of these matrices consists of functions of sine and cosine. The results of simulation for direct-on-line (DOL) start-up of the motor used in the investigation on a 240 V mains supply are shown in Fig.4.8-10. Since the same simulation results were also obtained when the *d-q* model was

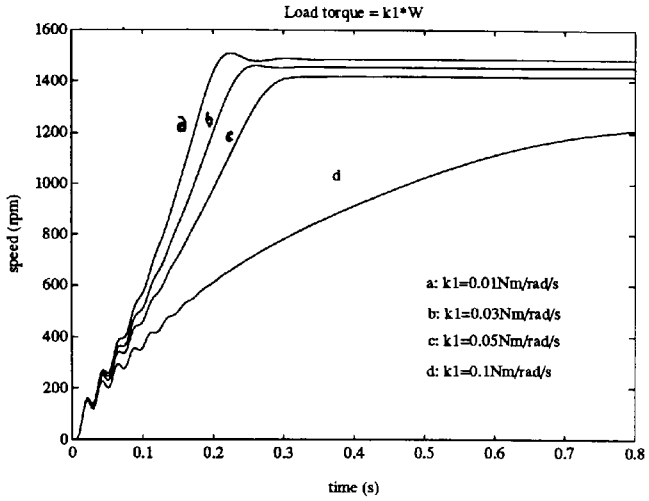


Fig.4.8 Speed profiles at different loads during DOL start-up

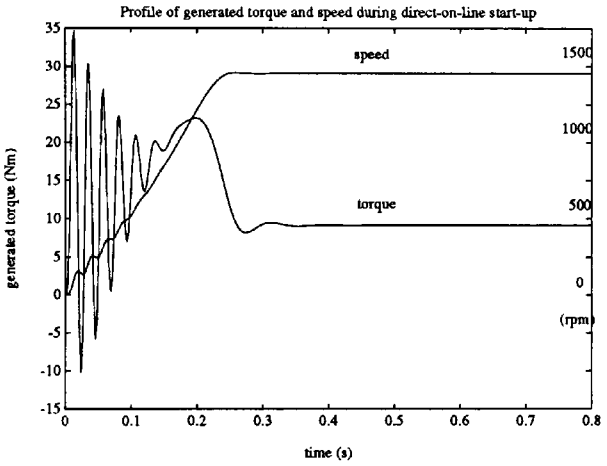


Fig.4.9 Profiles of torque and speed during DOL start-up

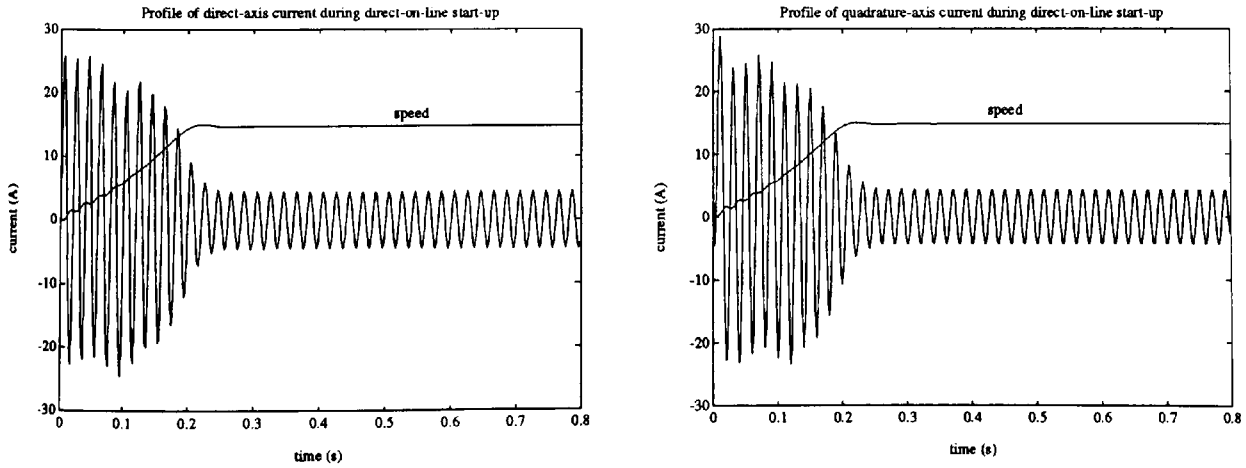


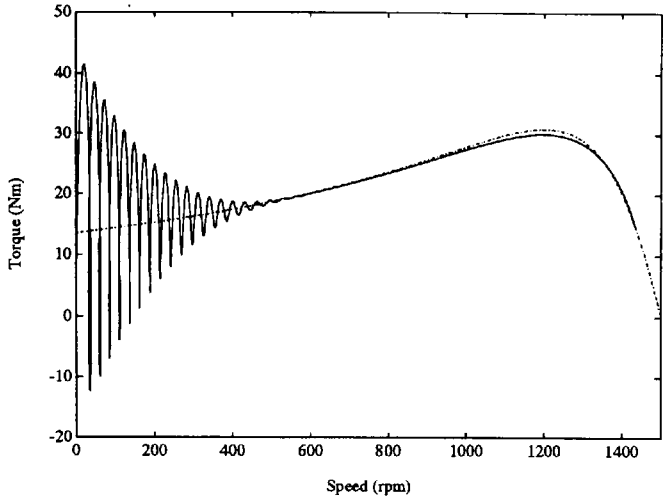
Fig.4.10 *d*-*q* currents during DOL start-up
(a) *d*-current (b) *q*-current

used in place of the *A-B-C* model, these results are discussed together in the following section.

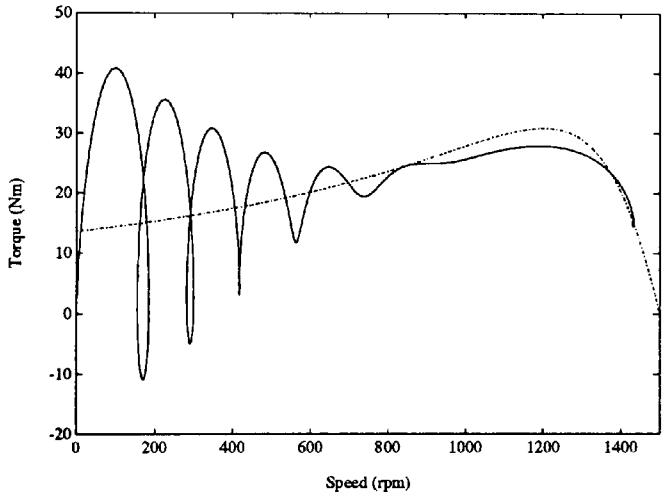
4.4.2 Direct On Line Start-up with Sinusoidal (mains) Supply by *d-q* Model

Since the inductance of the motor in *d-q* frame is constant irrespective of the rotor position, the impedance matrix of the machine equation is therefore constant. This reduces the computation time because the impedance matrix needs to be computed only once. Moreover, the algorithm is computationally more 'stable' and accurate because of the absence of the sine and cosine functions. The simulated results for the motor speed during DOL start-up when the motor model was supplied by a three-phase 240V mains at different loads (with a characteristic curve $T_L = k_1\omega$) are shown in Fig.4.8. The corresponding generated torque and the *d-q* currents at one of the specific loads are shown in Fig.4.9 and Fig.4.10 respectively.

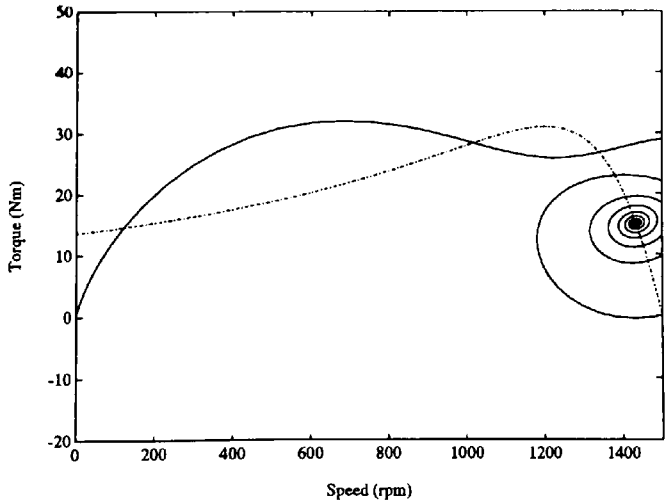
To further investigate into the cause and effect of the torque transient and the associated speed oscillation during DOL start-ups, the drive model was 'loaded' with three loads (of type $T_L = k_1\omega$) but of different inertia. The values of the constants k_1 used were such that all three loads were subjected to the motor rated torque when steady state condition was attained. The simulated results of torque against speed during a DOL startup are shown in Fig.4.11(a)-(c) respectively. It should noted that such a setup of simulation test, which does not appear to be discussed in the literature, could in fact be used to demonstrate how the well- known steady state torque-speed characteristic of the induction motor, which is shown in the dotted lines in Fig.4.11, relates to its transient counterpart. The profile of the generated torque during start-up can be explained by the unique feature of the induction motor due to the effect of the rotor motion, as described in section 2.2.1. In Fig.4.11(a) where the load inertia is highest, the rotor can be assumed to be virtually at rest during the first few cycles after start-up. The motor then acts like a transformer with secondary terminals being short circuited. The resulting rotor currents, which are driven by the induced secondary voltage, interact with the stator flux to produce a pulsating torque. When the rotor speed increases, the effect of motional EMF becomes more apparent and the transformer effect diminishes. Due to the high load inertia, the motor can be described as at 'pseudo steady state' at some time after the startup, when the



(a) Inertia = 0.5 kg/s/s



(b) Inertia = 0.01 kg/s/s



(c) Inertia = 0.001 kg/s/s

Fig.4.11 Transient characteristic of speed against torque during DOL start-up

transformer effect becomes negligible. Thus, the dynamic characteristic coincides approximately with that of the steady-state over this range of speed. In Fig.4.11(b), where the inertia is medium, larger oscillations can be seen from the 'loops' of the curve. The motor required less time to reach the 'practical' steady state speed when compared with the previous case. It can be seen that the transient characteristic does not 'follow' the steady state curve, but they meet at the steady state operating point. In Fig.4.11(c), where the inertia is lowest, the motor accelerates rapidly and oscillates about the steady state point. Thus, despite the high acceleration, it may take a number of overshoots and hence longer time before steady state is reached. The above results can be extrapolated to suggest that when the inertia approaches zero, the transient characteristic approximates the load line, as depicted in Fig.4.7.

4.4.3 Comparison of the *A-B-C* and *d-q* Models

Historically, the choice between the *A-B-C* and *d-q* models was quite controversial. The recent availability of high processing power computers and specialised software packages further influence the debate. The computation times for the two methods for different sampling size are tabulated in Table-4.2. The intrinsic feature of the MATLAB to handle matrix manipulations and the built-in functions to solve the trigonometric functions have improved the computational time of the direct simulation in *A-B-C* frame. For example, it can be seen from Table 4-2 that, by using MATLAB, the *A-B-C* method only takes about 26.8 % more time than the *d-q* method with a sample size of 200, whilst a much higher percentage (more than 100%) was recorded when a general purpose program was used instead. Furthermore, since MATLAB is much more efficient to run vector-oriented data than the `for-end` loops, as the number of `for-end` loops increases with the sample size, the effect of these loops on the overall computational time becomes dominant and the relative simulation time (p/q) converges to unity. However, since the accuracy of the *d-q* method is always higher than that of the *A-B-C* method for a given number of samples due to a lesser number of computations required, the *d-q* method was used for subsequent simulations.

Sample size (N)	Unit time for *A-B-C* model (p)	Unit time for *d-q* model (q)	Relative time (p/q)
200	208	164	1.268
400	418	373	1.255
800	638	518	1.232
1600	1326	1142	1.161

Table 4.2 Comparsion of simulation times for the ***A-B-C*** and ***d-q*** models

4.4.4 Direct On Line Start-up with PWM Supply by *d-q* Model

The simulation results of DOL start-up when the motor model was supplied by a regular sampled asymmetric PWM source at different loads are shown in Fig.4.12. The fundamental frequency of the PWM waveform is 50 Hz and the modulation index is 0.99. For the purpose of illustration, a constant carrier frequency of 1000 Hz has been used. However, the technique of 'gear-changing', which is commonly used in practice, can be readily incorporated into the model by changing the carrier frequency at certain speed regions. The loads used were of the characteristic equation $T_L = k_1\omega$. The corresponding generated torque and the line current at a certain load are shown in Fig.4.13(a)-(b) respectively.

The motor flux can be readily controlled by the voltage-to-frequency ratio in a PWM inverter. In Fig.4.14, it shows the torque and speed profile during a DOL startup when supplied by a PWM waveform with a modulating frequency of 20 Hz and a modulation index of 0.4. It is evident that the speed and torque profiles are similar to the tests in the previous section, where sinusoidal supply is used. Nevertheless, the magnitude is reduced because of reduced DC link voltage used and also the influence of the harmonic torques generated. The harmonic torques are also responsible for the torque ripples at steady state. The sampling rate for PWM simulation was found more crucial than in the case of sinusoidal supply. This was possibly due to the nature of the PWM waveforms, which only take discrete voltage levels. The results of some trial runs have suggested that a sampling rate of 10 samples/carrier cycle is optimal in terms of computation time, accuracy and memory requirement for display of results.

transformer effect becomes negligible. Thus, the dynamic characteristic coincides approximately with that of the steady-state over this range of speed. In Fig.4.11(b), where the inertia is medium, larger oscillations can be seen from the 'loops' of the curve. The motor required less time to reach the 'practical' steady state speed when compared with the previous case. It can be seen that the transient characteristic does not 'follow' the steady state curve, but they meet at the steady state operating point. In Fig.4.11(c), where the inertia is lowest, the motor accelerates rapidly and oscillates about the steady state point. Thus, despite the high acceleration, it may take a number of overshoots and hence longer time before steady state is reached. The above results can be extrapolated to suggest that when the inertia approaches zero, the transient characteristic approximates the load line, as depicted in Fig.4.7.

4.4.3 Comparison of the *A-B-C* and *d-q* Models

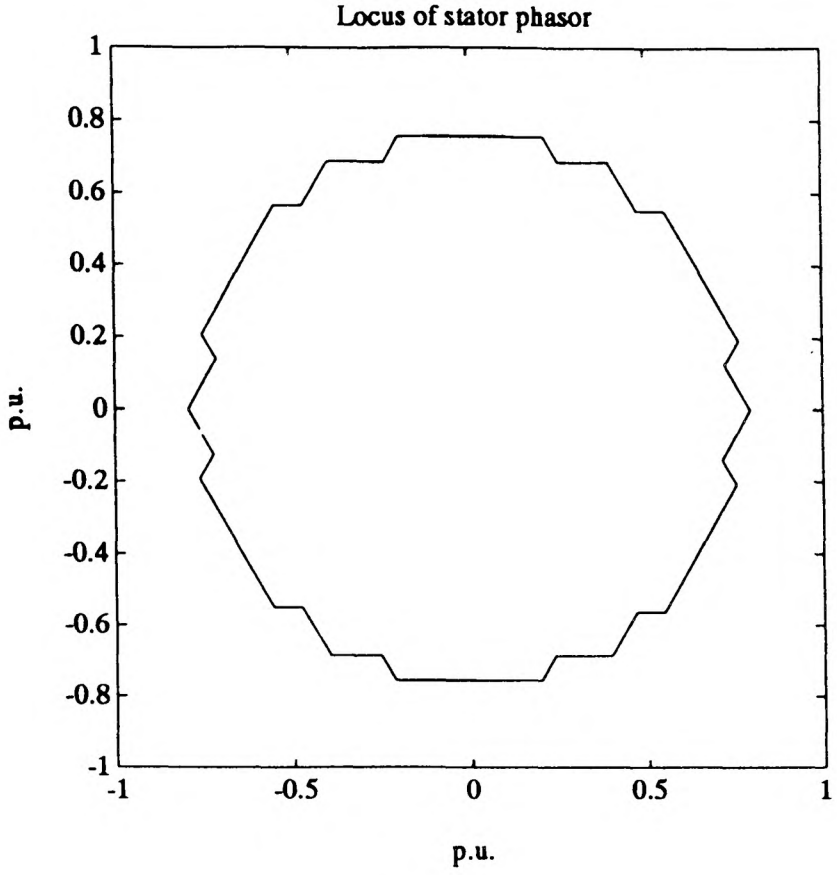
Historically, the choice between the *A-B-C* and *d-q* models was quite controversial. The recent availability of high processing power computers and specialised software packages further influence the debate. The computation times for the two methods for different sampling size are tabulated in Table-4.2. The intrinsic feature of the MATLAB to handle matrix manipulations and the built-in functions to solve the trigonometric functions have improved the computational time of the direct simulation in *A-B-C* frame. For example, it can be seen from Table 4-2 that, by using MATLAB, the *A-B-C* method only takes about 26.8 % more time than the *d-q* method with a sample size of 200, whilst a much higher percentage (more than 100%) was recorded when a general purpose program was used instead. Furthermore, since MATLAB is much more efficient to run vector-oriented data than the `for-end` loops, as the number of `for-end` loops increases with the sample size, the effect of these loops on the overall computational time becomes dominant and the relative simulation time (p/q) converges to unity. However, since the accuracy of the *d-q* method is always higher than that of the *A-B-C* method for a given number of samples due to a lesser number of computations required, the *d-q* method was used for subsequent simulations.

4.4.5 Steady State with PWM Supply by *d-q* Model

If the start-up simulations described in section 4.4.4 were allowed to run longer than the transient period, and the other parameters remained unchanged, the steady state condition could be obtained. Alternatively, the steady state condition can be obtained from the symmetry existing in the motor waveforms, as described previously in section 4.3.5. The motor flux in *d-q* form, when driven by PWM voltage waveform with frequency at 50Hz, frequency ratio at 21, and modulation index at 0.99, is shown in Fig.4.15. The steady state simulation of the flux behaviour of the induction motor, when supplied with a PWM waveform modulating frequency of 50 Hz, frequency ratio of 21, and modulation index of 0.99 is illustrated in Fig.4.15. It is noted that the waveform of the simulated flux is in close agreement with the actual flux obtained from the experimental motor. The experimental flux waveform was determined by means of a circuit technique which, together with other details of the experimental setup, is found in Appendix-H. Both of the simulated and experimental fluxes can also be illustrated by means of the locus of the flux vector. For example, when the flux is purely sinusoidal, then the locus of the flux vector is a perfect circle. Thus, by this method of representation, one can represent the motor flux when the motor is supplied with a PWM voltage waveform of frequency at 50Hz, frequency ratio at 12, and modulation index at 0.99, as shown in Fig.4.16, where it can again be seen that the simulated flux compares favourably with the experimental flux.

4.4.6 Harmonic Analysis

The results of the harmonic analyses presented in this thesis were all performed by means of the Fast Fourier Transform `'fft'` routine of MATLAB using a sample size of 1K (1024). The Discrete Fourier Transform `'dft'` of MATLAB was also performed for comparison purposes. It was found that if the number of samples was a power of 2, then both methods gave the same result. However, the `'fft'` method brought about an improvement of the order of about 10 in computing time when compared with the `'dft'` method.



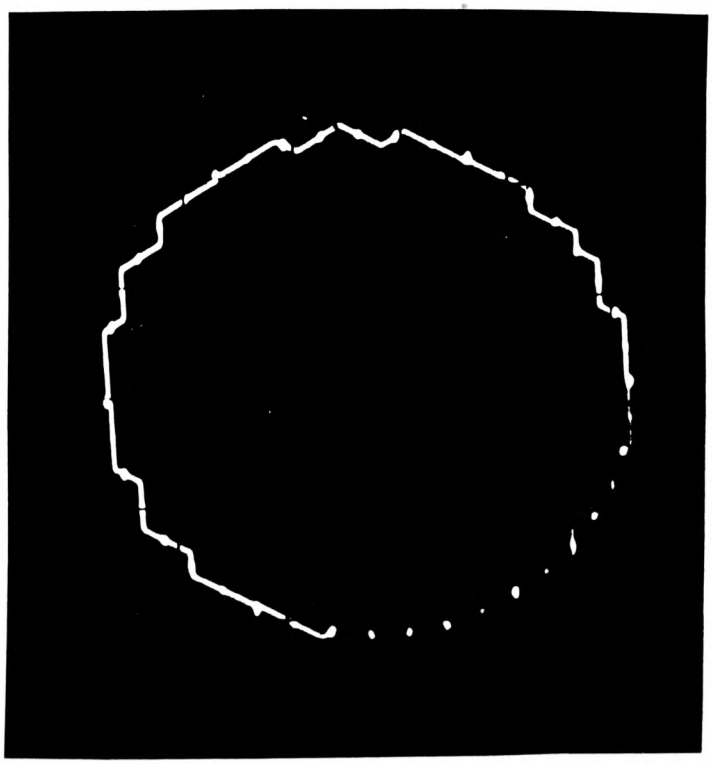
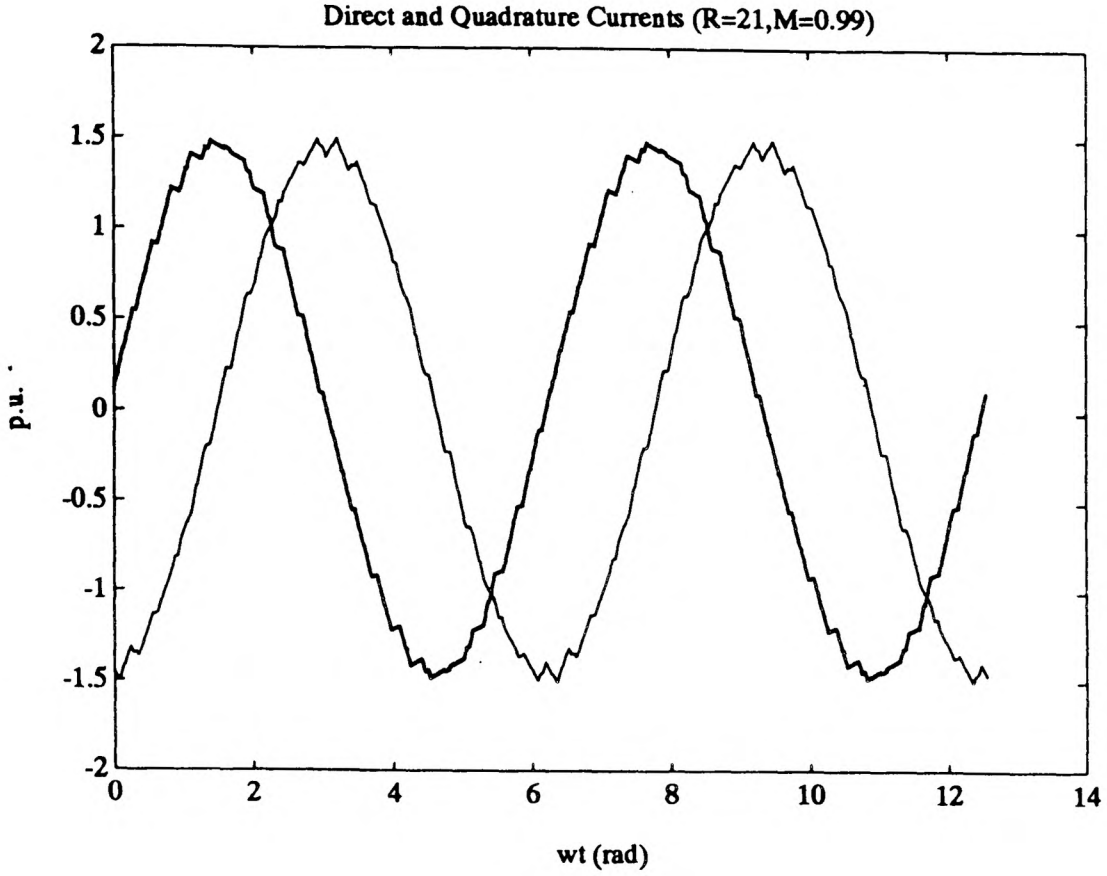


Fig.4.15 Motor flux in *d-q* form at steady state
(a) Simulation (b) Experimental



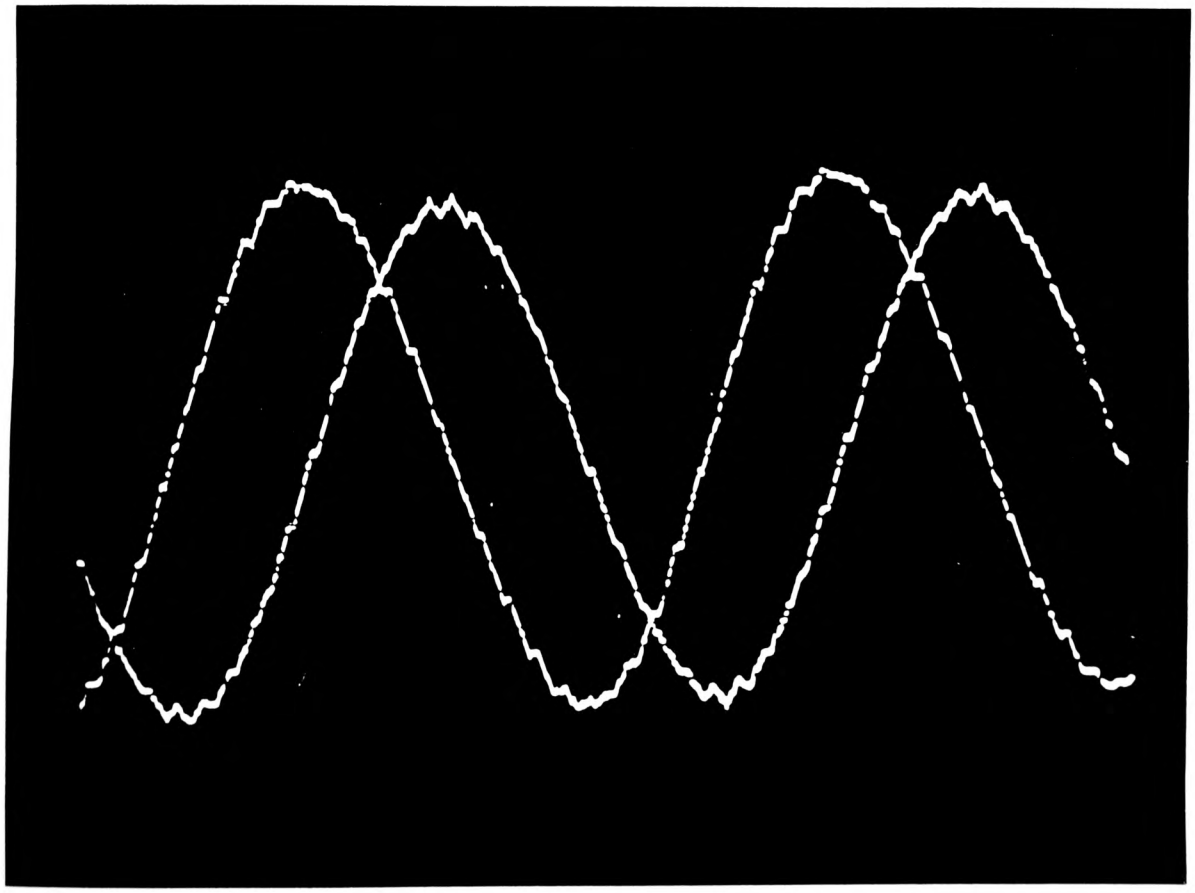


Fig.4.16 Motor flux phasor at steady state
(a) Simulation (b) Experimental

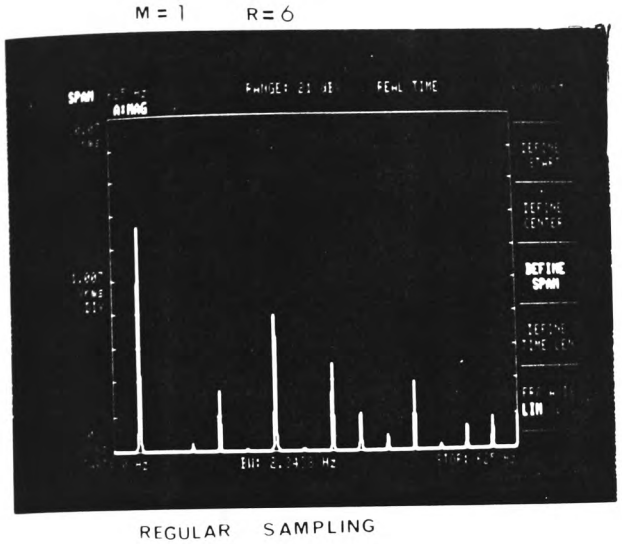
The harmonic spectra of two typical regular sampled asymmetric PWM waveforms are compared with the experimental results, in Fig.4.17 and Fig.4.18 respectively. The experimental PWM waveforms were generated by methods previously described in section 3.3.8. It can be seen that the simulation results closely agree with the experimental result.

4.4.7 Vector Control

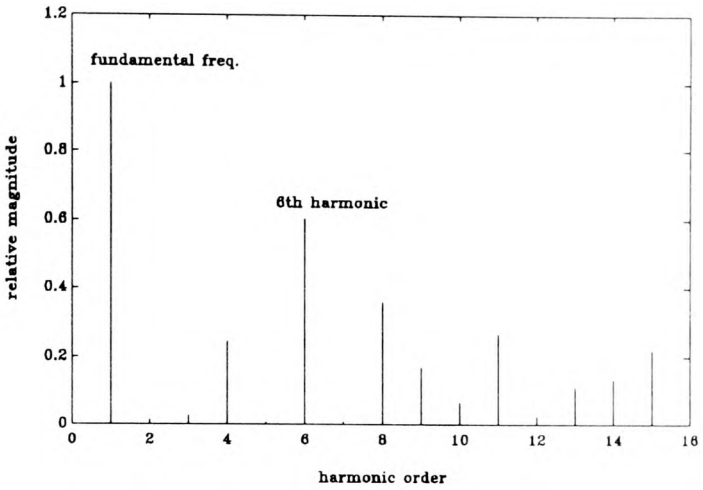
The vector control strategy developed by Harashima which is illustrated in Fig.3.15 of chapter three, was incorporated in the drive model. The simulation tests of the resulting system were carried and examined at several conditions. In the first test, the supply voltage was supposed to be from a sinusoidal source with no limit imposed on the output voltage and current magnitudes. The gain of the proportional controller was set to 0.2 Nm/rpm, and the step input of speed was 800 rpm. The dynamic response of the speed of the motor and the torque generated is shown in Fig.4.19. This is compared with the result of a normal direct-on-line startup, where the speed is indicated by the dotted line. The dynamic response is seen to be improved significantly as a result of the application of the vector control. The generated torque in the vector control system, in contrast to the oscillating torque as in the previous system, is unidirectional and has a higher average value during the acceleration period. This is consistent with the theory that vector control aims to decouple the torque component and field component of the motor currents, so that maximum torque is acheived at any time.

In the second investigation, it was assumed that the voltage and current values were finite and was therefore more representative of a real industrial situation. The simulated results from these conditions are illustrated in Fig.4.20, where it may be seen that the dynamic response of the system is inferior to that for the previous test. For example, the response time is approximately 0.2 s when previously it was 0.1 sec.

The third operating condition to be investigated was when the motor was supplied with a PWM source. A constant carrier frequency would help to simplify the model. Since the response time of the system is shorter, it is very important to use a short sampling period

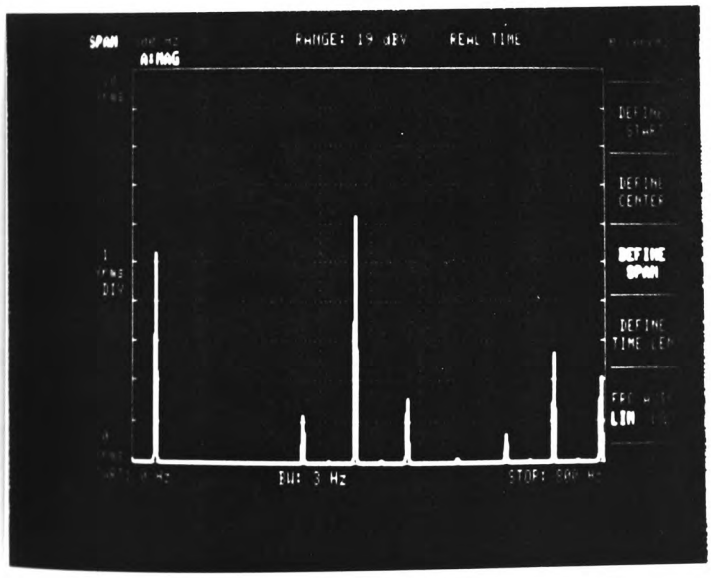


(a) Experimental

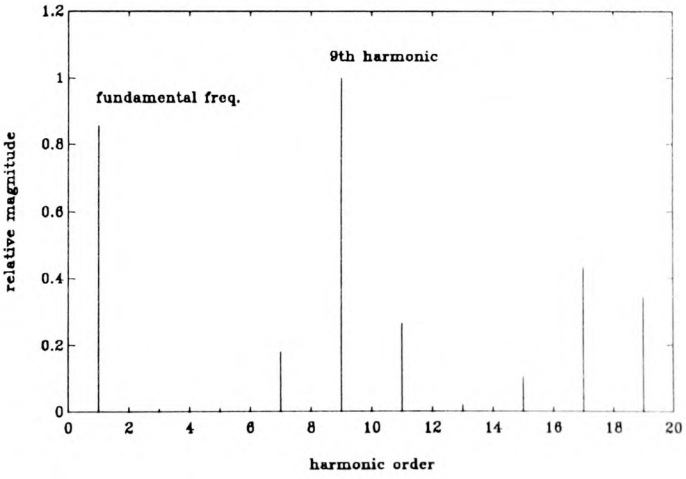


(b) Simulation

Fig.4.17 Harmonic spectrum of a single phase PWM waveform with f = 50Hz, M =1.0, R = 6.



(a) Experimental



(b) Simulation

Fig.4.18 Harmonic spectrum of a single phase PWM waveform with f = 50Hz, M=0.75 and R=9.

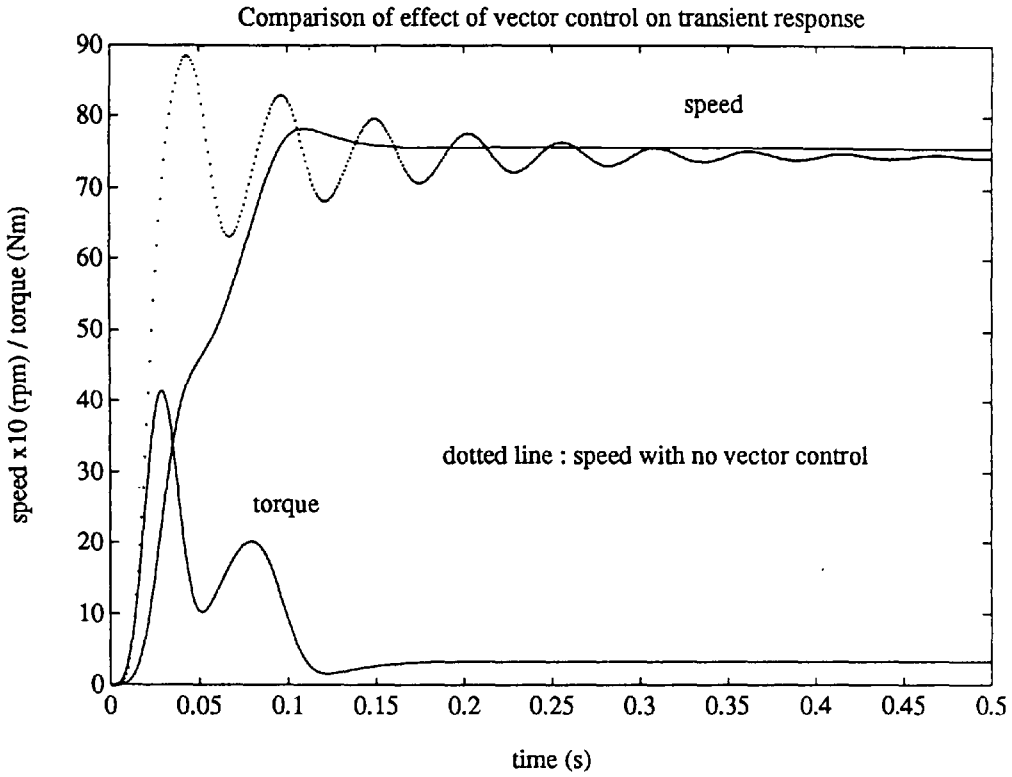


Fig.4.19 Simulation result of vector control
for the response of a step input of 800 rpm at a load of 0.02 Nm/rad/s

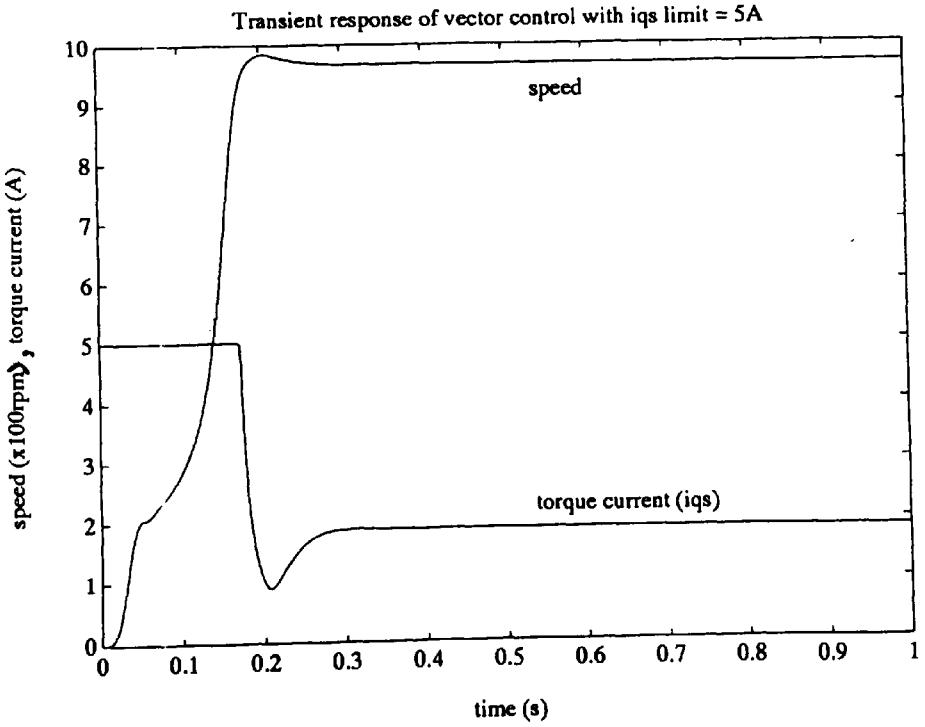


Fig.4.20 Simulation result of vector control
when current limit of power source is imposed

dt in the discretization of the model. Failure to adopt a suitable sampling time will lead to very misleading or wrong results. An exact modelling of the motor flux and the motor speed during the transient period is very involved and beyond the scope of the thesis. The effect of the harmonic content of the PWM waveform would result in degrading the dynamic response of the system. The result of the vector control with a PWM power source, which is more likely than a sinusoidal source in practice, is shown in Fig.4.21. Due to the complexity of setting up the necessary instrument and hence the time incurred to obtain a corresponding experimental result, the simulation result of Fig.4.21 has not been able to verified experimentally. It should also be noted that the assumption of a DC link supply with unlimited sourcing and sinking capability has been made. The simulation result should therefore be intrepreted with such allowance. However, it should be emphasized that the simulation result shows the improvement of dynamic response as a result the application of vector control.

4.5 INTERIM CONCLUSION

In the discrete state space equation, the value of the initial condition $x(0)$ and also the subsequent values of $x(1)$, $x(2)$, ..., affects the outcome of the complete solution of the equation. Therefore, the steady state results of the motor model in section 4.4.4 can be extrapolated to establish the validity of the model in dynamic state as well. The model can be used in the design of the controller, and in the investigation of more advanced control strategies such as sliding mode control and adaptive control.

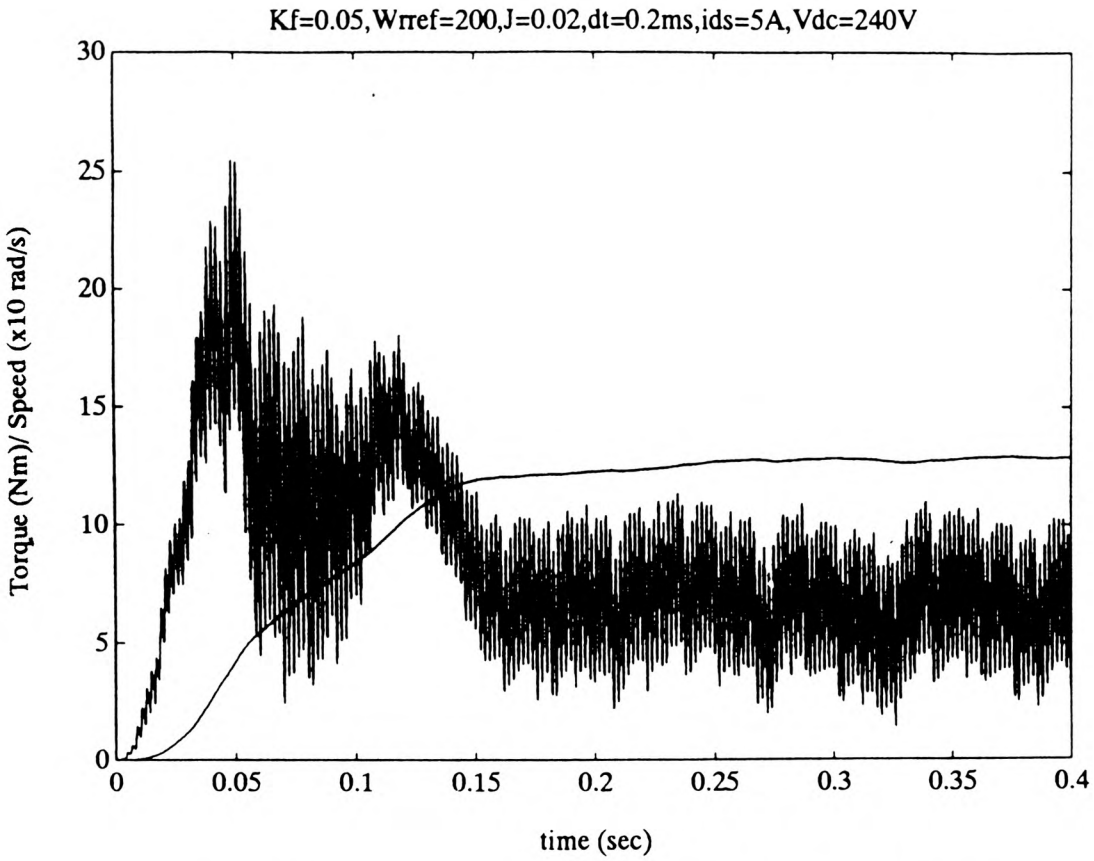


Fig.4.21 Simulation result of vector control with PWM source

Chapter 5 : Power Conditioning Stage of Induction Motor Drives

5.1 INTRODUCTION

It has been shown in chapter 3 that the adjustable speed AC drive system requires a power converter output stage that is capable of generating variable voltage, variable frequency (VVVF) power waveforms. A number of methods available for the generation of different PWM waveforms are also discussed in chapter 3. The PWM waveforms are low power signals, usually at logic levels, and have to be translated to a power level to drive the motor. The power conversion process can be achieved by the power inverter. The frequency range and output power are two important specifications for an inverter. The frequency range defines the maximum speed attainable from a given machine. AC machines used in the aircraft, for example, operate at a frequency of up to 400 Hz. The output stage to drive such motors is required to deliver the power at this frequency. This requirement not only imposes a constraint on the switching strategy to be used, but also affects the choice of the switching power devices.

In this chapter, a review of the recent development in power semiconductor devices is given. Emphasis is directed to those devices that are technologically more mature and well-experienced. The second half of the chapter is devoted to the design procedure of a prototype three-phase, full-bridge inverter that used Power Metal Oxide Semiconductor Field Effect Transistors (MOSFET) as the power switching devices. This inverter served as the power conditioning stage of the induction motor drive system under investigation.

5.2 POWER DEVICES

5.2.1 Recent Development in Power Electronics

The development of power electronics in recent years has been so rapid that power electronics design engineers have been presented with the challenging task of choosing the most suitable device for a particular application. Although the power conditioning part of the drive system did not constitute the main theme of the project, a highlight of some of

the important developments in power devices is desirable in order to provide a wider perspective of further improvement of the exsisting system at a later stage.

The present range of power semiconductor devices available for power conversion applications has grown from the original range of diodes and thyristors that were available into a large family including asymmetric thyristors, gate-turnoff thyristor (GTO), power field effect transistors (FET), field controlled thyristor, static induction transistor (SIT) and power bipolar junction transistors (BJT). The resulting decrease in switching times of the devices, particularly with MOSFETs, bipolar power transistors and asymmetric thyristors, has extended the frequency range of power electronics switching devices from the traditional power frequency of 50 Hz up to 1 MHz. In particular the fast switching times of the new devices available have reduced the thermal shock and heat dissipation problems which previously existed in switching devices with lower switching time. Further developments in the field of power switching devices may be expected to attempt to bring about improvements in the following characteristics:

(i) A decrease in the complexity of interface between signal electronics and power electronics

(ii) The lowering of switching transition times, which will result in a reduction of switching losses and an improvement in switching frequency

(iii) A reduction in the forward voltage drop of the device during conduction, which will improve in areas such as current rating, device cost, efficiency and cost of cooling requirements

(iv) A complete elimination or a reduction in the size of the forced commutation components in thyristor converters by the improvement of turn-off characteristics of the device.

5.2.2 Review of Existing Semiconductor Power Devices

5.2.2.1 Bipolar Junction Transistor (BJT)

The BJT is the cheapest form of self-commutating device available and is widely used in the low and medium range of power market. The Darlington and Triple Darlington configurations in which BJT may be used, considerably increase the overall current gain at high power levels, although conduction losses are still high at the present technology. A

further disadvantage is that the BJT requires relatively complex base drive circuits to turn the device off. However, the turn-off process of the BJT does not require additional commutating circuitry, and therefore does not incur the associated switching losses of the commutating circuit. This results in higher switching frequencies when compared with other devices such as thyristors.

5.2.2.2 Thyristor

The thyristor, when compared with the BJT, can have a much higher forward blocking voltage and current rating, which allows megawatts of power to be handled. The thyristor therefore dominates the highest power range market. The thyristor also has good *dv/dt* and *di/dt* performance, and therefore require smaller snubber circuits. Because of the low gate power requirement of the thyristor, the gate driver and the isolation circuit can easily be achieved by means of a low rating pulse transformer. The main disadvantages of the thyristor may be summarized as: (1) the low operating frequency, limited to typically a few kHz resulting from the high switching losses associated with its long turn-off time of tens of microseconds; (2) the extra commutation components required to turn off the device in inverter applications, which considerably increase the complexity and cost of the inverter, and decrease its operation reliability.

5.2.2.3 Gate Turn Off Thyristor (GTO)

The GTO combines the high blocking voltage and forward current capability of the thyristor with the less complex, fast turn-off, high frequency performance of the BJT. Turn-off is achieved by the reversal of the gate-cathode voltage, which results in reversal of the gate current. The reversal of the current results in the interruption of the regenerative switching action, and causes the device to turn off. Although the turn-off circuits required can sometimes be complex and expensive, these advantages are more than offset by the absence of commutating circuitry. The turn-off energy requirement of a GTO, when compared with a thyristor, is very much less. Present-day's GTOs of ratings 2500A-4500V are used in traction drive applications. Recently, MOS-gated thyristors that combine the low on-state losses of a thyristor and the high input impedance and ease of control of an MOS-gate become available [Nandakumar,1991]. Although the GTO has

dominated the market at medium power and at low and medium frequencies, the GTO is expected to be replaced by the MOS-gated controlled thyristor (MCT) in the future.

5.2.2.4 Power MOSFET

The power MOSFET represents the first application of LSI fabrication technology to discrete power devices. It has renowned excellent characteristics such as high switching speeds, low drive requirements, possibility of parallel connection and absence of secondary breakdown. An important consequence of these advantages is that they allow the PWM inverter techniques to be used cost-effectively in low power level applications. This is due to the low cost of interfacing the MOSFET power converter with digital systems such as microcomputers or gate arrays, which provide a versatile means of generating PWM waveforms of different sampling strategies. The main disadvantages of the MOSFET are the relative ease with which the gate can be damaged by static or by voltage transients, and the relative high on-state resistance. The maximum ratings of about 1kV and 20A of MOSFETs mean that they can only be used in low and medium power applications. The MOSFET and its applications will be further enlarged upon later in this chapter.

5.2.2.5 Insulated gate bipolar transistor (IGBT)

Various methods have been developed to combine the advantages of the BJT and MOSFET, of which the IGBT is the most commercially advanced. The IGBT is a voltage-controlled device in which the current flowing through the MOS channel provides the base current of the inherent PNP bipolar transistor. Typically, the IGBT provides much faster switching speed than the BJT and attains significantly lower ON-resistance than the MOSFET. The development of the IGBT widens the spectrum of the voltage-controlled transistors drastically, so that a semiconductor switch with finely adjustable blocking voltage values of 50V to 1600V and the same gate drive circuit is now available. In terms of maximum rating ranges, short circuit strength, robustness and controllability, the IGBT is superior to BJTs and GTOs.

5.2.3. New Devices and Technologies

A number of new semiconductor power switching devices have been developed with the aims of achieving higher switching speed, lower losses and higher reliability over a certain power range. This new range of new devices mainly consists of the static induction transistor (SIT) and the MOS-controlled thyristor in the higher power spectrum, and the bipolor-CMOS-DMOS (BCD) in the lower power spectrum. The highest power rating of these devices is the SITs, which has a forward blocking voltage of 4000V and forward current of 400A. There has also been rapid development in light-activated power devices,

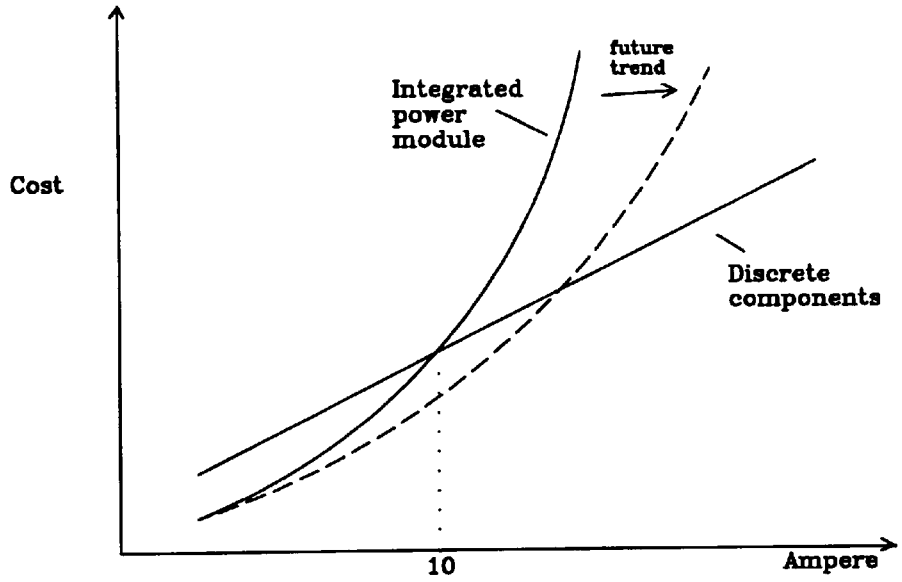


Fig.5.1 Cost versus current ratings for discrete components and power module

the most notable being light-activated thyristors. Development has also been made in packaging of devices commonly referred to as 'power pack' or 'power modules', where power devices, gate drive circuits and sometimes snubber circuits are all parts of one assembly. These new power modules have many advantages to offer. For example, the component counts in a particular circuit is reduced, the assembly time is reduced, and there are less wiring and interconnections which result in shorter and less inductive paths between power devices and hence allow higher frequency of operation. The main disadvantage of the power module is the characteristic of its cost function, as depicted in Fig.5.1. It can be seen that whereas the cost function of discrete component follows an approximately linear relationship with the current rating of the device, the cost function of

the power module follows an 'integrated function curve'. The state-of-the-art technology in power module suggests that 10A is the crossover point, as shown in Fig.5.1, beyond which power module is more difficult and expensive to build than its discrete counterpart. Future developments on power module technology are mainly aimed at advancing the crossover point by decreasing the 'integration constant' of the curve shown, as illustrated by the dotted curved in Fig.5.1. The further improvement of this 'integration constant' is expected to be depended very much on the future development in VLSI and packaging technology.

5.3 DESIGN PROCEDURE FOR A 3-PHASE INVERTER CIRCUIT

5.3.1 Specification

The 3-phase inverter was initially designed to drive a 1 kW 3-phase induction motor. However, it was only possible at the time to obtain a motor rated 2.2 kW, which was fitted with a shaft encoder necessary for implementation of the vector control scheme chosen. Thus, the test rig developed was only capable of running at 50 percent of its full load rating. Nevertheless, the main object of this investigation, which was the application of vector control to induction motor drive, could be readily demonstrated when the motor was run at such derated conditions. The specifications of the inverter may be summarized as follows:

(i) Output frequency range: 1 to 50 Hz

(ii) Output voltage range: 0 to 100 V

(iii) Rated current: 10 A

(iv) Protection: Over Current protection and interlocking circuit to prevent 'shoot through'

The 3-phase full-bridge MOSFET inverter, the schematic of which is shown in Fig.5.2, was designed, constructed and tested to satisfy the given specification. The design procedure is discussed in the following subsections.

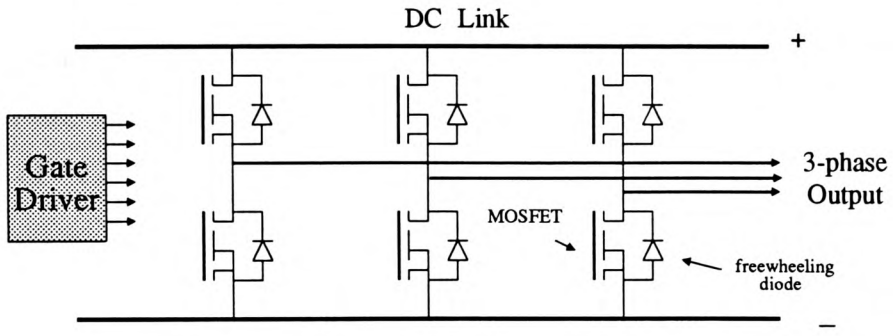


Fig.5.2 A three-phase full-bridge MOSFET inverter

5.3.2 Consideration Given to the Choice of Power Semiconductor Devices

For the power rating of the inverter used, the most popular devices are BJT and MOSFET. For high switching speed operations, the MOSFET has been the preferred choice because it incurs less switching loss than the bipolar transistor. However, for PWM converter drives operating in the 1 to 2 kHz frequency band, the MOSFET must be carefully compared with the bipolar transistor regarding the conduction losses and costs. The conduction losses in a MOSFET can be reduced virtually to any level desired by enlarging the die at the silicon level, or by paralleling devices at the circuit level. This implies that the cost and the conduction losses of the MOSFET device in a converter circuit can be exchanged [Grant,1987]. Thus, when comparing MOSFET and bipolar transistor for low switching frequency applications, it is therefore necessary to consider the total cost of the inverter circuit. An additional cost factor in developing a range of inverters is that a change in the power of a MOSFET inverter requires basically a change in the ratings of the MOSFETs, whereas it is always necessary to redesign the base drive and snubber circuits when a different type of bipolar transistor or GTO thyristor is used.

The simple design requirement of the gate driver and the ease of control of the MOSFET militated against the choice of the power switching devices for the inverter circuit used in this project. The design procedure is described in the following sections.

5.3.3 Theoretical Background of MOSFET

The MOSFET is relatively simple to use and its switching mechanism is well documented in the literature [Siliconix,1984]. It is believed that a simple theoretical treatment suffices the estimation of the switching times and the requirement of the gate current at the design stage. Thus, so far as the switching mechanism is concerned, a simplistic approach may be adopted in order to determine the switching times and gate current requirement.

The definition of the terms and symbols used in this approach is summarized as follows:

V_s = Supply voltage
V_{ds} = Drain-source voltage
V_{gs} = Gate-source voltage
V_g = Gate drive voltage
V_{th} = Gate threshold voltage
C_{dg} = Drain-gate capacitance
C_{gs} = Gate-source capacitance

A simplified circuit diagram used to illustrated the switching action of the MOSFET is shown in Fig.5.3(a). During the switching on process, the capacitances, C_{dg} and C_{gs} charge via R_g and results in the building up of the gate voltage, V_{gs}, according to the following equation:

$$V_{gs} = V_i(1 - e^{-t/\tau}) \qquad (5.1)$$

where $\tau = R_g*(C_{dg} + C_{gs})$, and V_i is the magnitude of V_g.

This initial switching on process corresponds to region 1, as illustrated in Fig.5.3(b). The time, t_d, also shown in Fig.5.3(b), is the time required for V_{gs} to rise to V_{th}, the threshold voltage. It may be seen from equation (5.1) that the time t_d is given by $\tau*\log(\frac{V_i}{V_i - V_{th}})$.

During the switching process when V_{gs} is constant (equals V_{th}) , the current, i_f, shown in

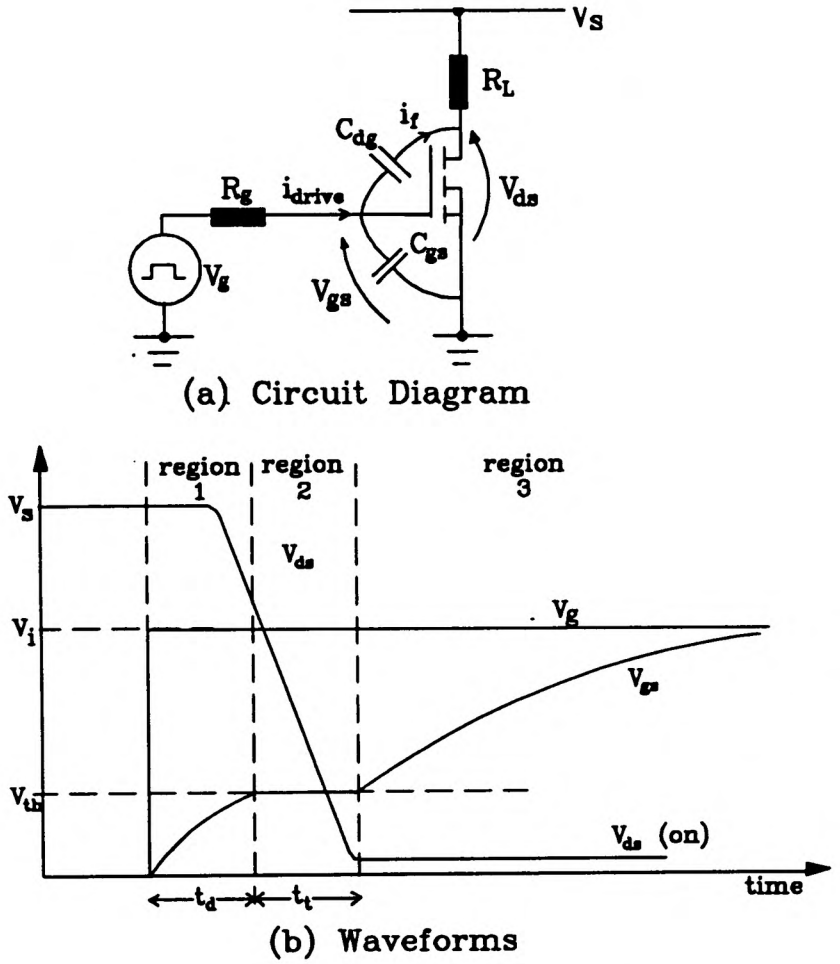


Fig.5.3 Operation of a MOSFET

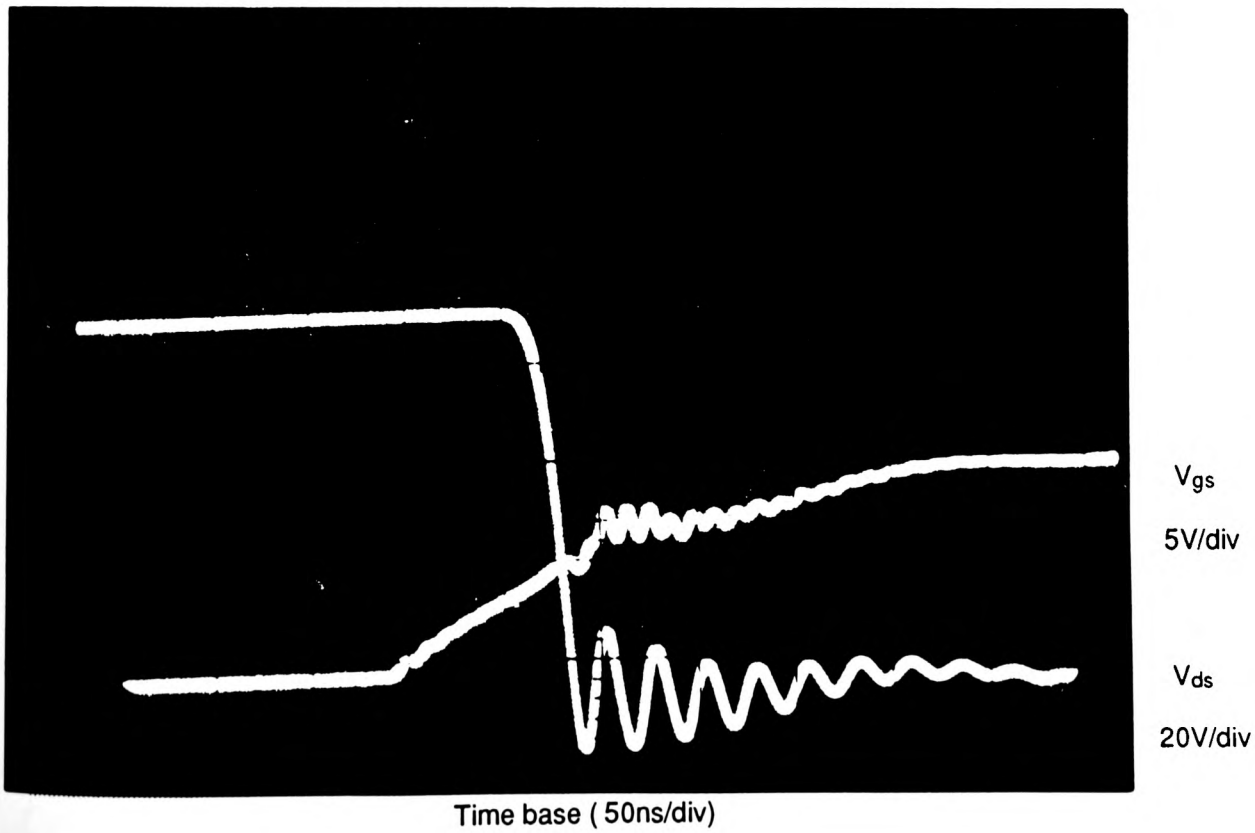


Fig.5.4 Experimental gate voltage

Fig.5.3(b), flows through the capacitance C_{dg} as a result of the changing voltage between the drain and the gate. The value of the current is given by:

$$i_f = C_{dg}\frac{d(V_{ds})}{dt} \tag{5.2}$$

The rate of change of V_{ds} is effectively controlled by the available gate drive current, thus,

$$i_f = i_{drive} = \frac{(V_i - V_{th})}{R_g} = C_{dg} * \frac{d(V_{ds})}{dt} \Big| \text{ during switching}$$

$$\text{or } \frac{d(V_{ds})}{dt} = \frac{V_i - V_{th}}{C_{dg} * R_g}$$

$$\text{or } dt = t_t = \frac{\Delta V_{ds} * C_{dg} * R_g}{V_i - V_{th}}$$

The gate voltage does not drop because if it did the device would turn off, $\frac{d(V_{ds})}{dt}$ would reduce, i_f would reduce and the gate voltage would rise. An equilibrium is thus formed, corresponding to region 2 in Fig.5.3(b). After V_{ds} has reached $V_{ds}=V_{ds(on)}$, gate voltage continues where it left off from equation (5.1). The corresponds to region 3 in Fig.5.3(b). An experimental gate voltage and the corresponding drain-to-source voltage of MOSFET (IRF630 type) is shown in Fig.5.4. The experimental setup was similar to Fig.5.3(a), where the supply voltage used was 60 V and the load was a 10 ohm tubular type wound resistance. The coupling effect of the drain-to-source voltage (V_{ds}) on the gate-to-source voltage (V_{gs}) due to parasitic capacitances can easily be identified from the figure. The frequency, magnitude and damping of the oscillation of the voltage waveforms, however, is dependent on the relative resistive, inductive and capacitive values of the circuit path.

5.3.4 Design Precautions When Using MOSFETs

5.3.4.1 Avoidance of High dv/dt Across Drain and Source

Although high switching speed is generally aimed at by MOSFET users, this can result in, as demonstrated by the experimental result shown in Fig.5.5., the feedback of the load voltage onto the gate because of the parasitic capacitance coupling. In a bridge arrangement, the turning on of one MOSFET in a totem-pole configuration can induce a voltage transient on the gate of the other MOSFET, as illustrated in Fig.5.6. In addition to the intrinsic gate-drain capacitance, C_{dg}, the MOSFET also possesses gate-source and drain-source capacitances, C_{gs} and C_{ds} respectively. If the upper transistor in an inverter arm is turned on, a step function is applied to the lower transistor. If the impedance of the drive stage is high, then the step function voltage is divided down in the ratio of the gate-drain to the gate-source capacitance. The resulting step function appearing at the gate of the lower MOSFET may be of sufficient magnitude to trigger a turn on, leading to the fault condition of 'shoot-through'. Thus, it is necessary to drive the MOSFET from a low impedance drive stage. This allows the shorting of any coupled steps to gate-source common. Low output impedance drivers also ensure a fast switching transition as they are capable of providing large transient currents to charge the gate capacitance C_{gs}. From the voltage and current waveforms illustrated in Fig.5.6, which is related to the circuit of Fig.5.5, it can be seen that the turning ON of the upper device in a totem-pole arrangement causes the lower device to trigger in the active linear region. Therefore, the magnitude of the current during this transient partial 'shoot-through' period is seen largely independent of the source resistance of the supply. Nevertheless, the oscillations of the current waveform are influenced by the *L-R-C* values of the path of the circuit, whereas the damping effect decreases with the resistance value.

Since the coupling effect is a function of the speed at which the upper device is turned ON, of the device, the resulting 'shoot through' problem may be resolved by lowering the turn ON speed of the upper device, $\frac{d V_{gs}}{d t}$. This can be achieved by connecting a capacitor across the gate and the source of the device or a snubber across the drain and source, the result of which, so far as the gate voltage and 'shoot-through' are concerned, is shown in Fig.5.7. It can be seen that as the induced gate voltage V_{gs} is kept below its threshold voltage (typically 3V), no 'shoot-through' current occurs.

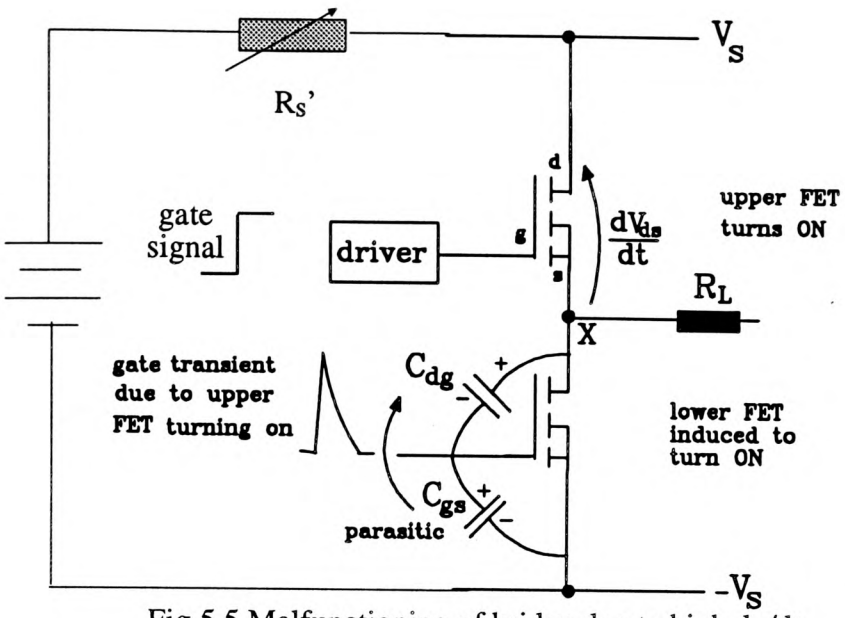
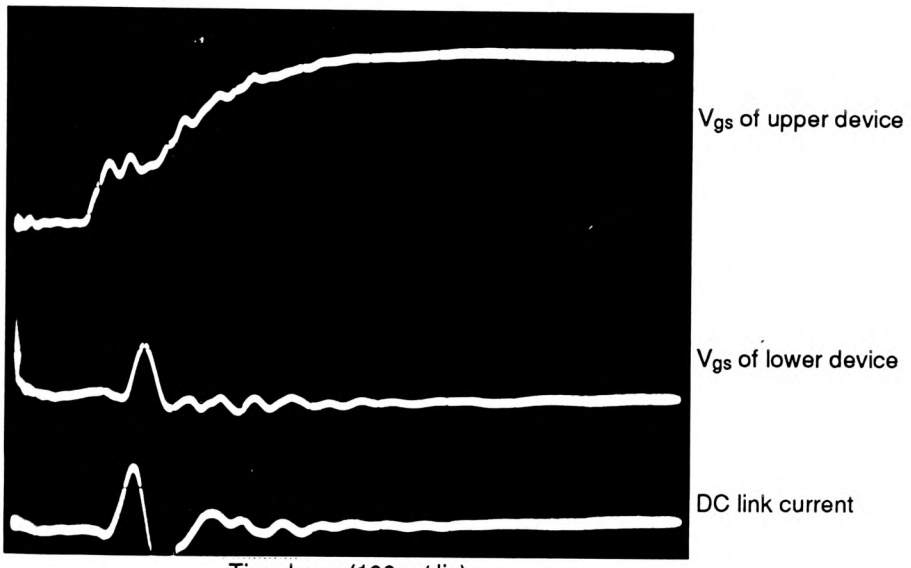
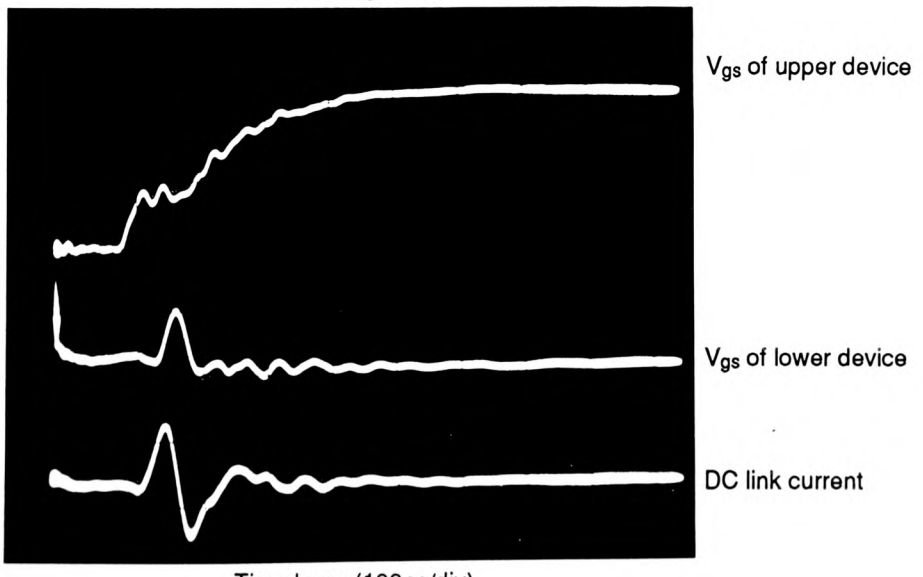


Fig.5.5 Malfunctioning of bridge due to high dv/dt



(a) High R_S'



(b) Low R_S'

Fig.5.6 Transient 'shoot-through' due to high dv/dt

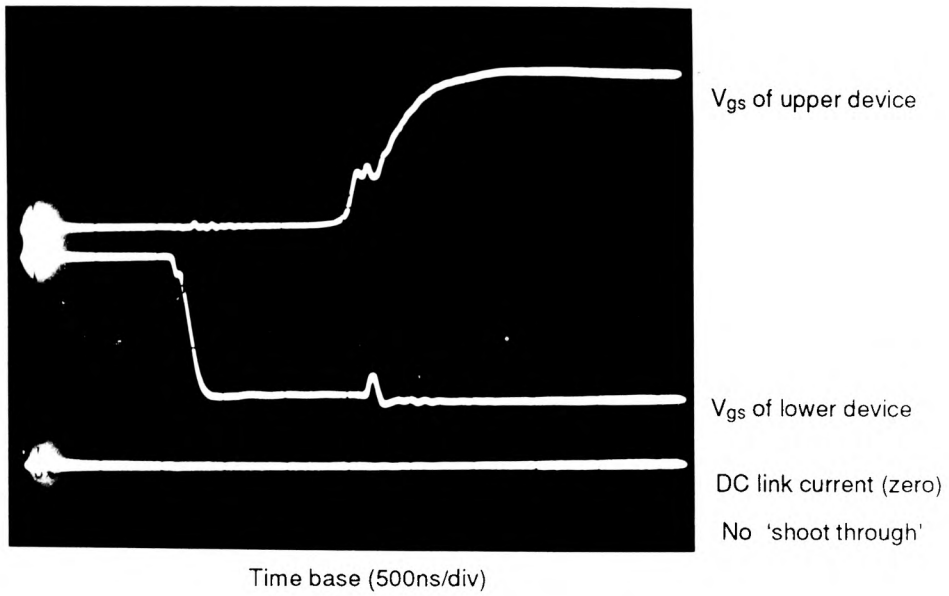


Fig.5.7 Prevention of 'shoot-through' by lowering dv/dt

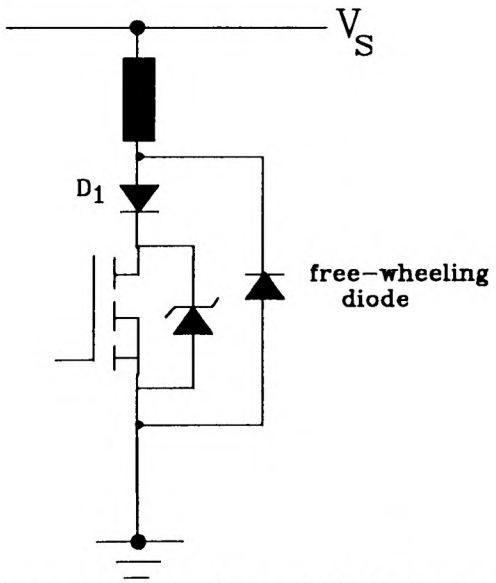


Fig.5.8 Circuit to prevent internal diode from conducting

5.3.4.2 Suppression of the 'integral body-drain diode' of the MOSFET

The 'integral body-drain diode' of the MOSFET's is an intrinsic part of the device itself but is not often required in certain applications. The main disadvantage of this diode is that it exhibits minority carrier reverse recovery, which can sometimes prove problematic. For example, if a MOSFET switch is made to switch on rapidly, the peak reverse recovery current rating can be exceeded and results in the destruction of the device. The dv/dt capability of the MOSFET will also be greatly reduced (a 100 times reduction is possible) as a result of the conduction of the integral body-drain diode. However, in practical circuits, this unwanted diode can be prevented from conducting by connecting a diode in series with the MOSFET and a fast diode connected in parallel, as shown in Fig.5.8.

5.3.4.3 Protection of Gate-to-source, Drain-to-source Voltage From Exceeding Maximum Limit

When an MOSFET inverter is used to supply an inductive load, then it is possible for voltage spike to occur in the inverter circuit. These voltage spike, which may be of very high magnitude, can be coupled to the gate via the parasitic gate-drain capacitance. This undesirable effect can be reduced by careful design of the drive circuit, for example, reducing the output impedance of the gate drive circuit. A further will minimize potential hazards due to capacitive coupling. The gate may be protected by connecting a zener diode, rated at some value less than the oxide breakdown voltage, physically close to the gate source terminals of the power device. Built-up static charge and transient gate over-voltage may cause the gate-to-source voltage to exceed its maximum voltage, resulting in permanent damage. This can also be protected by a zener diode connected across the gate and source for the protection.

5.3.4.4 Circuit Layout

As the rate of change of currents in inverter circuits using MOSFET's may be high, the minimization of stray circuit inductance is very important. Common source inductance acts to slow switching transitions, as it tends to act against the applied gate-source voltage. Unclamped stray drain inductance interacts with the gate-drain capacitance to produce the

Miller effect, which also slows switching transitions. The gate-drain capacitance is a non-linear circuit element as it depends on the size of the depletion region, which depends on the gate bias. Stray circuit inductance may be minimized by properly designed circuit layout. The length of leads should, wherever possible, be equalized. A symmetrical layout is always a simple and useful guide to achieve this. Indeed, a circular arrangement of power devices mounted on a circular heatsink may be contemplated. The power supply for the gate drive circuit should be well decoupled with electrolytic and disc ceramic capacitors. Capacitors are ideally mounted as physically near to the devices to be decoupled as possible. It may be necessary to use twisted pairs from the gate driver to the transistor itself. An addition of a small resistor (10 ohms) in series with the gate lead, however, may be necessary to damp the parasitic oscillations as a result of the twisted pair.

5.3.5 Choice of MOSFETs

A single arm of the three-phase inverter may be constructed using one *n*- and one *p*-channel device. This simplifies the drive circuit interface as the *p*-channel gate signal is referenced to the positive supply rail, and the *n*-channel gate signal is referenced to the negative supply rail. Thus for the three phases, only one floating power supply is needed for the three upper gate driver circuits; and another one for the lower three gate driver circuits. In addition, each device in an inverter arm is driven essentially by the same waveform, the polarity of each device automatically ensuring correct phasing. However, it is very difficult to match the *n*- type and *p*-type devices in a complementary totem pole due to the lower mobility of holes in the *p*-type compared with electrons in the *n*-type. This results in *p*-channel devices exhibiting increased ON- resistance for a given rating. Nevertheless, ON-resistances between *n*- and *p*- channel devices may be matched by paralleling a number of *p*- channel devices. This, however, has the adverse effect of increasing input capacitances, thus degrading high frequency performance. The highest power MOSFETs are only available as *n*-channel types because the chip size and cost of high power *p*-channel devices become prohibitive. In practice, it is possible to achieve a good compromise by selecting complementary *p*- and *n*- type devices with rating up to

about 200 V, 12 A. At present, only *n*-channel type devices are available for higher ratings.

A factor of at least two is usually adopted for choosing the semiconductor device voltage rating with respect to the DC link voltage. For a DC link of 100V, a device of 200V or above will be sufficient. However, since in practice the price of a 500 V device was the same as that of the 250V device, a 500 V device was chosen. Thus, such an over design was entirely justifiable and made no additional demand on the design of the gate drive circuit.

Therefore, taking all of the above factors into consideration, the ratings of the device chosen for the power MOSFET inverter reported in this thesis are given in Table 5.1. The symbols used are those given in the data book [IR,1987]

V_{ds}	$R_{ds}(ON)$	I_d	R_{th}	t_r	t_f
500V	0.4Ω	14A	80K/W	50ns	40ns

Table 5.1 - Manufacturer data of IRFP450 (MOSFET)

Similarly, the manufacturer's data of the free-wheeling diodes, which has a maximum recovery time of 50 ns to ensure protection for the MOSFET during switching OFF, is as given in Table 5.2.

V_{RRM}	t_{rr}	I_{mean}
200V	50 ns (max)	3A

Table 5.2 - Manufacturer data of UF5402 (ultrafast diode)

5.3.6 Gate Drive Circuit

When all of the MOSFET devices used in a 3-phase inverter bridge are of the *n*-channel, the voltage on the source terminal of an upper device in an inverter bridge is connected to the load and therefore changes voltage with the phase output. Thus a separate floating power supply is necessary for each phase of the bridge, so that the low voltage gate signal can be impressed onto the corresponding high voltage side. However, the three lower

drivers of each phase may all have a common ground and a common supply because the source terminals of the three MOSFET's are fixed at the negative DC link voltage. Thus, the apparent advantages of using six *n*-channel MOSFET's, such as a low $R_{ds(ON)}$ value, compared to its *p*- channel counterparts, as seen in their specifications above, are offset by the complication of the necessary power supplies. This was overcome by using inexpensive power supply as will be explained later.

So far the devices used in inverters have only been considered from their relative ability to supply the necessary power to the motor. All of these devices, however, require to be interfaced to the signal logic. The required interfacing circuit may be produced by digital circuitry ranging from discrete devices to signal processors. Due to the difference in power levels between the logic signals and the power waveforms supplied by the inverter, the interfacing circuitry should also provide the necessary isolation and shielding required for interference, not least a reliable and accurate reproduction of the signal input. Several methods were feasible, of which the interfacing by pulse transformer and opto-coupler appeared to be the more common. These two methods of interfacing were considered at the design stage:

(i) Use of pulse transformer - The application of a pulse transformer to a gate drive is illustrated in Fig.5.9. This method has the the advantage of providing DC isolation, a step-up or step-down capability and impedance matching to the driver circuit. However, such circuits can become quite complex and in some applications an anti-saturation circuit is required. Thus, these undesirable features affect the requirements for simplicity and reliability that are normally aimed for when designing MOSFET driver circuits. The literature available [Al-hosini,1985], nevertheless, does suggest simple pulse transformer driver circuits can be designed. However, this novel technique has not been examined in the investigation reported in this thesis.

(ii) Use of solid state opto-coupler - The use of opto-coupler in a MOSFET power circuit is illustrated in Fig.5.10. Recent developments have make available very fast, monolithic, solid state opto-couplers that can overcome the design problems associated with the use of pulse transformer. The driver circuitry becomes relatively simple in its design and the complex pulse amplifying circuitry and anti-saturation components are not required. The

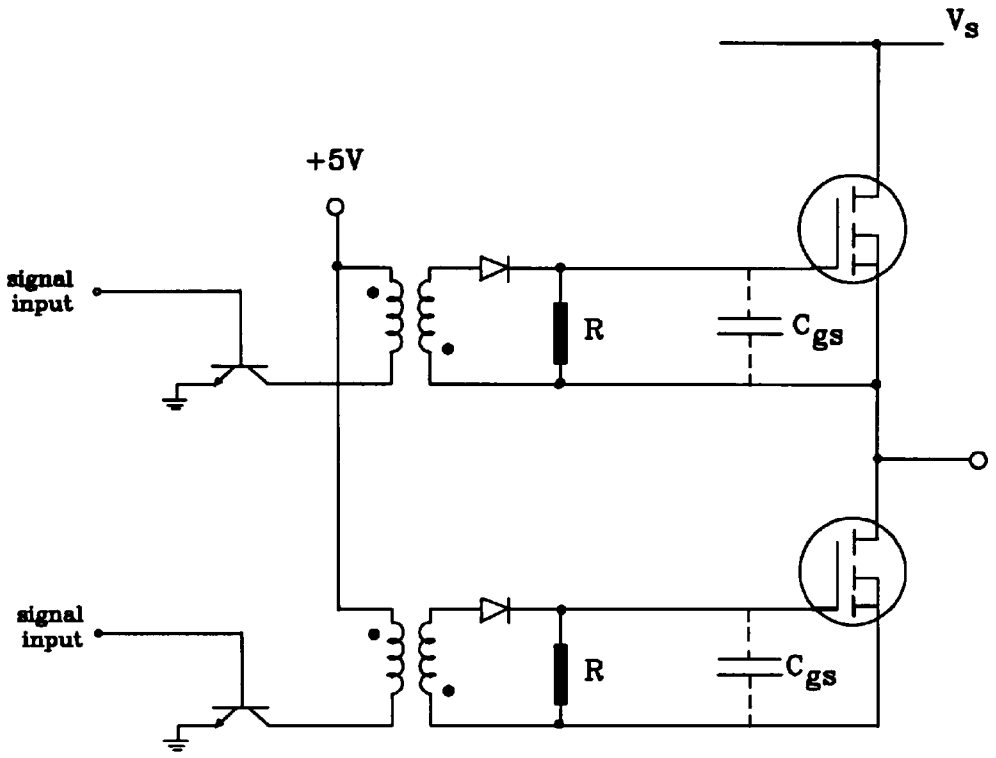


Fig.5.9 Pulse transformer gate driver

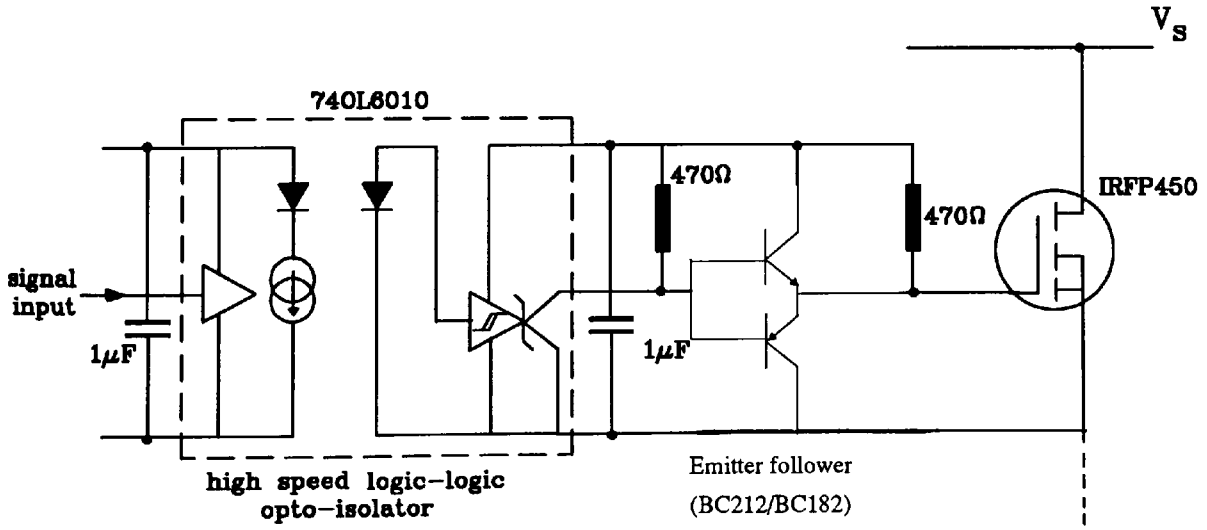


Fig.5.10 Opto-isolater gate driver

solid state opto-coupler driver circuit also dissipates less power, which results in a decrease in size and weight.

On the basis of easy design, component availability and cost, an opto-isolator was chosen for the gate drive circuit. Other considerations at this stage included the amount of logic circuits required, and the time required to build the circuit. The interfacing and isolation stage was achieved by a newly introduced device called the 'High speed logic-logic opto-isolator' [RS,1990]. It is a truly logic compatible optically coupled interface gate. The low power Schottky TTL to CMOS (LSTTL to CMOS) interfacing enables a direct drive from a TTL source to a MOSFET to be realised. The device also offers very high speed, internal noise shielding and noise immunity similar to that of the LSTTL logic.

5.3.7 Floating Power Supply Units (PSU)

On the basis of power consumption and efficiency, the switch-mode power supply (SMPS) appeared to be the ideal option to provide the four separate floating power supplies for the gate drives required. Specialized ICs for SMPS are available on the market that improve reliability and component count. However, these advantages have to be weighed against the cost and power consumption of the gate drive circuit. Since the power consumption required to drive the MOSFET was very low at the frequency range considered, the use of simple and inexpensive linear regulators was justified. Despite the simplicity of the circuit, the final power supply unit (PSU) unit that consists of four separate and identical linear regulators of type 7812, each providing an isolated output voltage of 12V and a reference point, provided a very reliable and stable output. The circuit diagram for the PSUs is given in Appendix-G.

5.3.8 Voltage Protection for Gate-to-source and Drain-to-source Terminals

The gate-source voltage of the MOSFET was protected by a 12V zener diode. The integral body-drain diode of each MOSFET is prevented from conducting by connecting an anti-parallel high speed diode, directly across the source and drain of each MOSFET.

These fast diodes clamp any spike voltage due to switching to the supply rail potential and provide a current path for the usual 'freewheeling' action during commutation of an inverter bridge.

5.3.9 Determination of Heatsink Rating

The calculation below is based for a DC link voltage of 100V and output current of 10A. The power dissipation, *P(total)*, of a switching device may be expressed as:

$$P(total) = P(switching) + P(conduction) + P(gate) + P(leakage) \qquad (5.5)$$

Since power dissipation due to leakage currents and gate currents of MOSFET can be neglected in comparison with conduction and switching losses in normal inverter drive circuits, the calculation was only performed with regard to conduction and switching power losses.

(a) Estimation of Conduction Losses

There are two distinct intervals of operation of the switching devices in the inverter — the conduction and commutation intervals. In the switching strategy employed, each device conducts for about 180^{o} over one cycle, and there are always three devices in conduction at any time. The assumption that the DC link current flows through 3 transistors at any time dictates the worst case for disruption. For an ambient temperature of 50^{o} C, and the MOSFET rated at I_D = 10 A, then the total conduction loss of a device is:

$$P(conduction) = I_D^2 * R_{ds}(ON) = 10^2 * 0.4 = \underline{40\ W}$$

It should be noted that the losses due to 'flywheeling' diodes are small during normal operating condition when power factor is maintained high, and therefore such loasses are neglected.

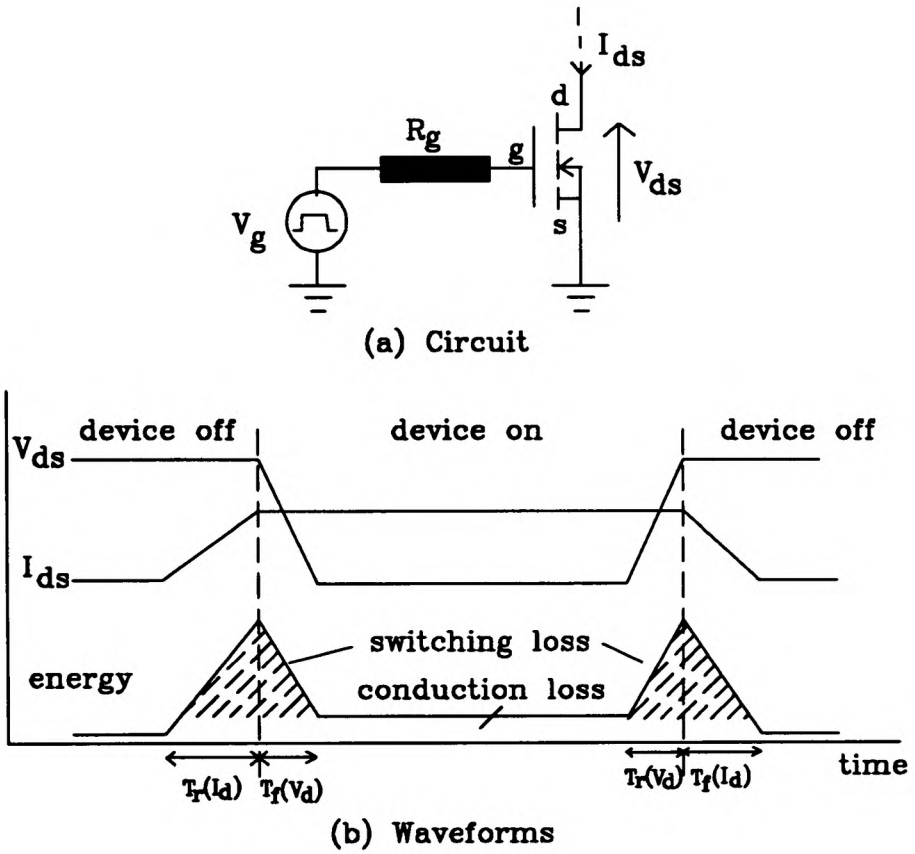


Fig.5.11 Energy loss of a MOSFET during switching

$$T_J - T_A \simeq P*(\Theta_{JC} + \Theta_{CS} + \Theta_{SA})$$

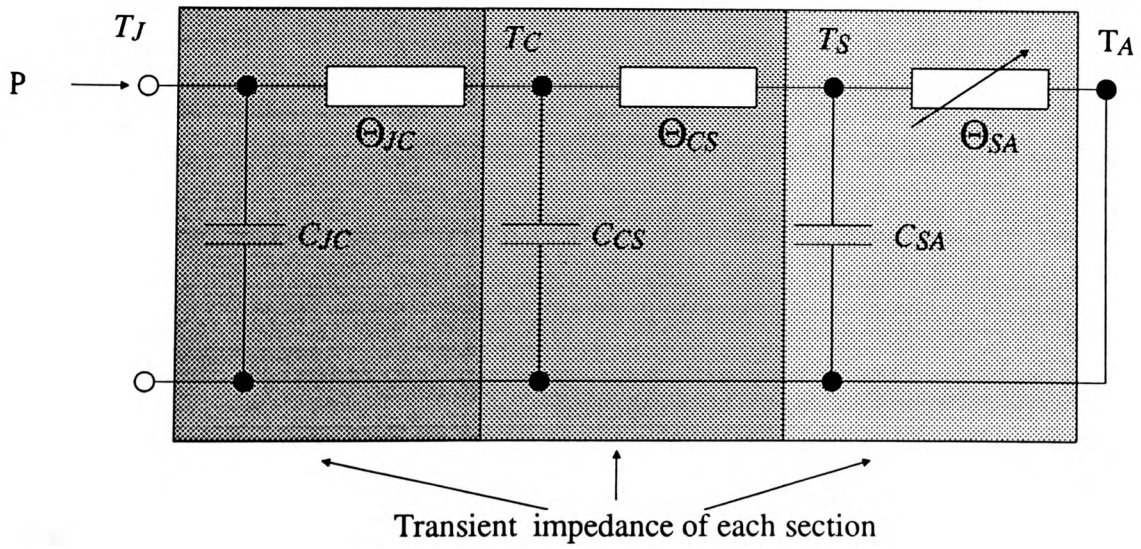


Fig.5.12 Thermal resistance model

(b) Estimation of Switching Losses

It can be seen that the losses can be reduced by minimizing the time required for a draining source transition, thereby reducing the time during which substantial voltages and currents are present simultaneously. Fig.5.11 shows the current and voltage waveforms, and the associated energy losses when a MOSFET is supplying a pure resistive load.

Using a triangular approximation for switching losses, the power lost due to switching is estimated by [Stevens,1985]:

$$P(switching) = [\frac{V_d I_d}{2}] * [T_r(I_d) + T_f(I_d) + T_r(V_d) + T_f(V_d)] * f_c \qquad (5.6)$$

where $T_r(Vd), T_f(Vd)$ = rise time and fall time of drain voltage,

$T_r(Id), T_f(Id)$ = rise time and fall time of drain current, and

f_c = Switching frequency

Thus, with V_d= 100 V, I_d=10A, T_r=1606ns, T_f=42ns, f_c = 4kHz,

$$P(switching) = (100x10/2)x(1606+42+1606+42)x10^{-9} x\ 4\ x10^{3} = \underline{16.5W}$$

c) Rating of Heatsink

In Fig.5.12, the symbols are defined by:

J = *Junction*, C = *Case*, A = *Ambient* and S = *Sink (heatsink)*

The junction temperature, T_J, from Fig.5.12, is given by :

$$T_J - T_A \simeq P*(\Theta_{JC} + \Theta_{CS} + \Theta_{SA})$$

Or,

$$T_J = T_A + \Theta_{JA} * P(total) \qquad (5.7)$$

For an operation condition of $T_{A(max)}$ = 50 °C and $T_{J(max)}$ =150 °C, then from equation (5.5), the total power dissipation in the device is as follows:

$P(total) = P(conduction) + P(switching)$ = 40 + 16.5 = <u>56.5 W</u>

Using equation (5.7), Θ_{JA} = (150-50)/56.5 = 1.77 KW^{-1}

From manufacturer data for IRFP450, Θ_{JA}=0.7 KW^{-1}, Θ_{CS} = 0.24 KW^{-1}. Thus, Θ_{CA} = 1.77 - 0.7- 0.24 = <u>0.83 KW^{-1}</u> per device.

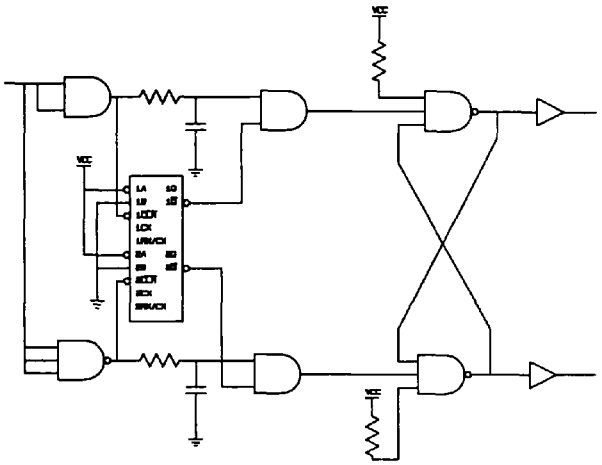
For better dissipation and ease of installation, it is advantageous to mount the six devices on a single heatsink. Since there are three transistors conducting at any time, a heatsink of 0.28KW^{-1} (0.83/3) or lower thermal resistance is required. The calculation above neglects power diode dissipation, and is justified by the fact that the ultrafast speed power diode used carried current for a much shorter time than that of the power MOSFET.

5.3.10 Inverter Interlocking Logic

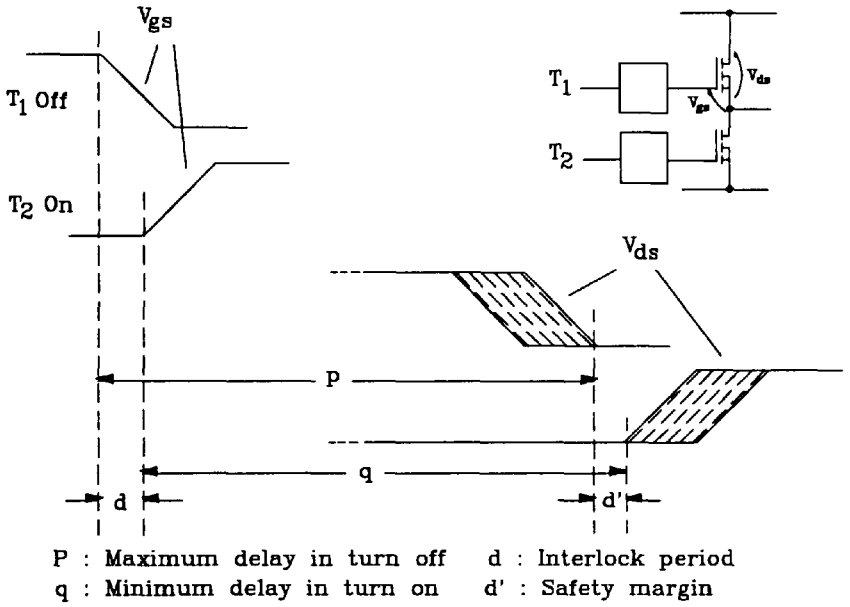
The interlocking logic provides complementary switching signals for the two transistors of each branch with the necessary delay, to prevent 'shoot through' between the two devices. The circuit mainly consists of a monostable and two *CR* circuits as shown in Fig.5.13(a). The delay is controllable by the *CR* time constant of external resistor and capacitance. The condition for preventing the inverter from 'shoot through' is illustrated in Fig.5.13(b).

For the MOSFET transistor, IRFP450, the maximum delay for turning off and minimum delay for turning on are:

$\left|t_{d(off)}\right|_{max} = 100\ ns$, $\left|t_{d(on)}\right|_{min} = 0\ ns$ respectively.



(a) Practical interlocking circuitry



(b) Timing consideration

Fig.5.13 Prevention of 'shoot through' by interlocking

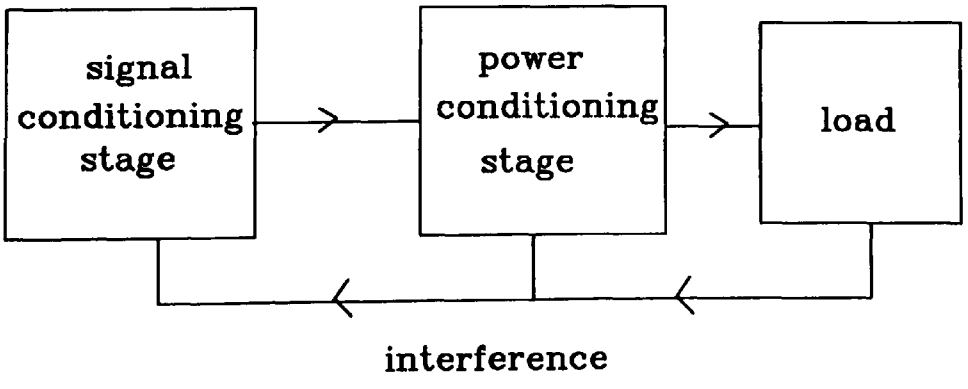


Fig.5.14 Interference in a power conditioning equipment

$|t_{PHL}|_{max} = 120\ ns, \quad |t_{PLH}|_{min} = 0\ ns$ respectively.

The interlocking delay, d, is determined by :

$$d > |t_d(on) + t_{PHL}|_{max} - |t_d(on)\ ,\ t_{PLH}|_{min}$$

Thus,

$$d > |t_d(on)|_{max} + |t_{PHL}|_{max} - |t_d(on)\ ,\ t_{PLH}|_{min}$$

or,

$$d > 100 + 120 - 0 = 220\ (ns)$$

Therefore, the interlocking delay, d, should be greater than 220 *ns*. For a 50% safety factor, then d was made equal to 330 *ns*, the delay pulse-width, t_w, is determined by [Loveday,1982],

$$t_w = 0.32\ RC(1 + \frac{700}{R}) \tag{5.8}$$

and if $R = 33\ K\Omega$ is chosen, it follows from equation (5.8) that:

$$0.33 * 10^{-6} = 0.32 * 33 * 10^3 * C(1 + \frac{700}{33000})$$

which gives, C = 30.6 pF.

The *RC* circuits in Fig.5.13(a) assume these (preferred) values for the resistance and capacitance.

5.3.11 Input and Output Filter Stages

Power devices which operate in unstable, unfiltered supplies, are subjected to overvoltage spikes, over-current and regenerative power. The effect of the interference generated on a

power converter during operation is illustrated in Fig.5.14. The interference from the power stage, if not properly suppressed, will be fed back to the signal stage, which in turn will affect the power stage. This would result in lower efficiency and degraded performance, or premature breakdown and permanent damage to components. Thus, it is particularly important when designing and using power supplies that minimum loss occurred in the supply, cost is kept to a minimum, reliability is maintained as high as possible. Large electrolytic capacitors are one of the means to filter away the ripple voltage in power supplies and thus eliminate the effect of the ripples on the performance and behaviour of the motor. However, the inclusion of such capacitors must be considered against the additional cost incurred. Unwanted high frequency voltage spikes are normally reduced or eliminated by the inclusion of decoupling capacitors connected across the switching devices. It is important, when using such capacitors, that they are placed physically as near to the point where such voltage spikes occur as possible, to minimize stray inductance.

The harmonics in the output PWM voltage waveforms of the inverter used for induction motor control may be reduced or eliminated by the inclusion of *LC* filters. The required cutoff frequency and damping factor of such filters may then be determined accordingly for a given set of motor parameter values. In practice, however, such an inclusion of *LC* filters may only be optimised for a specific output frequency range of the inverter, and is therefore not suitable for applications where a wide range of speed operation is required. On the basis of cost and the fact that leakage inductance of the motor acts as a low pass filter, it was decided not to include filter on the output of the PWM inverter.

5.4 DESIGNED INVERTER CIRCUIT

The complete inverter circuit used in the investigation reported in this thesis, was designed taking full account of the factors described in the early sections of this chapter. A complete circuit diagram of the inverter is included in Appendix-G. The design of the inverter was designed such that it was considerably over-rated for the drive application

investigated. This was done for a number of reasons. (a) it allows a good safety margin, (b) the price of the device used does not vary linearly with the ratings of the device, (c) the choice of a device is often more influenced by the price policy of the supplier rather than on technical bases.

Another important factor that had to be taken into consideration was the effect of the radio frequency interference produced by the inverter on the operation of the transputer. The interference was found to be higher when the motor was subjected to higher load. Therefore, to restrict this interference, a limit was introduced on the load the drive can be subjected to. It is strongly thought that greater consideration must be given to the methods of shielding the inverter and motor from the transputer network. A slide of the 3-phase, full- bridge MOSFET inverter is shown in Fig.5.15.

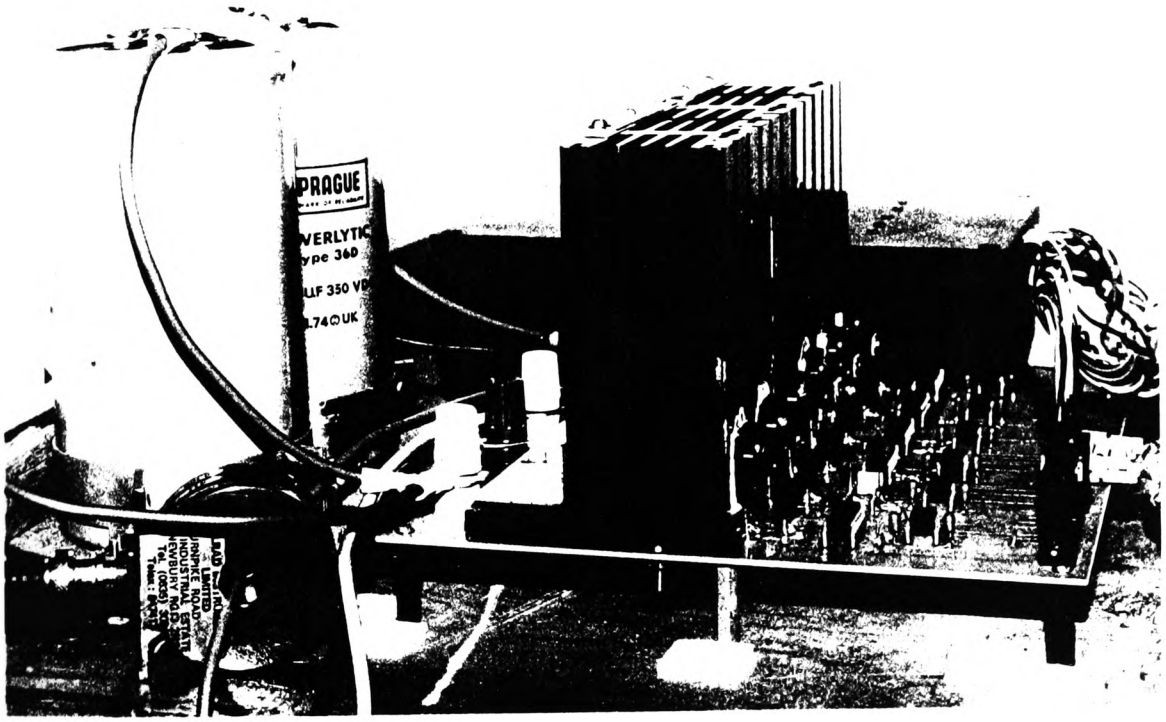


Fig.5.15 A 1kW 3-phase full-bridge MOSFET inverter

Chapter 6 : Implementation of the Three-Transputer Interactive Pulse-Width Modulated Control System

6.1 INTRODUCTION

The 'parallelism' existing within a typical induction motor control drive has not previously attracted the attention of designers of electrical drives, probably because the treatment of such 'parallelism' would have been very difficult to implement with sequential microprocessor systems, and not cost-effective to be realised commercially [Davies,1991]. The introduction of the transputer in 1986, however, has produced a microprocessor system that may be used to implement the necessary 'parallel processing' for the control of PWM induction motor drives. The waveform generations, control algorithms and other house-keeping functions of a complete drive system, which could once only be handled by a very powerful sequential processor, can now be 'divided' and 'distributed' to different processors for implementation, with the promise of greater flexibility and much lower development cost.

In this chapter, the fast processing speed and the parallel processing support facility of the transputer are fully utilized in an 3-transputer network that forms the processing and monitoring (input and display) parts of a PWM drive system for a three-phase induction motor. The processing speed of the transputer allows the PWM waveforms to be generated in real-time. Its dedicated facilities for the support of parallel processing enable the multi-processor network system to be implemented with far less software and hardware design complexity when compared with conventional multi-processor systems. The simplified block diagram of the complete induction motor drive system is depicted in Fig.6.1. The transputer network consists of a host transputer mounted inside the host computer, and four transputers mounted on an external board, of which only 2 were used at the initial stage. The interface includes the PWM generator, the speed measurement circuit and the analog-to-digital converter. The power conditioning unit is the three-phase MOSFET full-bridge inverter, which has been described in the previous chapter. The speed measurement part and the A-to-D converter will be discussed in the next chapter.

This chapter describes the development of a user-interactive drive system for a 3-phase induction motor using a transputer network, under the transputer development system

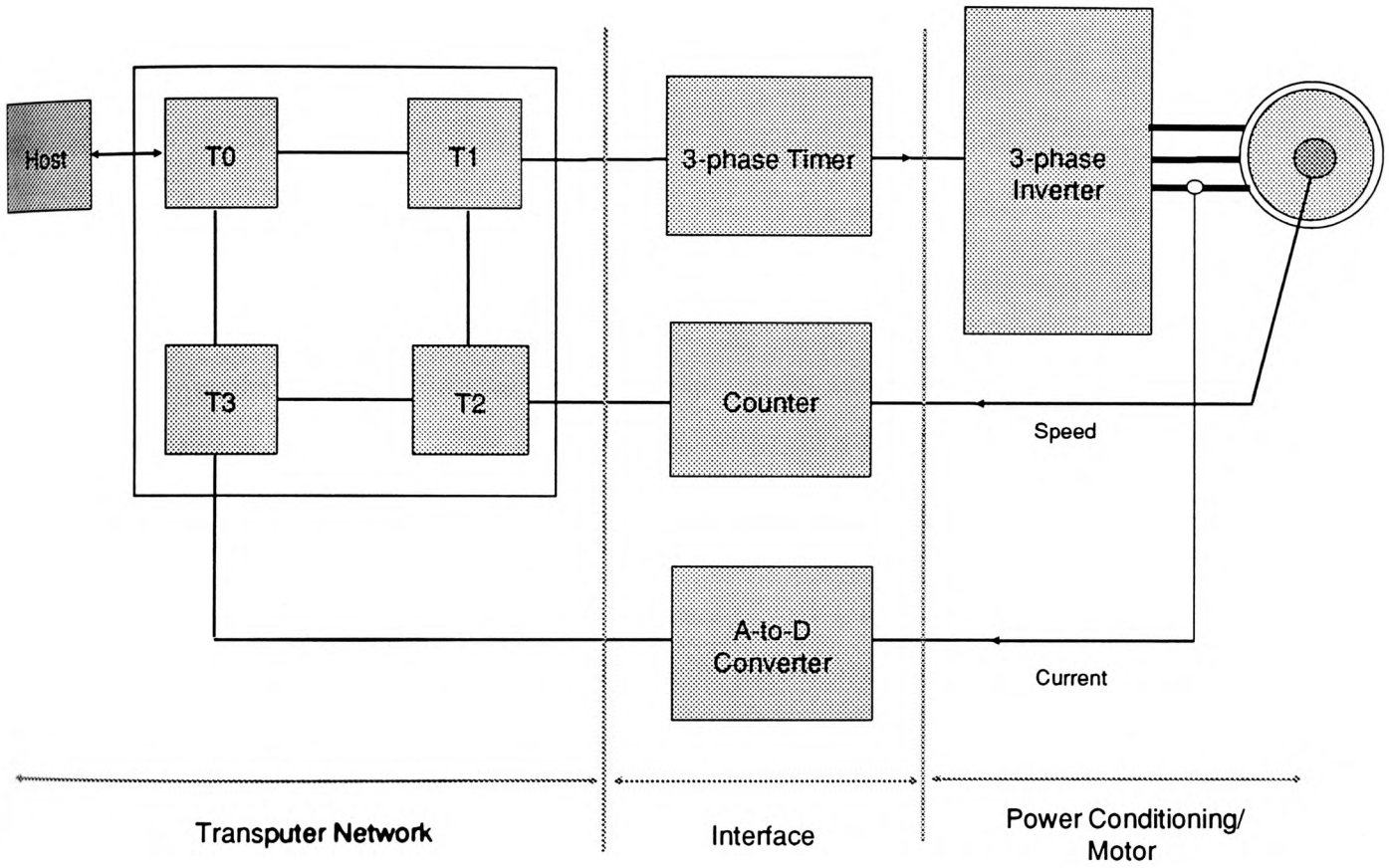


Fig.6.1 Block diagram of drive system

(TDS) environment. The transputer and its 'natural' language occam will first be introduced to an extent that necessary information and clarity are achieved. After the specifications of the drive system have been discussed, the development of the PWM waveform generating part, which constitutes the main part of this chapter, will be described in detail. This includes the discussion of the hardware PWM generator and the software program written in occam. Critical parts of the program are described in detail, and the full program listing is given in Appendix-C.

6.2 THE TRANSPUTER AND occam

6.2.1 Overview

A comprehensive treatment of the transputer and its associated programming language occam is beyond the scope of this thesis. Whereas a wealth of literature is available elsewhere [Inmos,1988], an abstract of the transputer architecture, the basics of the occam programming language, and other technical aspects such as the 'configuration' procedures of transputers, which are relevant to this thesis, are found in Appendix-A. Therefore, this section of the thesis (6.2) is mainly intended to give a non-technical overview of the transputer and its language occam.

6.2.2 Basic Transputer Philosophy

The emergence of the transputer can be seen as a combined result of the rapid development of the Very Large Scale Integrated (VLSI) devices and the increasing demand for processors that support parallel processing. The name 'Transputer', derived from the two words 'TRANSistor' and 'comPUTER', underlines its original design intention. As transistors have been the building blocks of computers for many years, so transputers will become the building blocks of a new range of machines from compact array processors to vast supercomputers. Developed along side with the transputer has been the language occam, which is basically a simple programming language. occam is based explicitly on a model of concurrency and communication well matched to the

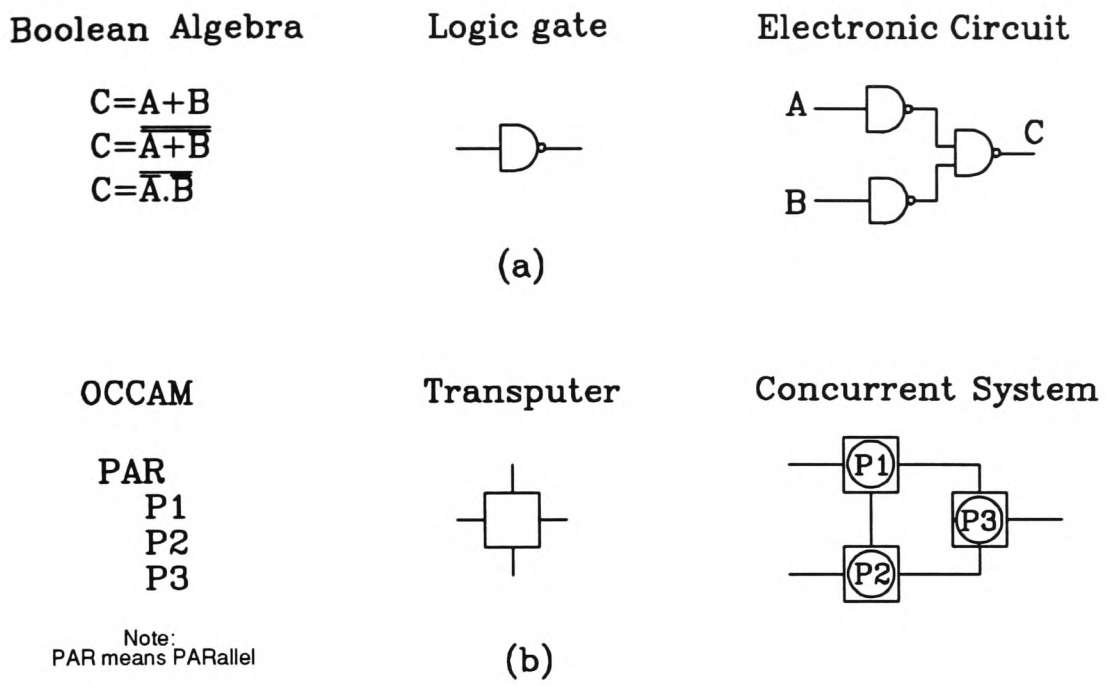


Fig.6.2 Comparison of relationship of (a)Logic gate/Boolean Algebra with (b) The transputer / **occam**

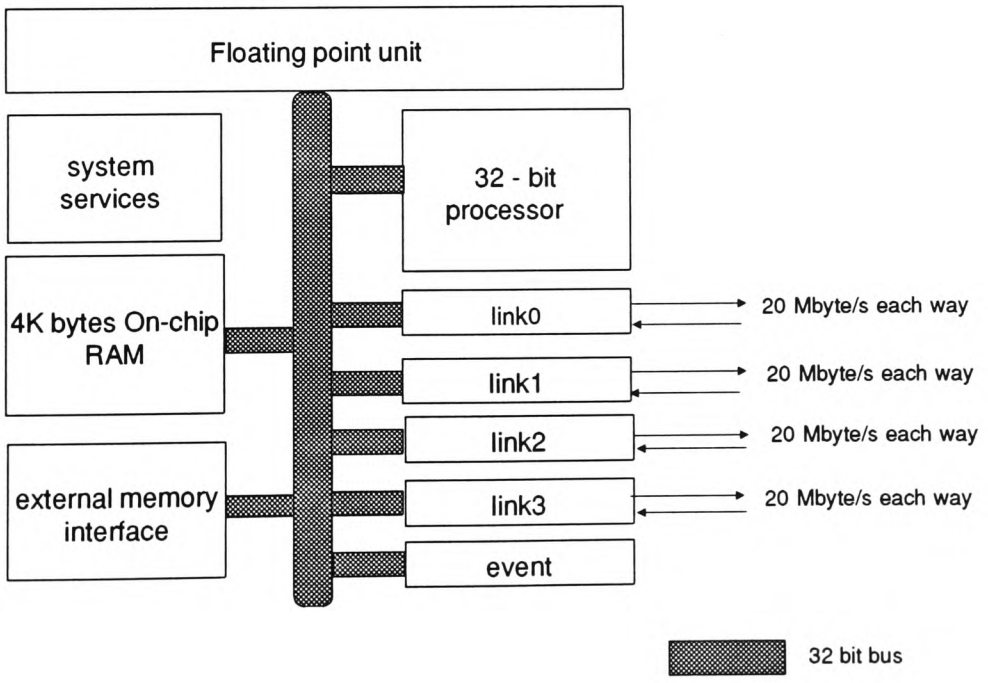


Fig.6.3 Architecture of the inmos IMS T800 transputer

transputer. The close relationship of occam with the transputer is analogous to that of Boolean algebra with logic gates, as illustrated in Fig.6.2.

6.2.3 The Transputer

The transputer is a generic term for describing a family of programmable VLSI devices including — disk controllers, floating point processors, graphics processors, signal processing devices and 32-bit and 16-bit general purpose processors. The first and second generation of transputers basically comprise a processor, a memory unit, communication circuitries and a floating point unit all fabricated on a single chip. The architecture of the T800 transputer is shown in Fig.6.3. Three salient features make the transputer an ideal component for the development of parallel processing in an electrical drive system. Firstly, the transputers can inter-communicate and function cooperatively with each other using its hardware links. This allows transputers to operate concurrently without the traditional management software overhead that in sequential system limits the amount of performance improvement attainable by using further sequential conventional processors. Secondly, the transputer is a powerful processor in its own right. Thirdly, the transputer can be programmed in high level languages with virtually no loss of efficiency.

6.2.4 The occam Programming Language

The name occam was chosen in recognition of a 14^{th} century philosopher, William of occam, who was responsible for the adage, known as occam's razor, that: ''Entities are not to be multiplied beyond necessity''. The occam language is relatively less complex than other concurrent programming languages such as *Ada.* In fact, the early version of occam (occam1) was much less complex, and in a sense, adhered more to the original design philosophy of ''keeping things simple'', than the later versions of occam (occam2), which are claimed by its developer to have wider data support facility.

occam is the lowest conceptual level at which it is necessary to think and design for real-time use, everything else below this is taken care of by the compiler and transputer hardware automatically. It provides a framework for designing concurrent systems using transputers in the same way that Boolean algebra provides a framework for designing electronic systems using logic gates, as already illustrated in Fig.6.2. The system designer's task is made easy because of the architectural relationship between **occam** and the transputer. A program running on a transputer is formally equivalent to an **occam** process, so that a network of transputers can be described directly as an **occam** program.

The software building block of **occam** is the 'process'. A system is designed in terms of an interconnected set of processes. Each process can be regarded as an independent unit of design. It communicates with other processes by channels. The system design is therefore hierarchically structured. Thus, at any level of design, the designer is concerned only with a small and manageable set of processes. The processes can be used to build up larger and more complex processes called 'construct'.

6.2.5 occam Model

The design philosophies applied to the transputer and its language **occam** have been closely related to each other during the development stage. Every transputer implements the **occam** concepts of concurrency and communication. As a result, **occam** can be used to program an individual transputer or to program a network of transputers. When **occam** is used to program an individual transputer, the transputer shares its time between the concurrent processes and channel communication is implemented by moving data within the memory. When **occam** is used to program a network of transputers, each transputer executes the process allocated to it. Communication between **occam** processes on different transputers is implemented directly by transputer links. Thus the same **occam** program can be implemented on a variety of transputer configurations, for example, with one configuration optimized for cost, another for performance, or another for an appropriate balance of cost and performance. All transputers include special instructions and hardware to provide maximum performance and optimal implementations

of the occam model of concurrency and communication. All transputer instruction sets are designed to enable simple, direct and efficient compilation of occam. Programming of input/output, interrupts and timing is standard on all transputers and conforms to the occam model.

6.2.6 Interim Conclusion

The combination of occam and the transputer presents a new concept in the implementation of parallel processing. The close association between occam and the transputer hardware provides programmers with a design language for high-level descriptions of the behaviour of parallel systems. As far as the induction motor control area is concerned, as will be discussed in later chapters, it is important to understand the degree of parallelism within an induction motor drive in order to benefit from this new and challenging technology. 'Parallelism' within an induction motor drive has only been recently considered [Asher,1990]. Only a very limited understanding and knowledge of the effect of parallel processing on the control of induction motors has been reported so far (January 91). With the introduction of the transputer and its ever decreasing market cost, the 'search' (rather than research) for the application of the transputer to PWM induction motor drive systems will not only result in superior drive performance, but also at an acceptable cost.

6.3 SPECIFICATIONS OF A USER INTERACTIVE PWM DRIVE SYSTEM

The most important function in any PWM control scheme is undoubtedly the switching strategy used to generate the edges of the PWM pulses. In addition to the requirements for an efficient means of generating the edges of the PWM pulses, it is also necessary in a variable speed drive system to incorporate certain features that make the drive system practically viable. For example, the simulation results reported in chapter 3 show that the 'direct-on-line' startup of an induction motor rated at about 2.2 kW or above could cause high current inrush and pulsating electromagnetic torque resulting in mechanical noise

and stress, and voltage dip in the supply system. A soft-start method is therefore needed to control the magnitude of the current during the acceleration stage in the starting up of the motor. Since most variable speed drives are required to generate constant torque over a wide frequency range below rated speed, the flux in the machine in such operations must be kept constant by maintaining the induced EMF in the machine proportional to the applied frequency. Because of the voltage drop across the stator impedance, the terminal voltage is higher than the induced EMF. At high voltage and high frequency operations, the effect of the voltage drop across the stator impedance is negligible. At low voltage and low frequency operations, however, voltage boosting is required to compensate for the increased voltage drop across the stator. Other parameters and issues such as frequency ratio or pulse number control, PWM to quasi-square transition, and minimum pulse-width control are also of considerable importance to the performance of a standard PWM drive system.

Whilst most of the above mentioned features have been thoroughly researched, the drive system proposed in this thesis was designed to achieve a front-end flexibility by providing an on-line user interactive environment. The values of the PWM parameters such as the modulation index and pulse number can be changed by the system user when the motor is running. The advantages of the proposed system offer the possibility of optimizing the PWM strategy of a particular application. The more important features of the proposed system may be summarized as follows:

(a) On-line user inputs of PWM parameters and monitoring system to display motor speed, modulating frequency, carrier frequency and other relevant information.

(b) Generation of the PWM waveform in real-time to provide an output range frequency of 1 Hz to 50 Hz.

(c) Optional soft-start procedure

(d) Forward and reverse mode of operation

(e) Provision of dynamic braking

The implementations of specifications (a) and (b) are discussed in this chapter, whereas those of (c) to (e) will be described in the next chapter, which also includes the incorporation of vector control into the system.

6.4 DEVELOPMENT OF ON-LINE USER INPUT AND MONITORING SYSTEM UNDER THE TRANSPUTER DEVELOPMENT SYSTEM

The transputer development system, or TDS, (see Appendix-A for detail) provides an excellent platform for the development of an on-line iterative monitoring system. The TDS uses a personal computer (PC) as a host system. The host provides the environmental support to the transputer module by providing mechanical support for the board, power supply, power on reset, file servers, keyboard and monitor. The host-server model is depicted in Fig.6.4. The '**server**' program, which runs under the disk operating system (DOS), supports protocol down to the links, and provides access to the screen, keyboard, and the filing system of the host computer.

The PWM parameters, which values are available for the user to change on-line, are tabulated as follows:

Parameters	resolution /step	keys for execution (increase,decrease)
Modulating frequency	1Hz	↑ , ↓
Frequency ratio	1	→, ←
Modulation index	0.05	i , m

Table 6.1 On-line parameters

It should be noted that the resolution of the parameters can also be changed by the user.

6.4.1 Terminal Interfaces

The channels named **keyboard** and **screen**, can be accessed by a program of the type executable procedure (**EXE**). A description of the **EXE** program is given in Appendix-A. These channels allow the communication between the terminal and keyboard of the host computer and the TDS system. The channels are not connected directly to the devices of the host terminal, but to a terminal driver process called **term.p**. Various commands may be sent to the driver process, **term.p**, that implements a virtual screen and keyboard interface. The process **term.p** consists of two processes running in parallel, one driving the screen and the other the keyboard.

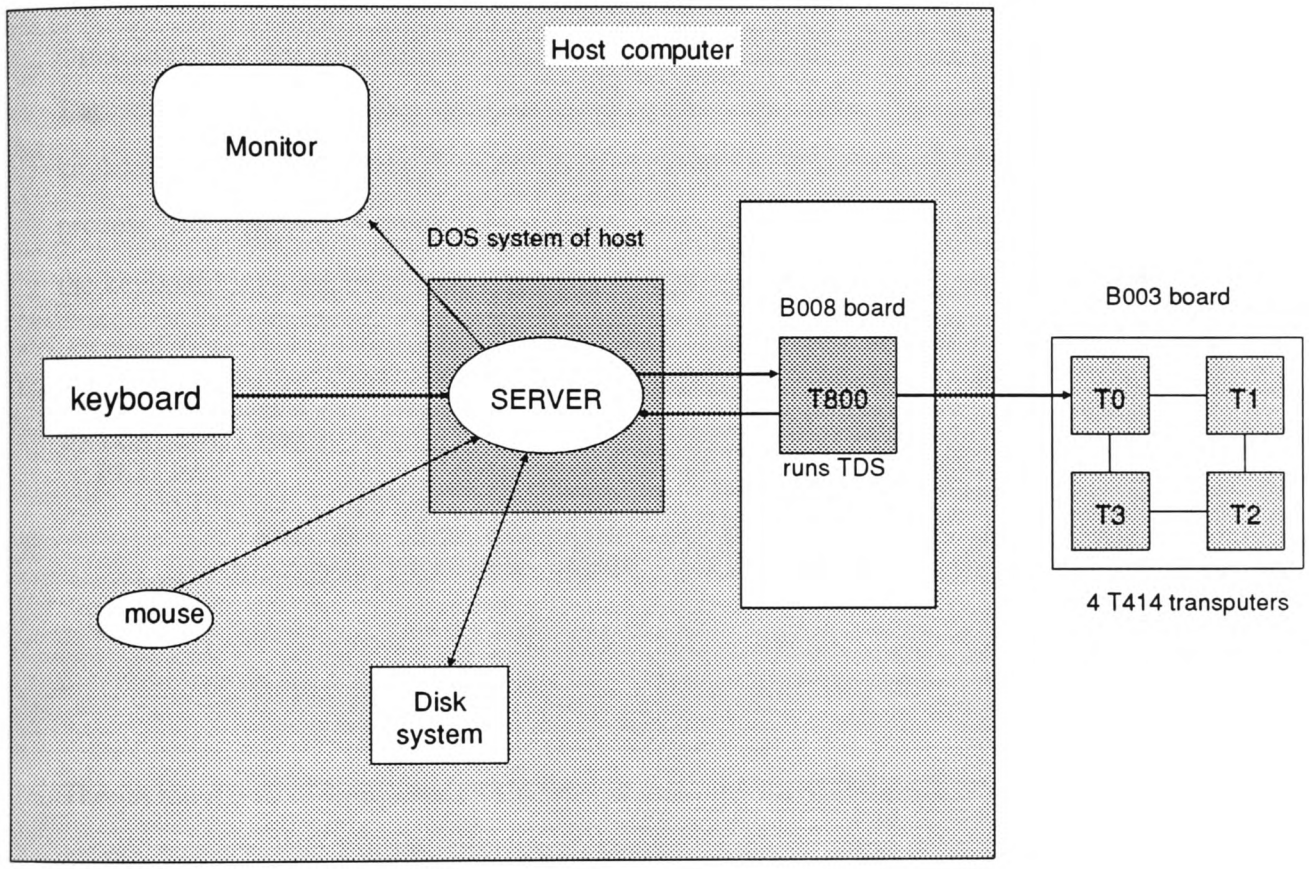


Fig.6.4 Host-Server model of the transputer development system

The main, or top level **occam** program statements that are used to provide an interactive mode for system user to input data from the keyboard, are given in Program 6.1 below. The Boolean variable **cycling**, which can assume either the 'TRUE' or 'FALSE' Boolean constant, is first set to **TRUE**. The **WHILE** loop will be activated until **cycling** is set to **FALSE**. The **PRI ALT** (**PRI**ority **ALT**ernate) command serves the **keyboard** channel first or at a higher priority to **another process input**. This means that any input from the **keyboard** will be served first in favour of the **another process input** when both are demanding attention at the same time. This is necessary because keyboard input is relatively slow and asynchronous as compared to input from some other process, for example, a speed measurement process that inputs at a rate of 5 ms. Setting the keyboard input channel to low priority may prevent it from being served.

```
cycling:=TRUE
WHILE cycling
  PRI ALT
    keyboard ? code
      ... execute the command for the selected code
    ...  another process input
```

The symbol '...' indicates a fold, which contains more program statements within. A general remark that follows the '...' symbol, such as those shown above, serves to improve the readability of the program. This unique feature of the folding editor within the TDS facilitates a top-down design approach and structured programming practice. The folding editor is described in more detail in Appendix-A.

The following **occam** commands concerning with outputting to the screen were extensively used in formating the operation menu of the drive system and displaying the information concerned [INMOS,1988]:

(a) screen ! tt.goto; INT x; INT y

This command takes the cursor to the coordinate (x,y) of the screen, where (0,0) corresponds to the top left corner and (24,80) corresponds to the bottom right corner.

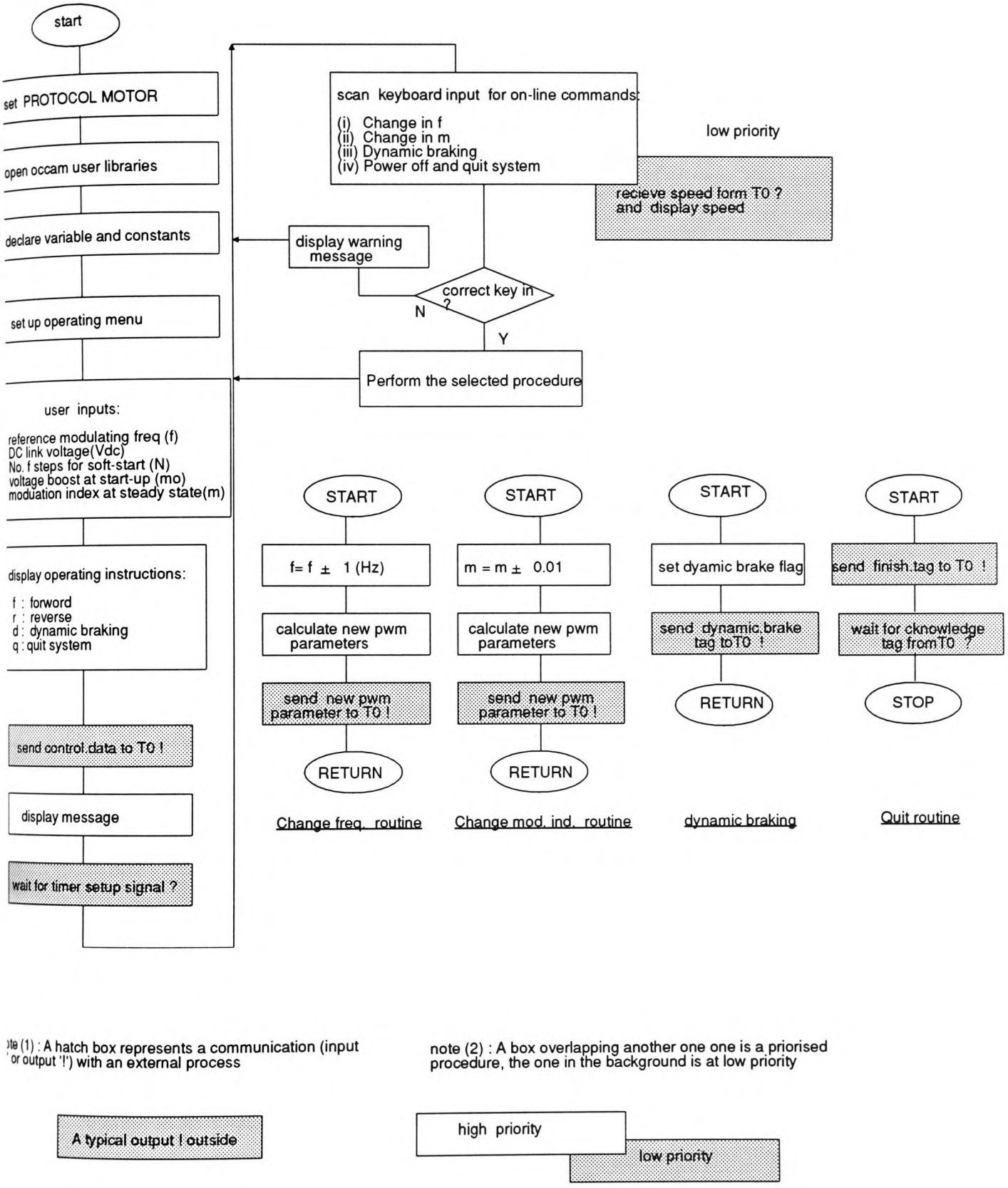


Fig 6.5 Flowchart of interactive PWM control drive system

(b) write.full.string('a certain message')

This command writes the message in quotation to the screen at the current cursor position, and is therefore usually used with the previous command.

(c) screen ! tt.beep

This command gives a beep sound, and is used to provide audio warnings and messages for situations such as an input of a too high reference speed, completion of a soft-start or a dynamic braking process.

The flowchart of the overall interactive system is shown in Fig.6.5, in which some features such as the **PROTOCOL MOTOR**, the speed measurement and the dynamic braking will be explained at a later stage. The system also assumes the processes have been mapped into the processors. The process of mapping, called 'configuration', was considered to be more conveniently to be described under the heading of implementation (section 6.7). At this stage, it is sufficient to know that the shaded box in the flowchart represents a communication, where a '!' denotes an output and a '?' an input, with the external processor **T0**. The processor **T0** is the 'root transputer' in the network which is connected to the host transputer, on which the TDS runs and **occam** programs are developed.

6.5 DEVELOPMENT OF PWM GENERATOR

6.5.1 Development of the model of a PWM generator under the TDS

The logical development of an **occam** process can be done within a single transputer environment such as the TDS system. The process may contain several sub-processes running in parallel, or is just a single conventional sequential procedure. It is the former case of sub-processes running in parallel that is of particular interest to a multi-transputer application. The **occam** process, after being tested and compiled, is then mapped (or configured) to the physical processor(s) for implementation.

The development procedure adopted a top-down structure approach. The **occam** model of a PWM generator, together with the monitoring process developed in the previous section, may be illustrated as in Fig.6.6. The '**monitor**' process inputs PWM parameters from the **keyboard** channel, and supplies the values of the PWM parameters to the '**PWM**' process via an internal **occam** channel. The '**monitor**' process also outputs relevant information for display via the **screen** channel. The '**PWM**' process basically performs the computation of pulse-width from the PWM parameters supplied, and sends the values to the external timer for real-time generation of PWM waveforms.

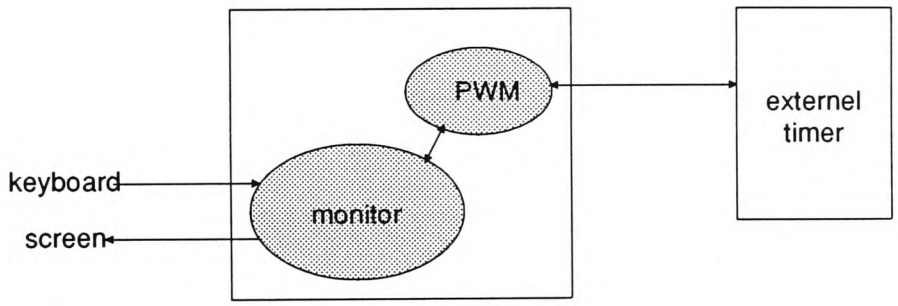


Fig.6.6 An occam model of the PWM drive system

6.5.2 Generation of The Three-phase PWM Waveform

6.5.2.1 The '4-timer Method' of Generation of a 3-Phase PWM Waveform

The '4-timer method' has been used for the generation of a 3- phase PWM waveform by a microprocessor-based hardware system [Bowes,1981]. A possible configuration for the implementation of the '4-timer method' is shown in Fig.6.7. In the figure, in addition to the three timers corresponding to the three phases, an extra timer is required to generate an interrupt signal to the microprocessor, enabling the reloading of the three timers at a frequency corresponding to that of the carrier waveform. This configuration makes use of the common characteristic of the regular sampling processes, (indeed other sampling strategies such as natural sampling and the suboptimum strategies) which is that they all possess a single switching instant per half carrier cycle, if no over-modulation occurs. The process of PWM waveform generation using the '4-timer method' is illustrated in Fig.6.8. From this figure, it may be seen that instead of calculating the total width, Δ_a, of a pulse

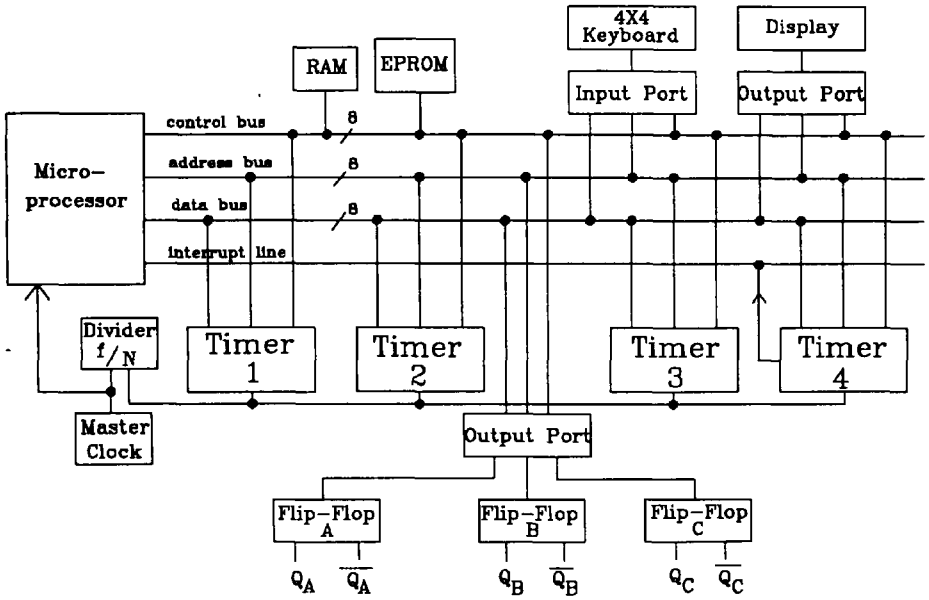


Fig.6.7 A microprocessor system for '4-timer' method implementaion

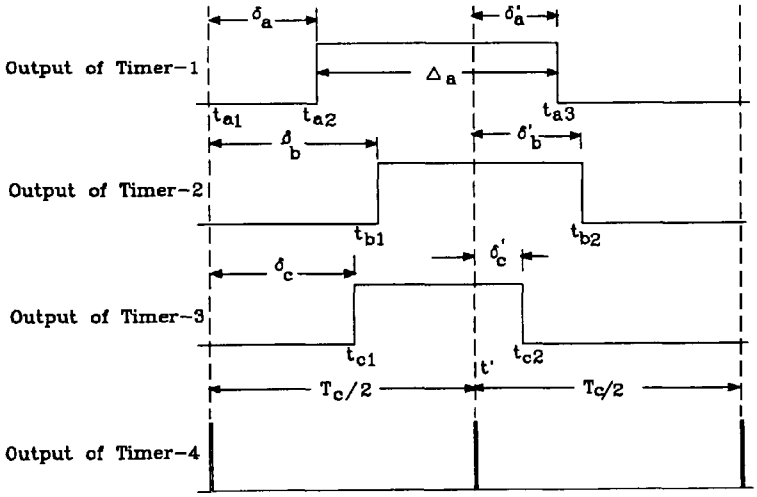


Fig.6.8 Timing diagram of '4-timer' method

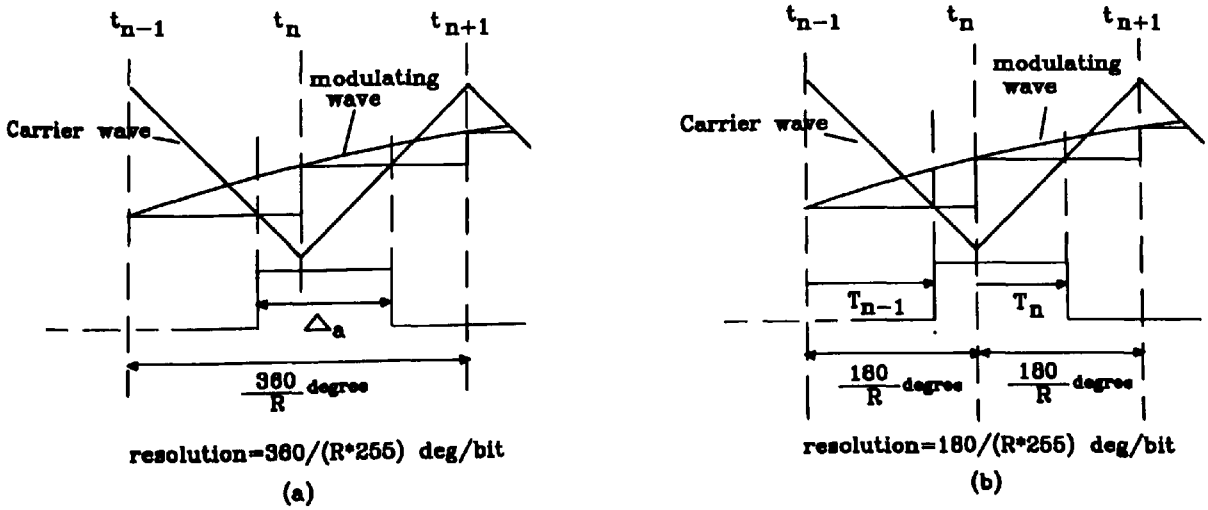


Fig.6.9 Comparison of resolution of pulse-widths
(a) conventional system (a) '4-timer' method

for phase A say, only the distance from a sampling point (apex of the triangular waveform) to the switching instants, δa, is required. For three-phase PWM, the distances δ_a, δ_b and δ_c are thus required. At the sampling instant, t_{a1}, the microprocessor is interrupted by the signal from the fourth timer, it then loads the timers of phase A, B and C with the values corresponding to δ_a, δ_b and δ_c respectively. In addition, the microprocessor loads the fourth timer with a value corresponding to $T_c/2$, where the fourth timer is configured to generate a small pulse at an interval given by the value stored in the timer. When the timers corresponding to phases A, B and C stop counting, their corresponding outputs go 'high' (logic '1'), thus changing the PWM pattern without interrupting the microprocessor. Following these operations, the fourth timer finishes counting at time t', and then generates an interrupt signal to the microprocessor such that the microprocessor loads δ_a', δ_b', δ_c' and $T_c/2$ in the timers corresponding to phase A, phase B, phase C and the fourth timer respectively. The timer of each phase is also instructed to change its mode of timing out, so that its output goes 'low' (logic '0') when counting is terminated for the next operation. Thus, an additional instruction to change the mode of operation for the three timers is required at every interrupting point.

The main advantage of this method is that it increases the accuracy of the pulse-width. For example, in Fig.6.9, it is shown that the resolution of the pulse-widths is doubled when the '4-timer method' is used. For example, if 255 is the maximum number that could be stored in a memory location with an 8-bit system, then Δ_a will be stored with $(\frac{360}{R*255})$ degrees/bit resolution, whilst δ_a will be stored with $(\frac{180}{R*255})$ degrees/bit resolution, where R is the frequency ratio.

6.5.2.2 A Novel Method For the Generation of PWM Waveform by the Transputer

In the previous section, it is found that the generation of the pulses corresponding to δ_a, δ_b and δ_c in Fig.6.8 requires a 3-phase timer that outputs 'low' during its 'count-out' process, and 'high' when counting terminates. Also, the generation of the pulses corresponding to δ_a', δ_b' and δ_c' requires the timer to output 'high' during its 'count-out' process, and 'low' when counting terminates. These conditions require a timer that can be programmed to change its counting mode as described in the previous section.

An alternative method, which is novel and found simpler to implement than the '4-timer method' described in reference [Bowes,1981], is to relegate the process of changing the counting mode to a logic gate by means of a simple software program. The novel method is illustrated in Fig.6.10, it may be seen that the triangular waveform (i) represents the fictitious carrier wave of a period of T_C. The waveform (ii) depicted is the regular sampled asymmetric PWM waveform required. The value of the pulse-width, T_n, corresponding to the n^{th} apex of the carrier wave is computed and scaled in real time by the transputer from the following equation (6.1) [Williams,1982].

$$T_n = (\frac{T_C}{4})*[1 + (-1)^n * M*(sin\ A_n)] \qquad (6.1)$$

where $A_n=p*n/R$, M is the modulation index and R is the frequency ratio of the carrier wave to modulating wave.

After T_n is determined, it is then loaded in the timer at the instant t_n, by means of the timer reloading signal (iii), generated by the internal hardware timer of the transputer. The output of the timer is shown in waveform (iv). To convert waveform (iv) into the PWM waveform required, an inverting signal (v) in synchronism with the apexes of the carrier waveform is also generated by the transputer. The inverting signal and the output from the timer are used as input to an 2-input Exclusive-OR gate, which then outputs the final PWM waveform (vi).

6.5.3 PWM Generator Hardware

The schematic block diagram for the regular sampled asymmetric PWM generator is shown in Fig.6.11. It consists mainly of two standard **inmos** link adaptors (LA) - the IMSC011, an Intel 8254 programmable interval timer, transistor-transistor logic (TTL) buffer/logic gates, and a stabilized 5 MHz clock circuit. The link adaptors are both configured in 'Mode 1', which is a serial to parallel peripheral interface format with handshaking lines. In the figure, LA1 is used for the output of data to the 8254 timer data bus, whereas LA2 is used for the output of control and address signals. TTL buffers (74LS244) were inserted in the link-to-link connections to provide extra protection of the link at one end, should the link at the other end become faulty. The buffers, although

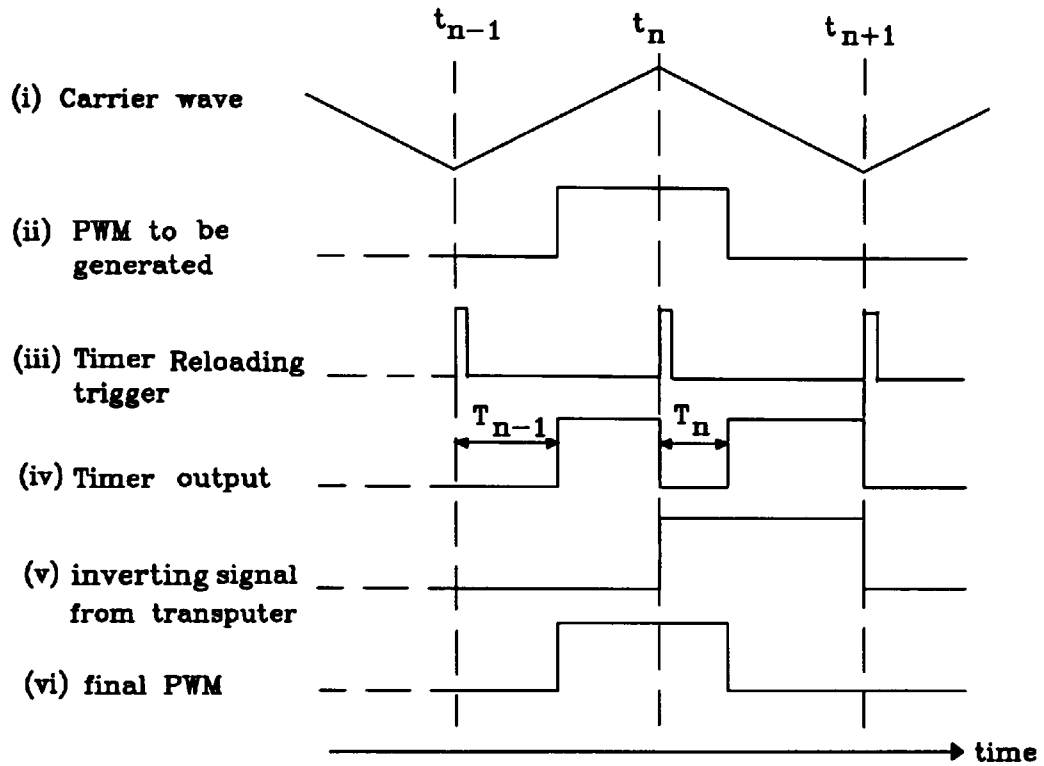


Fig 6.10 Timing diagram of novel method for the generation of regular sampled asymmetric PWM (one phase shown)

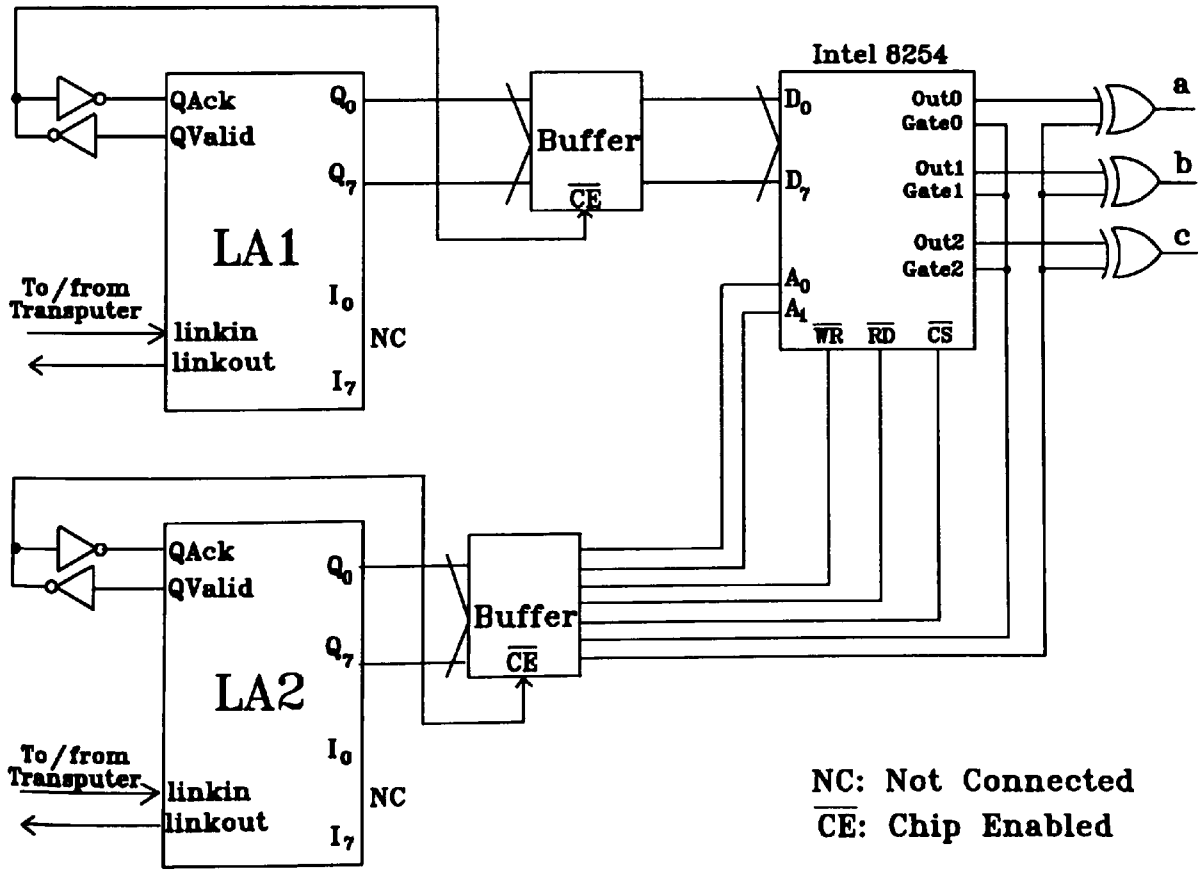


Fig 6.11 Schematic block diagram of PWM generating circuit

introducing typical TTL propagation delays, do not affect the speed of communication. The 'data valid' lines (**QValid**) of both link adaptors are connected to their respective 'data acknowledged' lines (**QAck**) via two TTL inverter gates (74LS04) connected in series. The propagation delays (typically 20 ns per gate) introduced by the two gates automatically provide handshaking.

The time delay introduced by the TTL inverter gate and a maximum delay of one clock cycle (1 μs) which is the time for the output of the timer to take 'low' after the gate signal received create small unwanted glitches in the PWM as illustrated in Fig.6.12. However, it is believed that such glitches, with a pulse-width of about 1 μs, would not cause any noticeable effect on the operation of the motor.

6.5.4 **Programming the Intel 8254 Timer** [Intel,1987]

The Intel 8254 Timer, which consists of 3 independent 16-bit timers/counters, is perfectly suited to the generation of a 3- phase PWM waveform. The functional block diagram of the 8254 timer illustrated in Fig.6.13 shows the 3 independent 16-bit counters that can operate at a count rate of up to 10 MHz. The clock input frequency of 1 MHz to the three timers of 1.0 MHz, was derived from the 5 MHz clock used in the **inmos** link adaptors by means of a divided-by-five frequency divider circuit. The gate input of each counter was used for monitoring counting activities such as start counting and prohibit counting. The timer has six modes ('mode 0' to 'mode 5') of operations available. It is found that 'mode 1', in which each counter behaves as a programmable, retriggerable one-shot, is most suitable for the generation of PWM. The combined 'extra hardware' and 'software overhead' requirements to generate a 3-phase PWM waveform by this mode was found to be minimal. For speed measurement, the event counting mode ('mode 0') was used. Only 'mode 1' and 'mode 0' for the programming of the timer will therefore be described.

6.5.4.1 Operating 'Mode 1'

A typical timing diagram for a 'mode 1' operation is shown in Fig.6.14(a). The output of the counter will go 'low' on the count following the receipt of a rising edge of the gate input. However, the output will always go 'high' on the last count. If a new count value is

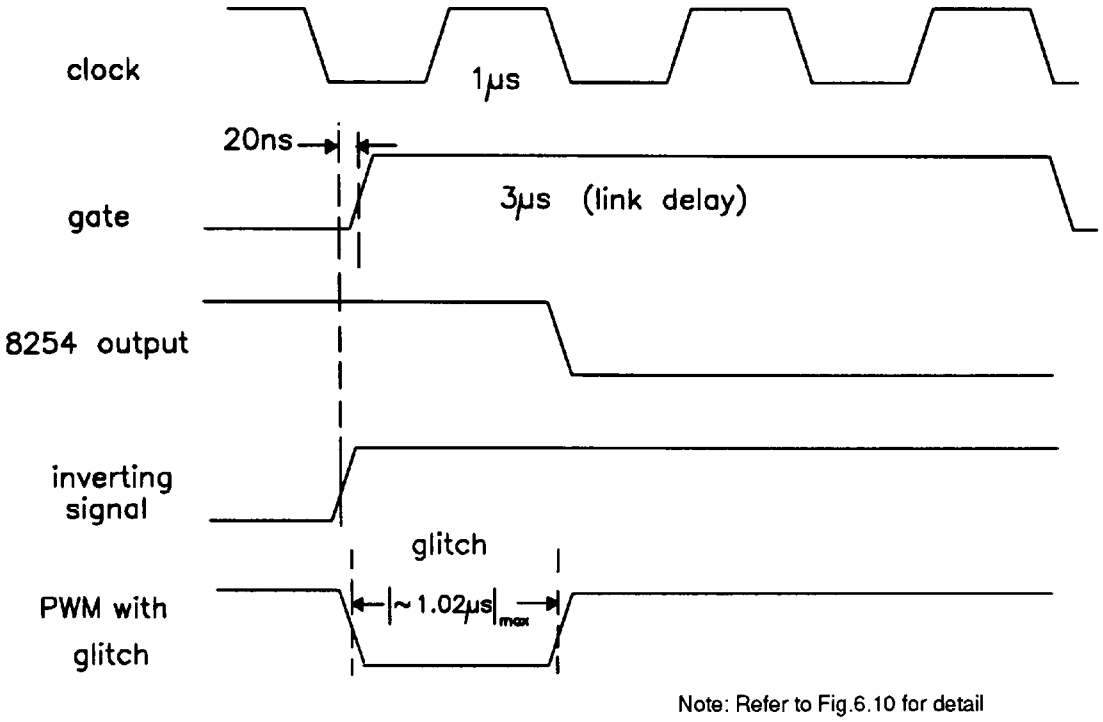


Fig.6.12 Formation of glitch due to delays in circuit

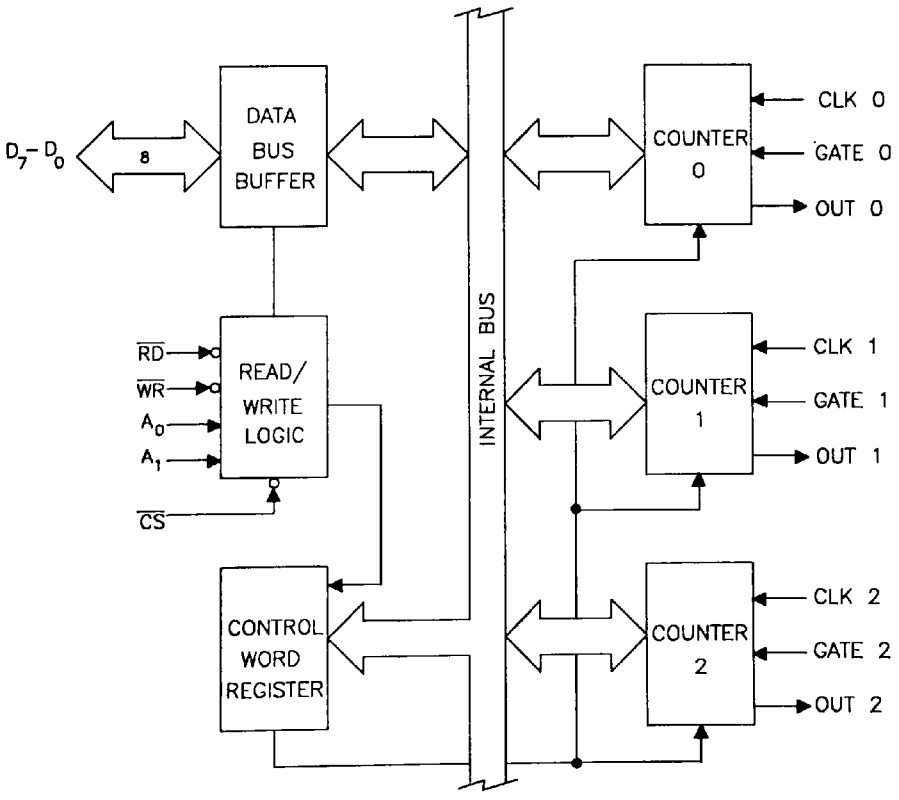


Fig.6.13 Functional block diagram of Intel 8254 timer

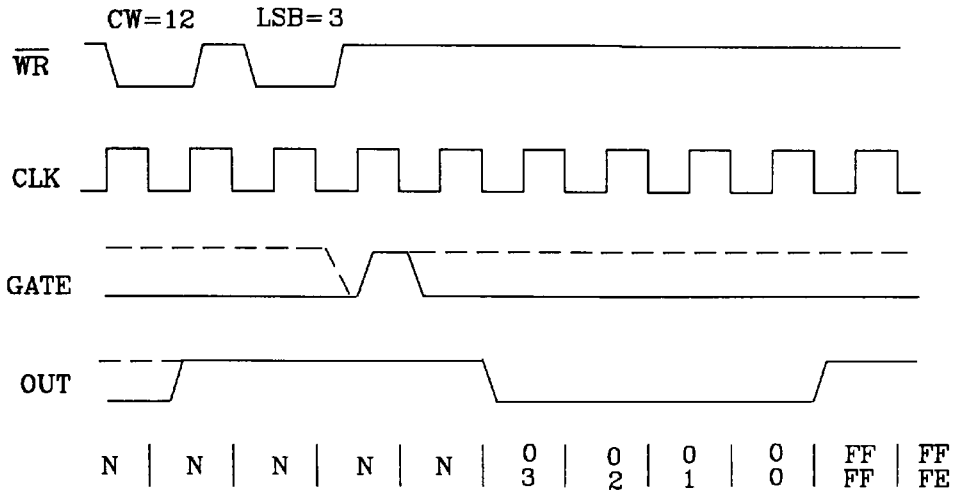
loaded to the timer whilst the output is low it will not affect the duration of the one-shot pulse until receiving the next trigger signal. The count values stored in the timer can be read at any time without affecting the one-shot pulse. Therefore, the timer outputs the modulated wave directly by timing out the loaded value, which corresponds to the pulse-width in its counter register.

6.5.4.2 Operating 'Mode 0'

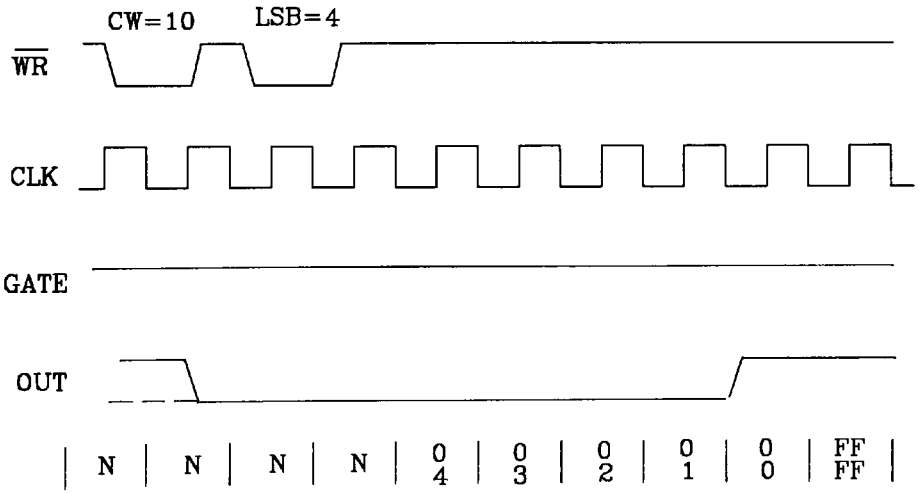
'Mode 0' is typically used for event counting and a typical timing diagram for this mode is shown in Fig.6.14(b). However, a modified method was used in the measurement of rotor speed. The output of the digital encoder of the motor served as the clock input to the 8254 timer. The number of pulses during a sampling interval of the motor speed is found by loading the timer with a full count (255 for an 8- bit binary counter) whilst at the same time enabling the timer to count down by setting 'Gate = 1'. Provided the timer does not count down to zero before the next sampling point, the value stored in the register of the timer can be accurately read by setting 'Gate = 0'. Therefore, if N is the value stored in the timer, then $(255\text{-}N)$ will be the number of pulses during the sampling period. This method of measuring the motor speed will be described in more detail in the next chapter.

6.5.4.3 'Control Word Register' Format of the Intel 8254 timer

Each counter of the 8254 is individually programmed by writing a data byte word into the Control Word Register. The Control Word itself specifies which Counter is being programmed. The Control Word is selected when A0=1, A1=1, $\overline{\text{CS}}$=0, $\overline{\text{RD}}$=1, $\overline{\text{WR}}$=0; and it has the following format as shown in Table 6.2(a). The two most significant bits D7, and D6 are designated with the SC0 and SC1 bits, which specify which of the three timers is selected, as shown in Table 6.2(b). The next two bits designate the RW1 and RW0, which specify the read/write mode according to Table 6.2(c). The next few bits are the M2, M1 and M0 bits, which are used for selecting the mode of operation. There are 6 possible modes of operation available, as tabulated in Table 6.2(d). The last bit, the BCD bit, specifies whether binary counter or coded decimal is used, as given in Table 6.2(e). Once the Counters have been programmed, they are ready for loading new count values. As with the Control Word, the counters are also selected by the address inputs A0, A1 as shown in Table 6.2(f).



(a) Mode 1



(b) Mode 0

Fig.6.14 Timing diagrams of 8254 timer (a) Mode 1 (b) Mode 0

MSB							LSB
D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RW1	RW0	M2	M1	M0	BCD

(a)

SC1	SC0	operation
0	0	select counter 0
0	1	select counter 1
1	0	select counter 2

(b)

RW1	RW0	operation
0	0	counter latching operation
0	1	Read/load LSB byte only
1	0	Read/load MSB byte only
1	1	Read/load LSB, then MSB byte

(c)

M2	M1	M0	operation
0	0	0	mode 0
0	0	1	mode 1
x	1	0	mode 2
x	1	1	mode 3
1	0	0	mode 4
1	0	1	mode 5

(d)

BCD	operation
0	binary counter 16-bits
1	binary coded decimal counter(4-decade)

(e)

A1	A0	operation
0	0	select counter 0
0	1	select counter 1
1	0	select counter 2
1	1	select Control Word

(f)

Table 6.2 Control word formats of Intel 8254 timer

6.5.4.4 Programming Timer Using occam

Considering the connections of the 8254 timer with the two link adaptors (LA1 and LA2) in Fig.6.11 to be mapped with the **occam** channels **to.timer.data.chan** and **to.timer.control.chan** respectively, the strobing of 8254 timer for data communication to/from the transputer is achieved by sending firstly the data word via the **to.timer.data.chan** channel, and then two control words (note: it is different from the Control Word for the 8254 timer) to **to.timer.control.chan** channel, the **control.word.1** and **control.word.2**, as shown below:

```
SEQ
  to.timer.data.chan ! data.word
  to.timer.control.chan ! control.word.1
  to.timer.control.chan ! control.word.2
```

The **occam** command **SEQ** refers to a sequential procedure is to be followed. The first **occam** output statement sends the byte, **data.word**, to the data bus. The data bus takes a few nanoseconds for settling. Since each channel output takes about 3 μs if a communication rate of 20 Mbit/s is used, the data bus is always valid before the next channel output. No introduction of delays is therefore needed. The next output statement generates a 'going low' edge to the $\overline{WR}$ pin of the timer by means of the one byte word **control.word.1**. The last statement completes the writing cycle by resetting the signal ready for the next cycle with the word **control.word.2**.

Referring to the wiring diagram of the 8254 timer and the link adaptor LA2 of Fig.6.11, the format of the word (in one byte) to control the timer signal to strobe the timer is as follows:

MSB **LSB**

—	inv	gate	$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A1	A0

Table 6.3 Word Format for strobing the 8254 timer

The MSB is uncommitted, and its value therefore does not affect the addressing process. Since the format of the BYTE in **occam** takes only integer in decimal rather than binary,

the corresponding decimal integers to strobe different timers are tabulated in Table 6.3 to facilitate the understanding of the following programs. These values are not unique, and other values are feasible.

Address / data (in byte)	byte in binary	byte in decimal
Address of Timer-0	11101000	232
Address of Timer-1	11101001	233
Address of Timer-2	11101010	234
Address of control word for timers	11101011	235
Date word for resetting timers	11111111	255
Data word for 'mode 1' of timer-0	00110010	50
Data word for 'mode 1' of timer-1	01110010	114
Data word for 'mode 1' of timer-2	10110010	178

Table 6.4 Address /data of Timers in binary and decimal digits

Thus, to program Mode 1 for **timer-0**, **timer-1** and **timer-2** of the 8254 timer, the occam statements required are:

```
SEQ
  to.timer.data.chan ! BYTE 50     -- timer-0
  to.timer.control.chan ! BYTE 235
  to.timer.control.chan ! BYTE 255
  to.timer.data.chan! BYTE 114     -- timer-1
  to.timer.control.chan ! BYTE 235
  to.timer.control.chan ! BYTE 255
  to.timer.data.chan ! BYTE 178    -- timer-2
  to.timer.control.chan ! BYTE 235
  to.timer.control.chan ! BYTE 255
```

The symbol '--' denotes a remark. The text following '--' will therefore be ignored by the compiler.

Therefore, it can be seen that a loading time of 9 μs (3x3μs) is required to write a byte of data. This is relatively slow compared with a typical write cycle in a conventional microprocessor system.

6.5.4.5 Loading the Timer Using occam

The following program segment loads the **timer-0** with a 16-bit data word, assuming the low byte is held in '**ta.lo**' and the high byte is held in '**ta.hi**':

```
SEQ
  to.timer.data.chan ! BYTE ta.lo  -- load low byte data
  to.timer.control.chan ! BYTE 224
  to.timer.control.chan ! BYTE 247
  to.timer.data.chan ! BYTE ta.hi  -- load high byte data
  to.timer.control.chan ! BYTE 224
  to.timer.control.chan ! BYTE 247
```

Therefore, it takes 18 μs (6x3μs) to load a 16-bit data to each timer, and 54 μs (3x18μs) to load the timers of for a three-phase PWM waveform at every instant of change of switching pattern. The minimum pulse width for the PWM waveform is therefore 54 μs.

6.5.4.6 occam Statements for The Generation of PWM Signals

As previously stated, the timer of the each phase requires a triggering pulse at the same time to initialize counting. The following occam statements send a toggling pulse to each gate input pin of the timers (Table 6.3) to start the 3 timers simultaneously,

```
SEQ
  to.timer.control.chan ! BYTE 255 -- 11111111 (binary)
  to.timer.control.chan ! BYTE 223 -- 11011111 (binary)
```

As described in section 6.4.2, an inverting signal is also required at every carrier period interval to produce the required PWM, this can easily be achieved by toggling the bits corresponding to 'inv' and 'gate' at the same time, such as the following statements,

```
SEQ
  to.timer.control.chan ! BYTE 127 -- 01111111 (binary)
  to.timer.control.chan ! BYTE 31  -- 00011111 (binary)
```

The width of the triggering pulse is thus about 3 μs, which is the practical communication time for one byte when the channel is configured at a communication rate of 20 Mbit/s.

6.5.5 Flowchart of PWM process

The flowchart of the PWM process is shown in Fig.6.15. From the figure, it can be seen that the set up process which consists mainly of the setting of the PROTOCOL motor, opening of occam user libraries and declaring of constants, is similar to that described in the flowchart of the monitor process. However, the 'heart' of the program illustrated by

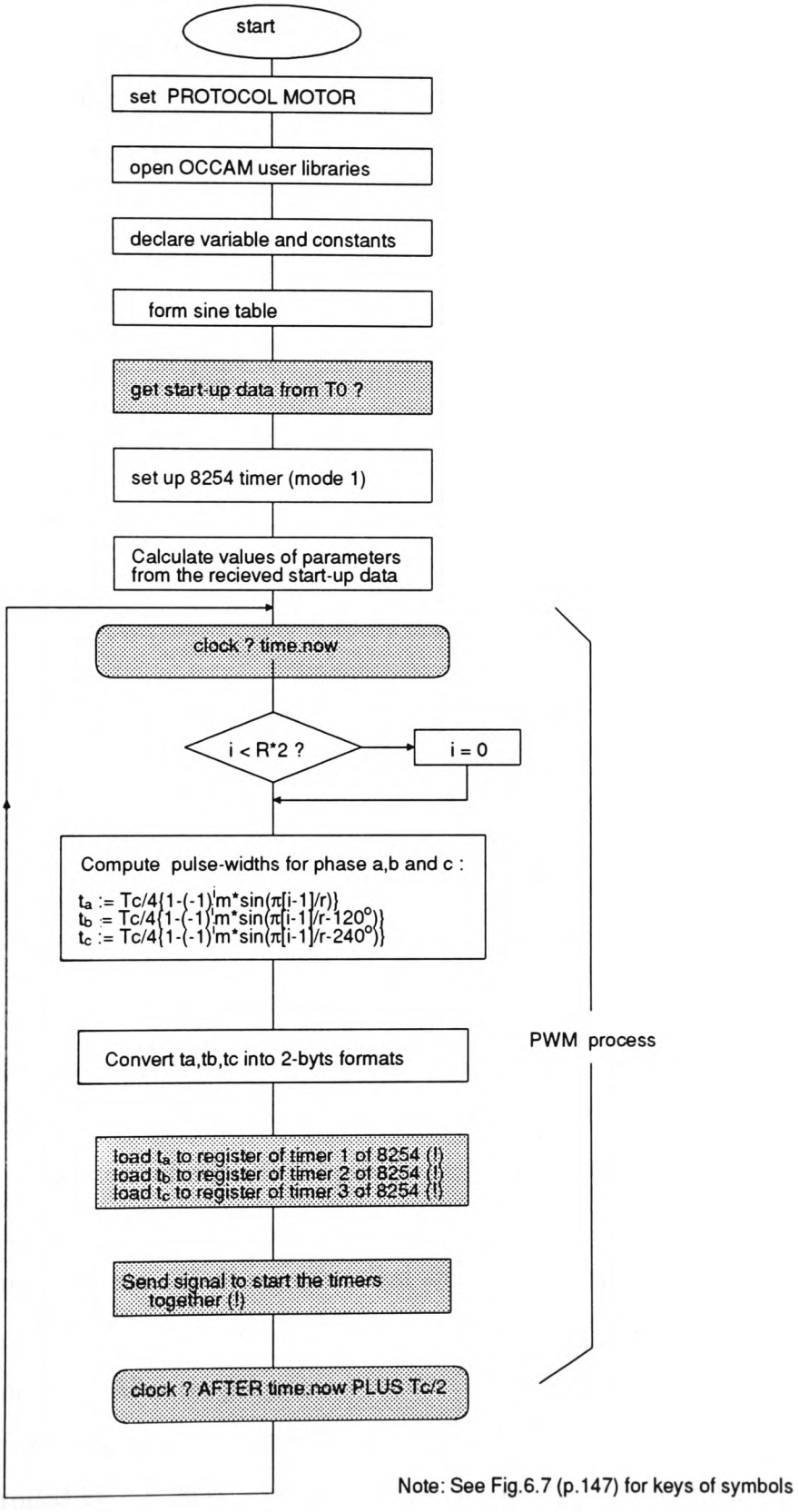


Fig.6.15 Flowchart of '**PWM**' process

the flowchart is the lower half indicated by the 'PWM process'. It is important to stress that the 'PWM process' is an entirely novel method of using the internal hardware timer of the transputer to generate a programmable triggering pulse and thus eliminates the need for an external timer. The shaded boxes indicated by represent internal inputs to the process from the internal transputer timer, which is given an `occam` name, `clock`. The statement, `clock ? time.now`, represents an 'always ready' input, in which the variable, `time.now`, takes the current value of the clock of the internal timer. The process will then execute the following statements in the flowchart, which are concerned with the computing of pulse-widths and the sending of the pulse-width values to the external timers until it arrives at the point of the next internal input from the internal timer, corresponding to the statement — `clock ? AFTER time.now PLUS Tc/2`. Now, `Tc/2` is the time unit equivalent of half of the carrier period. The action of this statement is such that the clock will give an input at exactly `Tc/2` (unit time) after the first clock input statement. Since the clock runs at a frequency of 1 MHz, the theoretical carrier frequency of the PWM waveform generated by this method is 2 MHz.

6.6 THE TRANSPUTER NETWORK SYSTEM

Two common transputer boards were used in the multi-transputer system described in this thesis — the `inmos` IMS B003 and IMS B008 boards [Imnos,1988]. The T800 transputer on the B008 board, hosted by an Intel 386 IBM compatible personal computer, and the four T414 transputers of the B003 board, form the multi-transputer control system for the induction motor. The T800 transputer of the B008 board, on which the Transputer Development System (TDS) runs, is called the 'host' transputer. As described in the previous chapter, the single T800 transputer provides the `occam` model at the program development stage. After the programs were developed, compiled and tested, the program codes were then loaded to the four T414 transputers on the B003. Since these transputers are controlled by the 'host' transputer, they are sometimes called the 'slave' transputers.

6.6.1 The IMS B003 and IMS B008 Transputer Boards

The IMS B003 board used is a double extended Eurocard containing four T414 transputers, each with 256 Kbytes of dynamic RAM. The board is one of the family of compatible evaluation boards, and is the first in this family to include more than one transputer. The B003 board is normally used with one of the evaluation boards such as the B008 board, from which programs are downloaded to the B003 board via a link. The transputers on the board are connected in a square configuration, which has a rotational symmetry, where Link2 of each transputer is connected to Link3 of the next transputer, a diagram of this interlink configuration is shown in Appendix-A.

The IMS B008 is a TRAnsputer Module (TRAM) motherboard that plugs into one of the slots inside the IBM PC-AT or compatible machines. The B008 board has slots for up to ten TRAMs. The TRAM is the inmos board level transputer and is provided with a simple and standard interface. The TRAMs integrate a processor, memory and peripheral functions, thus allow powerful and flexible transputer-based systems to be produced with a minimum of design effort by means of TRAM motherboards. A TRAM of type IMS B414, which consisted of a T800 transputer and 2 Mbytes of RAM memory, was used in the system described. The B414 TRAM occupied only two of the ten slots available on the B008 board. These slots may therefore be used for additional TRAMs for further development. More technical detail of the B008 and the B414 are also found in Appendix-A.

6.6.2 Inter-transputer Connection

The transputer link connections were achieved by using the inmos standard link cable. The standard link cable consists of one incoming and one outgoing signal wire, with each signal wire twisted together with its own ground. This design helps to reduce the noise that appears between the signal and ground (different mode noise) as well as maintain a constant characteristic impedance along the cable. As the transputer is designed to work in an 'electrically quiet' environment, it is important to avoid using long cable links. In addition, the link cables should be kept as far away from the inverter and the motor as

possible, as they may generate radio frequency interference (RFI). Hence, in this project, the length of the cable links used was no more than 45 cm.

Whilst direct link connections may be used for transputers, provided they are not separated by more than 300 mm, it is always necessary to use the standard **inmos** link adapters for the connection of transputers with other non-transputer devices that use a parallel bus for communication. The link adapters were relatively expensive devices (£7 each in 1989). A total of four were used in the PWM waveform generator circuit and the speed measurement circuit used in the investigation. However, it is expected that the cost of such devices would fall rapidly in the coming years.

6.7 IMPLEMENTATION OF ON-LINE INTERACTIVE DRIVE SYSTEM

6.7.1 Allocation of Processes to Processors

After the 'logical model' of the interactive drive system was developed in the form of **occam** program within in the transputer development system, the **occam** programs were debugged, compiled and tested. The different processes in 'logical model' were then mapped to different processors. This mapping process would not affect the logical outcome of the program, but would affect the performance of the system. Since there are four transputers on the B003 board, and two **occam** processes, one of which the '**monitor**' runs on the 'host' transputer, then only one of the four transputers of the B003 board was needed. However, **link0** of the 'root' transputer **T0** is designed by the manufacturer to be connected to the 'host' transputer. Moreover, link2 and link3 are connected internally (see Appendix-A), only link1 is available for addressing external timer. As described in earlier sections, two links are necessary for addressing the external timer. Thus, at this stage of the design of the system, the 'root' transputer **T0** was used as a dummy processor, and the next transputer **T1** was used as the processor for PWM generation. The remaining two transputers were not used. The function of transputer **T0** was to pass data between the 'host' transputer **T1**. This process which merely consists of passing data was given the name '**control**'. Because of its trivial nature, its operation

will not be described further. However, it will be seen in the next chapter that this '**control**' process was extended to incorporate the 'vector control' at a later stage.

6.7.1.1 The MOTOR PROTOCOL

Several types of data were required to be communicated between the transputers within the multi-transputer network for the control of the induction motor drive system. A simple 'stop the motor' command from the host computer to the network, for example, is nothing more than a 'tag'. However, a measured value of rotor speed in order to transfer back to the 'host' transputer for processing, may require floating point number for the necessary accuracy. The operational parameters for the PWM generator that consists of such parameters as carrier frequency, frequency ratio and modulation index, are therefore best to be transmitted as a packet of data.

The **MOTOR PROTOCOL** is a variant protocol that specifies a number of data types including simple 'tags', **INT** (integer), **REAL32** (32-bit real number) and a combination

Type of tags	Name of tags/protocols, and type of data
On-line command tags	change.reference.speed.tag change.carrier.freq.tag change.pwm.parameter.tag forward.tag reverse.tag dynamic.brake.tag
Operational tags	full.speed.tag acknowledge.tag finish.tag
On-line command Protocols	pwm.parameter;INT;INT;REAL32 speed.ref;REAL32 flux.ref;REAL32
Feedback command Protocol	speed.measure;REAL32 position;REAL32 Iabc;REAL32;REAL32;REAL32
Protocols	machine;data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32 startup.data;REAL32;REAL32;REAL32;REAL32;INT carrier.freq;REAL32

Table 6.5 Types of Commands in the PROTOCOL MOTOR

of **INT** and **REAL32**. There are several groups of message in the system design such as the on-line command tags, the operational tags, start up data, PWM parameters, reference speed, measured speed and measured currents. The **PROTOCOL MOTOR** was designed in order to allow for different types of data to flow along the links. The links connecting the transputers are all assigned with the **PROTOCOL MOTOR**. It is found that the character '.', which is allowed to use in naming of a variable in **occam**, provides a very powerful means for documenting the program. Table 6.5 lists the types of commands and the name of the protocols and 'tags' in the **PROTOCOL MOTOR**.

6.7.1.2 Network Topology

Two types of **occam** program were developed — the **EXE** that runs in the host transputer, and the **PROGRAM** that runs in the transputer network. The **PROGRAM** contains two Separately Compiled programs, abbreviated as **SC**, and the program statements for the configuration of the network. An **SC** contains the program codes to be run on a single processor. The topology of the network is shown in Fig.6.16. The process '**monitor**' is to be run on the T800 'host' transputer. Transputer **T0** executes the '**control**' process, whereas transputer **T1** executes the '**pwm**' process. Each hardware link consists of two wires, one for outgoing data, and one for incoming data. Each wire is mapped to an **occam** channel. A channel connecting two processes on two different processors must be placed at the input link address on one processor and at the output link address on the other processor. A channel can be connected to an environment outside the scope of the configuration being described by means of the link adaptors to the outside world. The **CHAN OF** *type* (**CHAN** is abbreviation for **CHANnel**) command allows only a certain type of data to pass through the channel, where '*type*' specifies the type of data that can be passed. Two types of data are used for the **occam** channels in the method described here. One is the **CHAN OF BYTE** and the other is **CHAN OF MOTOR**. The channels assigned with the **occam** instruction **CHAN OF BYTE** pass only data with the type **BYTE**. The **CHAN OF MOTOR**, when set up by the **PROTOCOL** command, allows the user to setup the type(s) of data that the channel can support.

In Fig.6.16, **Link0** and **Link1** of **T1** are used as data and control bus for the 3-phase PWM generator. Each serial link is transformed to parallel bus by means of an **inmos**

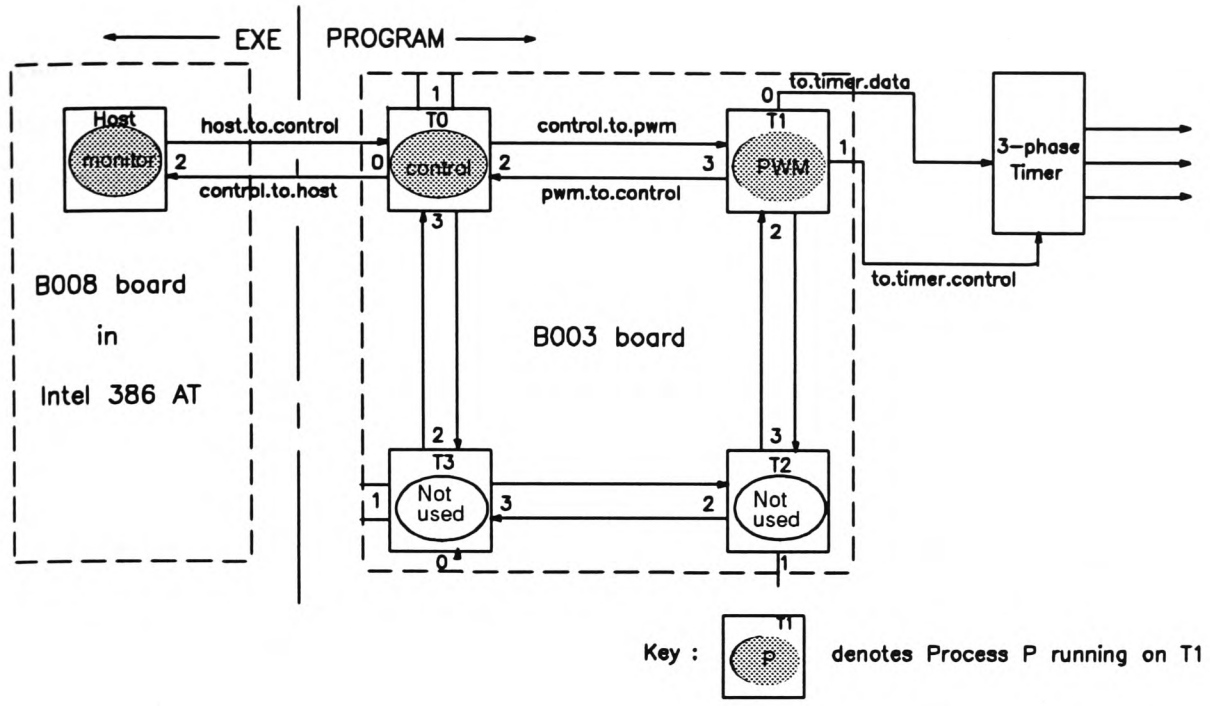


Fig.6.16 Topology of B003 transputer network

Processor	Channel Name	Link Number	Direction
0	control.to.host host.to.control control.to.pwm pwm.to.control	0 0 2 2	out in out in
1	pwm.to.control control.to.pwm to.timer.data to.timer.control	3 3 0 1	out in out out
2 and 3	Not configured		

Table 6.6 Configuration map of B003

link adaptor. Since the data and control signals are always output from T1 to the PWM generator, the outgoing wires of Link0 and Link1 are mapped with the occam channels to.timer.data and to.timer.control respectively, whereas the incoming wires of Link0 and Link1 are not used and therefore not mapped. The mapping of the channels is summarized in Table 6.6.

6.7.2 Program Flow Diagram for the Implementation of Interactive PWM Drive System

Whereas the conventional flowchart is ideal for the illustration of a sequential program, the implementation of programs running in parallel may best be described by a "program flow diagram" — a name coined by the author. In a program flow diagram, the flow of the programs running in parallel, characterized by the communications between the programs (processes), is shown. The program flow diagram of the three occam processes running concurrently on three separate processors is shown in Fig.6.17. The process 'monitor' that is an EXE runs on the T800 'host' transputer within the TDS environment. The processes 'control' and 'pwm' are SC programs that run on the T414 transputers designated T0 and T1 of the B003 board respectively. These two SC programs, together with the configuration statements, are put under the PROGRAM fold. The process 'monitor' provides the user interface by means of the two keyboard and monitor occam channels, which provide the access to the actual keyboard and monitor of the host computer under the TDS environment. The values of the PWM parameters, which include modulating frequency (f), the modulation index (m) and the frequency ratios (R), are inputted to the process 'monitor'. The processes, 'control'and 'pwm', which are SCs, run on the processors T0 and T1 of the network respectively. The process 'control' manages the flow of data, whereas the process 'pwm' generates the PWM in real-time. After the initialization procedures, the process 'monitor' is ready to output the startup data. The process 'control' is also ready to input the startup data from the process 'monitor' whereas the process 'pwm' is ready to input data from the process 'control'. As shown in the Fig.6.17, 'control'and 'monitor' communicate with each other via the channels at the instants corresponding to input (symbol '?') and output requests (symbol '!'). Communications take place when the processes at both ends of the channel are ready. Each process then runs a looping procedure to detect any input of

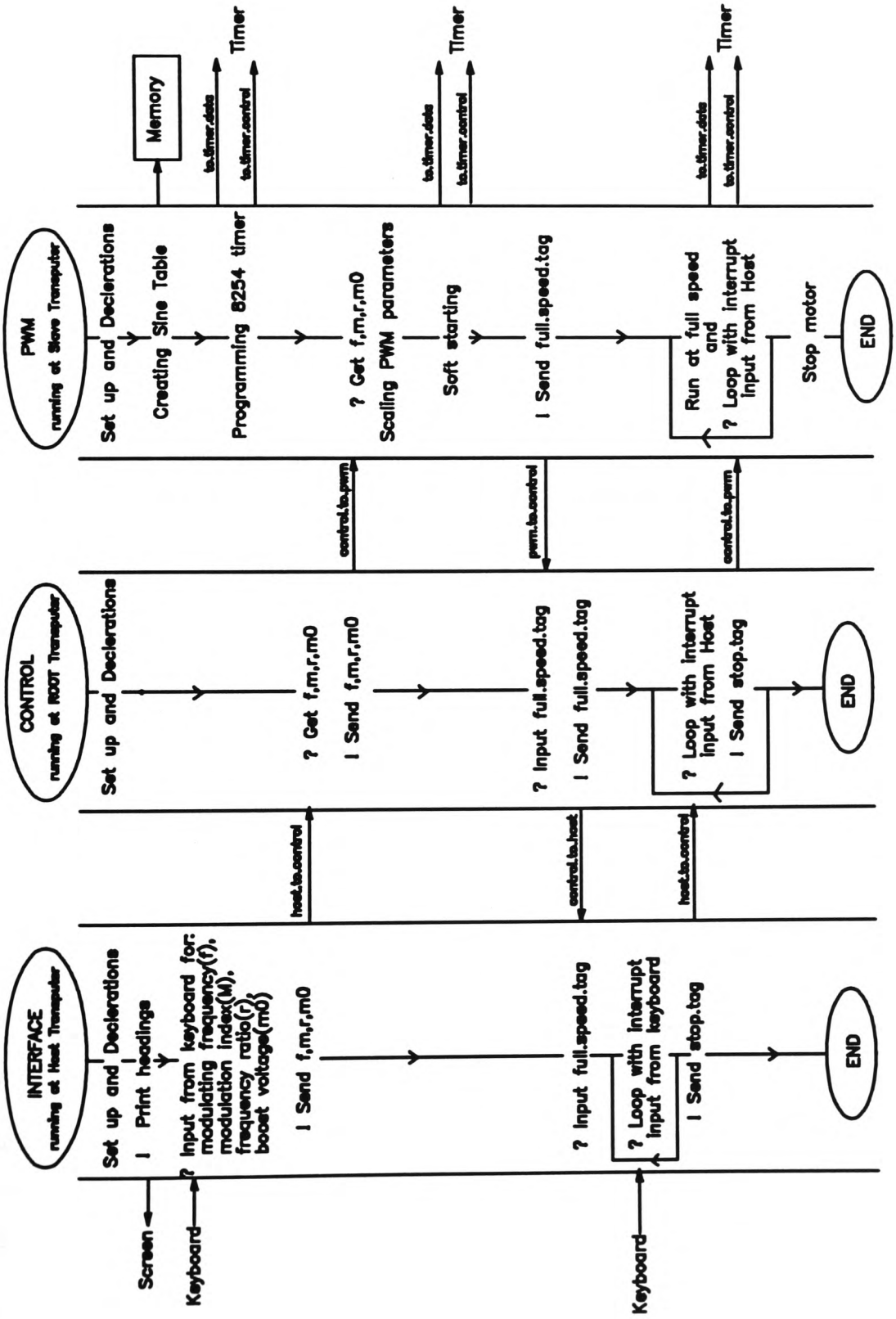


Fig.6.17 Program flow diagram for 3 OCCAM processes running on 3 processors

command, such as a change in the value of the modulating frequency (f) or the frequency ratio (R), from the keyboard by the user. The process '**pwm**' then computes the switching instants, 'on-line', and sends the values of switching instants to the timer. If the user command is to '**quit**' the system, the process '**monitor**' will send a '**finish**' tag to the network. The network will then respond by sending back an '**acknowledge**' tag. The processes will then be properly terminated.

In parallel processing, it is important to ensure that all the processes terminate properly. The termination procedure of the 3-transputer network used is shown in Fig.6.18. A '**finish.tag**' is sent from the host transputer to the two transputers in the network, which will send an acknowledgment tag, '**ack.tag**', back to the host, to complete the termination process.

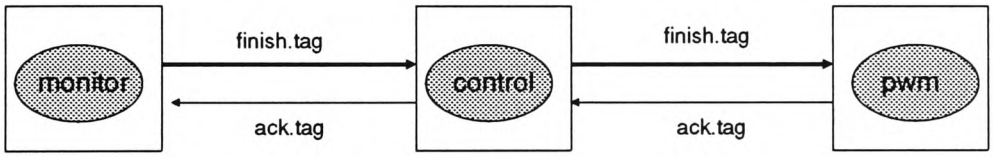


Fig.6.18 Termination procedure of multi-transputer system

6.8 EXPERIMENTAL RESULTS

6.8.1 Display Menu of Interactive PWM Drive System

The monitoring display menu of the drive system exploits the facilities supported by the host computer. The display menu of the drive system during a typical operation is shown in Fig.6.19. The program updates the screen every 0.5 second.

6.8.2 PWM Waveform Generation in Real-time

The result of the generation of one phase of a typical three-phase regular sampled asymmetric PWM waveform by the transputer, in real-time, is shown in Fig.6.20. The design system provides the facility for modulating frequency, frequency ratio, and modulation index of the PWM waveform to be changed on-line via the keyboard. The soft-start process, which was also designed in the system at this stage, is described in greater detail in the next chapter. The corresponding values of the pulse-widths, which are calculated on-line by the transputer, are then calculated and sent to the timer for PWM signal output. A proper termination procedure for the processes running in the network was also provided.

6.9 INTERIM CONCLUSION

The software development time for the interactive system was remarkably short under the TDS. This was mainly due to the relative ease of programming the system using the high level **occam** programming language and the facility provided by the TDS, which allows the programmer to make use of the host computer's resources such as the filing system. It is also important to realise that the flexibility and the low cost of hardware development offered by the TDS and the transputer are found to be very eminent advantages over conventional multi-processor systems, such as that developed by Harashima [Harashima,1985].

The theoretical upper limit of the carrier frequency of the PWM waveform in the method reported in this chapter is in the order of 2 MHz, which could be realised by the high priority internal timer of the transputer. The practical value, in contrast to the theoretical value, is greatly reduced by such factors such as the time to address the external timer and the clock frequency of the external timer. For example, the time required to address and load data to one of the three timers of the 8254 timer chip through the transputer serial link requires is 6 μs for a link speed of 20 Mbit/s. Thus, the maximum carrier frequency for a single phase PWM is therefore limited by 1/(6μs) or 167 kHz. Further reduction on

the carrier frequency occurs when all the 3 timers of the 8254 chip are used. However, by using the conventional 'memory-mapping' for communications between processor and peripheral devices, significantly lower communication time can be achieved. Nevertheless, most of the commercial inmos boards are not designed with a 'memory-mapping' facility. However, it is believed that the new generation of transputers — T9000 (released in late 1992), which has a communication speed of the order of ten times greater than the T800, could eventually provide a direct solution to the problem address.

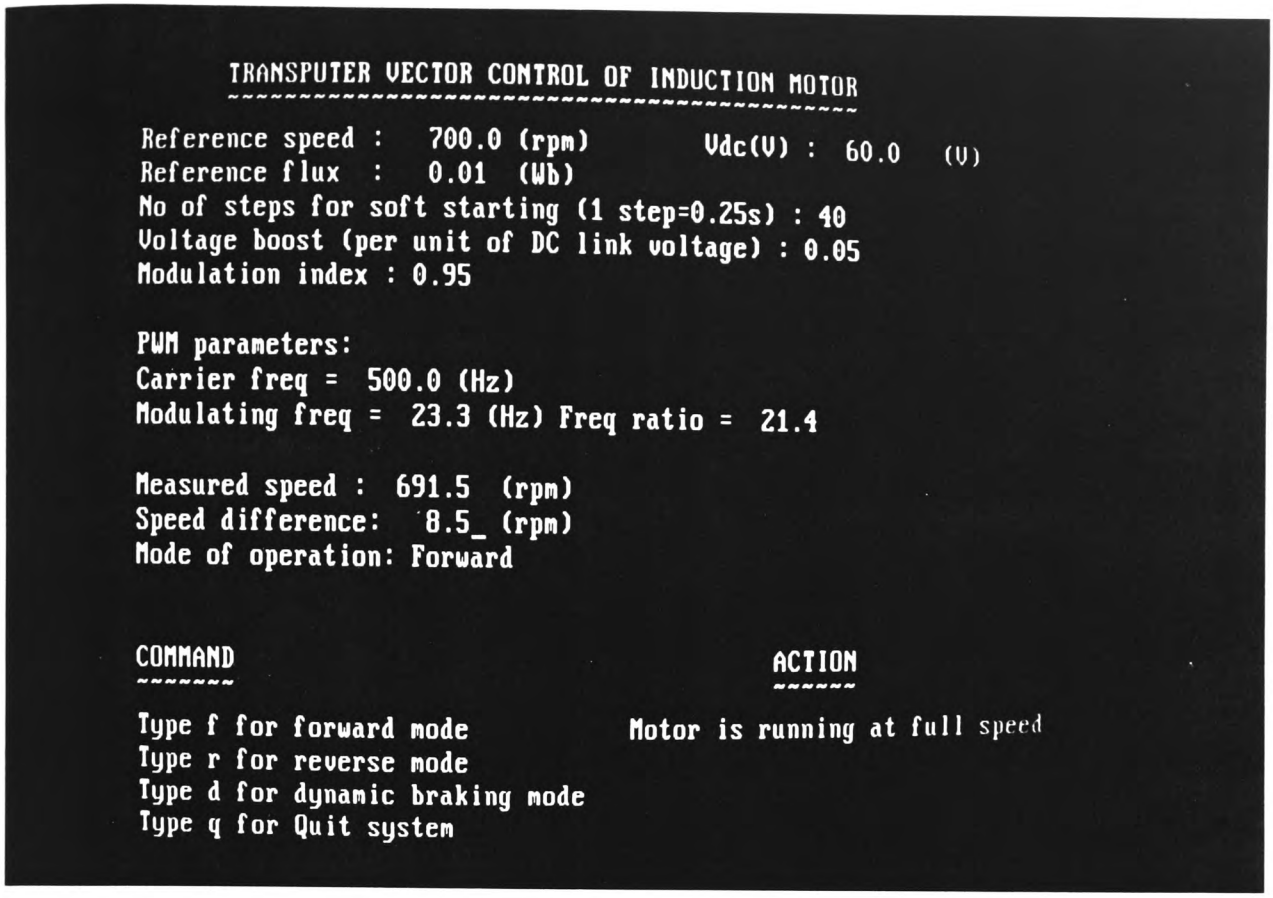


Fig.6.19 Display menu of interactive PWM drive system

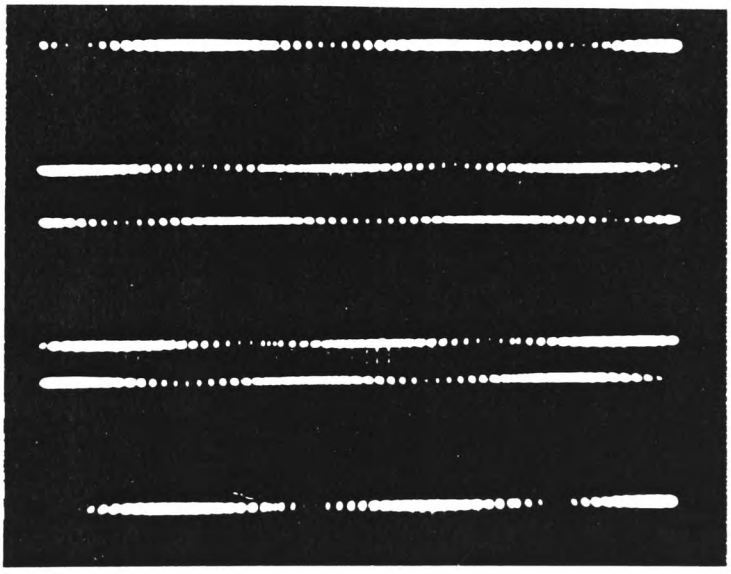


Fig.6.20 Generation of 3-phase PWM waveforms in real-time
(f=33Hz, R=21, M=0.9)

Chapter 7 : Further Implementations of The Interactive Induction Motor Drive System by a Five-transputer Network

7.1 INTRODUCTION — PARALLELISM IN A MOTOR DRIVE SYSTEM

In general a motor drive exhibits a task-oriented 'parallelism', in which each process is responsible for different tasks, in contrast to the sharing of the same task. A typical vector control may therefore be seen as a task-oriented parallelism as depicted in Fig.7.1. Typically the tasks can be classified according to their priority and execution time. The tasks requiring very high speed and high priority consist mainly of the signal processing and conditioning required for the power converter, and fault handling facilities. Generally, in existing systems, these tasks are implemented by hardware circuits, because if the task were implemented using a processor, the processor will be burdened and would not be able to carry out any low priorised task. However, using a multi-processor approach overcomes this problem and allows the implementation of high and low priority tasks. Drive systems also consist of medium speed and priority tasks such as the transformations of frame of reference and vector control algorithms. The low speed and low priority tasks that can be identified in a drive system mainly consist of rotor speed and position measurements and the handling of on-line user input/output by means of the keyboard and screen. These tasks are characterized by their relative large time constant or slow response time.

This chapter is mainly devoted to the description of a novel transputer implementation of control strategies for the induction motor drive. The specifications of (c) to (e) outlined in the last chapter are fully implemented. In addition, the implementation of the vector control strategy presented in chapter 3 is also described. As previously described, for open loop control drive systems, only the tasks of '`monitor`', '`control`' and '`pwm`' are required to be implemented. However, for closed loop control system, and for open loop vector control system, more 'tasks' are required to be added to the `occam` model during software development. From the view point of an `occam` model of the drive system, the additional processes or tasks, namely the '`speed`', '`current`' and '`vector control`', have to be included in the new drive model. These additional processes could either be mapped onto the two processors, `T0` and `T1`, which were already being used in the system described earlier, or the additional processes could be mapped on to the two

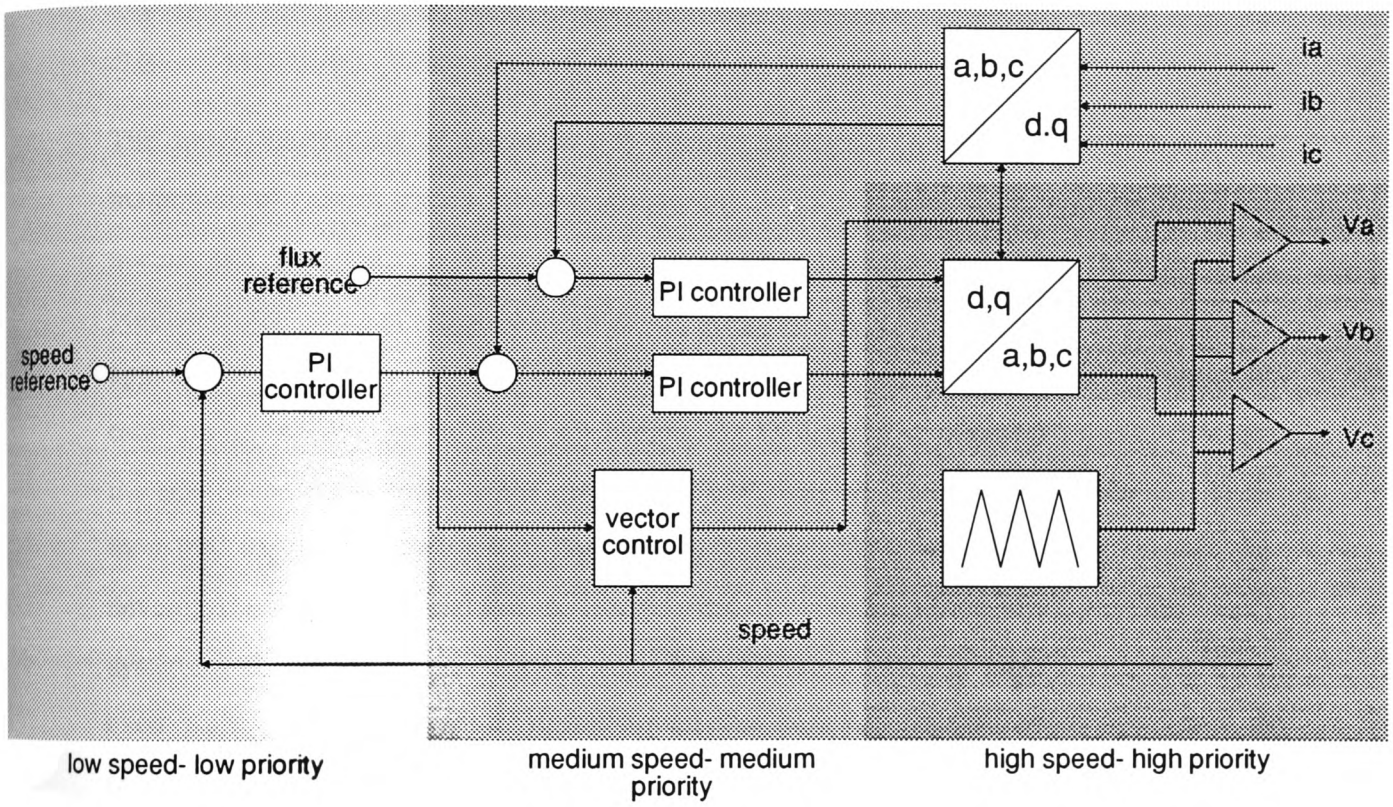


Fig.7.1 Parallelism existing in a vector control drive

remaining processors, `T2` and `T3`, which are on the same B008 board. Since `T1` was already committed to real-time PWM generation requiring high priority attention, no attempt was made to map the additional processes onto it. Therefore, the addition of more processes to the system naturally invoked the use of the remaining two transputers available on the B003 board.

7.2 MODIFICATION OF THE '`PWM`' PROCESS TO INCLUDE SOFT-STARTING

The simulation results of direct-on-line starting presented in section 4.5 show that the magnitude of current inrush into the motor may be in the range of six to eight times the normal full load current. When currents of such magnitude are drawn from the line, problems arise due to momentary voltage dipping that can seriously affect other motors or apparatus already in service and connected to the same supply. The overloading and speed oscillations encountered by the motor itself under DOL must also be considered. Although the size of the motor used in the project, 2.2 kW, was within the limit on the size of an industrial AC motor for direct-on-line starting (below 3 kW), the soft-starting process was developed and implemented to avoid the adverse effect of the inrush current on the supply system of the research laboratory in which the investigation was carried out. Moreover, even with the 2.2 kW motor, it was found that soft-start offered advantages such as reducing the effect of motor starting on the laboratory power supply, which was in fact connected to several workstations and personal computers in other teaching laboratories.

It is described in section 3.2 that constant torque drive applications require a constant voltage-to-frequency (*v-to-f*) ratio from a VVVF supply. In a PWM system, the voltage output is proportional to the modulation index, *M*, provided that the harmonic components of the PWM waveform are small enough. Therefore, a constant *v-to-f* ratio can be effected by controlling the modulation-to-freqency (*M-to-f*) ratio. Since the voltage drop across the stator winding impedance is relatively significant when the supply voltage is low, the 'voltage boost' procedure is usually adopted to compensate this stator impedance voltage drop. Thus, the graph of modulation index against frequency with 'voltage boost' is shown in Fig.7.2.

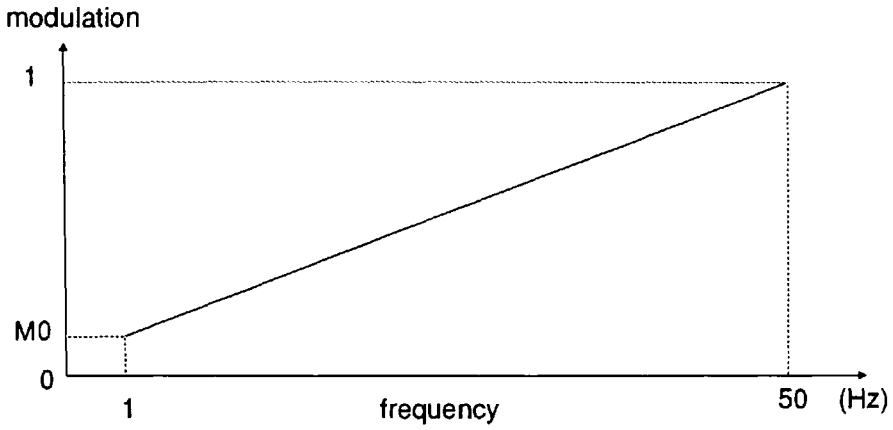


Fig.7.2 Controlled characteristic of modulation index
against operating frequency with voltage boost

The value of 'voltage boost' is indicated in Fig.7.2 by M_0 in the system developed. The lower frequency limit implemented is 1 Hz, because it is impossible to start from zero frequency in practice — which would require the timer to be loaded with an infinite value.

To make the system flexible, the values of the soft-start parameters are allowed to be inputted by the user. The method of the PWM generation during soft-starting is similar to the method described in section 6.5. However, the frequency and modulating index are computed at every step increment of frequency. For example, suppose N is the number of steps required to start the PWM waveform from 1 Hz to the reference frequency, $f.ref$; M_0 is the required voltage boost, and M is the required modulation index at reference frequency, then the processor would be required to compute in real-time the frequency, $f.ramp$, and modulation index, $M.ramp$, at each incremental step of frequency. The required values of $f.ramp$ and $M.ramp$ are therefore determined as follows:

$$f.ramp := f.ramp + f.step, \quad \text{where} \quad f.step = \frac{f.ref - 1.0}{N} \tag{7.1}$$

$$M.ramp := M.ramp + M.step, \quad \text{where} \quad f.step = \frac{M.ref - M0}{N} \tag{7.2}$$

The actual flowchart of the soft-start process can be seen in Fig.7.3.

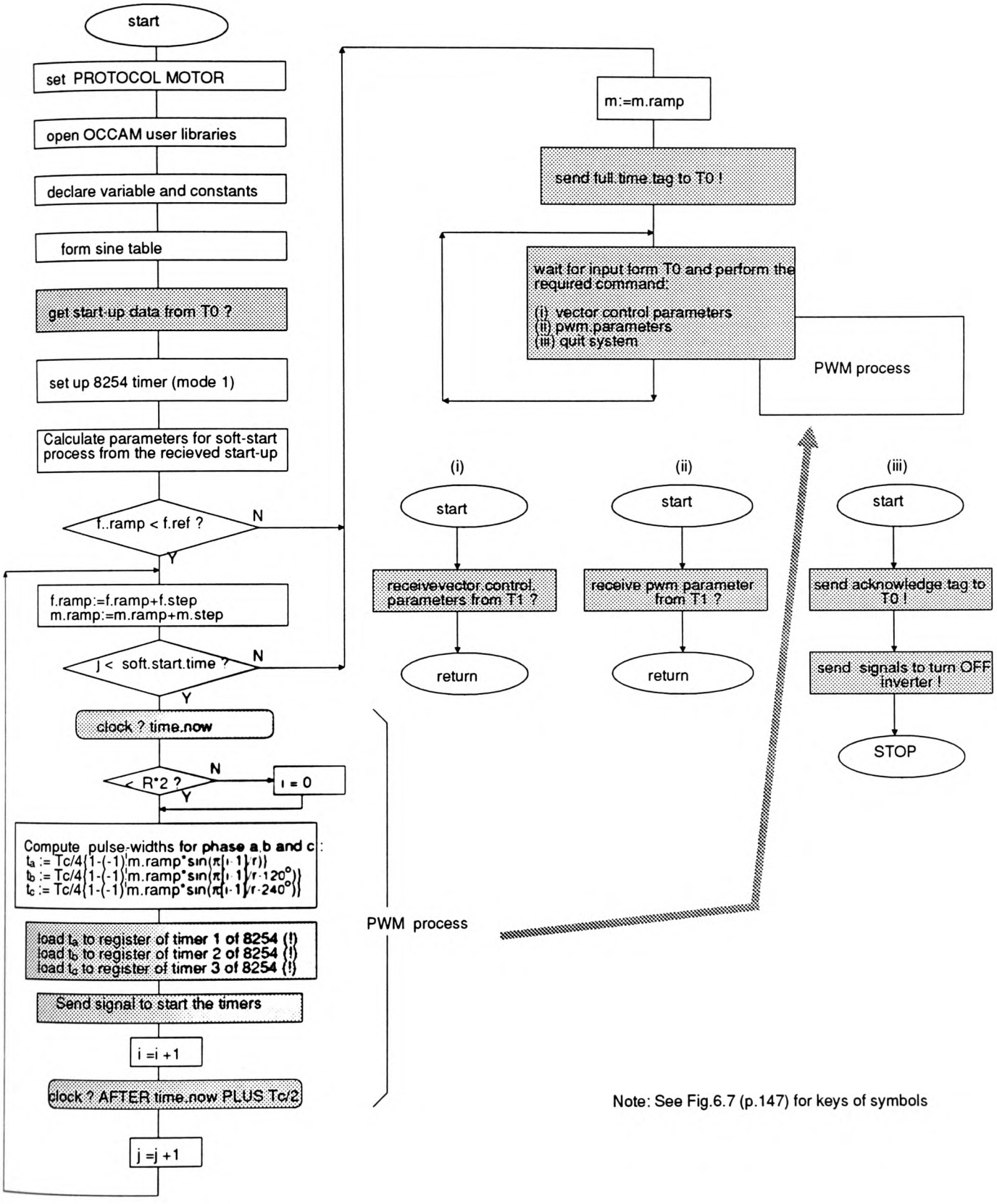


Fig.7.3 Flowchart of the '**pwm**' Process including soft-starting

7.3 'speed' PROCESS

7.3.1 Speed and Position Feedback Circuit

In a drive system, it may be necessary for speed to be fed back to the system for data acquisition such as information for displaying the result on the screen and sending it to the plotter. Usually, a moderate degree of accuracy in the measurement of speed may be sufficient. However, in systems which employ advanced control strategies such as vector-control [Leohnard,1985], the speed, and sometimes the position, of the rotor are required accurately at the sampling time. The resolution of the speed measurement is therefore important for high-performance drive systems. On the other hand, however, the required high resolution of the speed encoder can add significantly to the overall cost of the drive system. The built-in encoder system for the induction motor used in the investigation provided TTL-compatible outputs of 1024 pulse/rev plus a zero-marker. This resolution, however, may not be considered adequate for a very high-performance drive [Leonhard,1985]. To increase the resolution of the encoder, the outputs of the normal and quadrature signal of the encoder, usually used for direction sensing, were made use of. In fact, the two signals were inputted to an Exclusive- OR gate, as shown in Fig.7.4. This increased the resolution rate to 2048 pulses/rev.

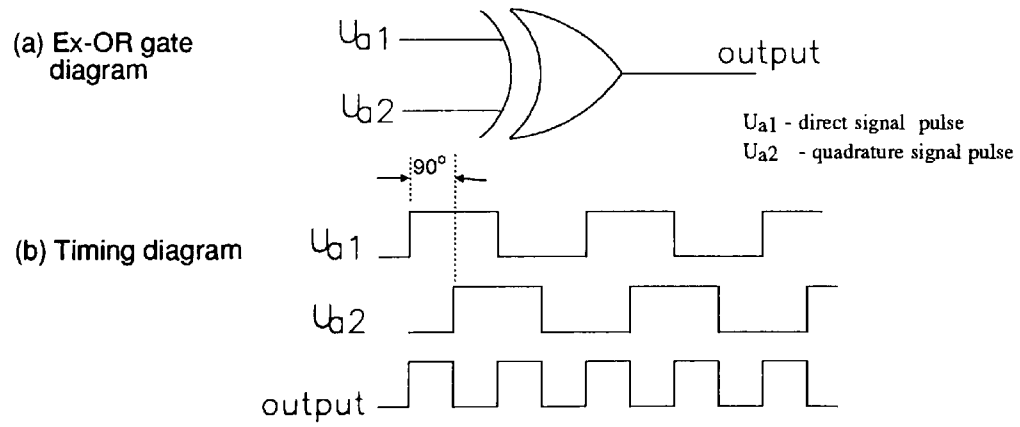


Fig.7.4 Increase of resolution of encoder by a factor of two

The block diagram of the feedback interfacing circuit is shown in Fig.7.5. The interfacing technique used is very similar to that for the PWM generating circuit. The interface consists of two standard **inmos** link adaptors (LA) chips, IMSC011, an Intel 8254

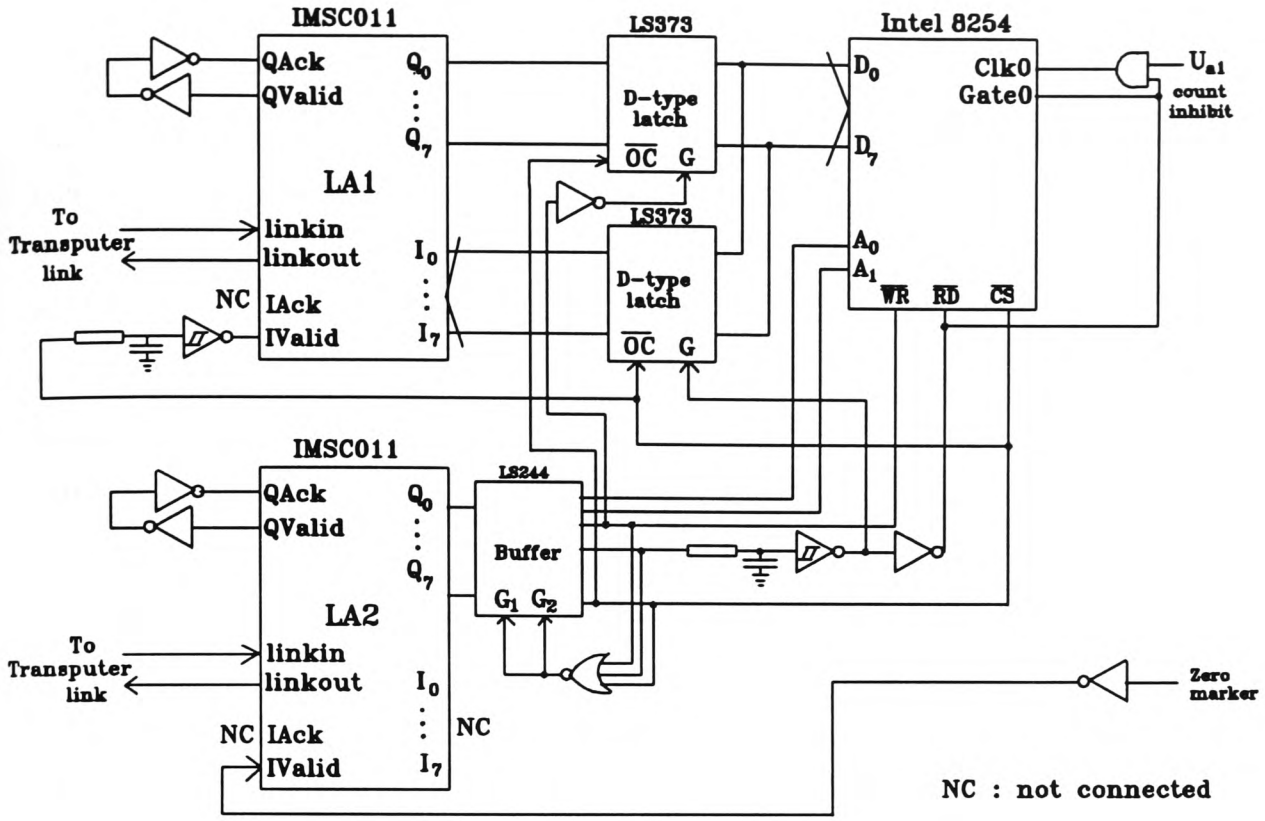


Fig.7.5 Block diagram of speed measurement circuit

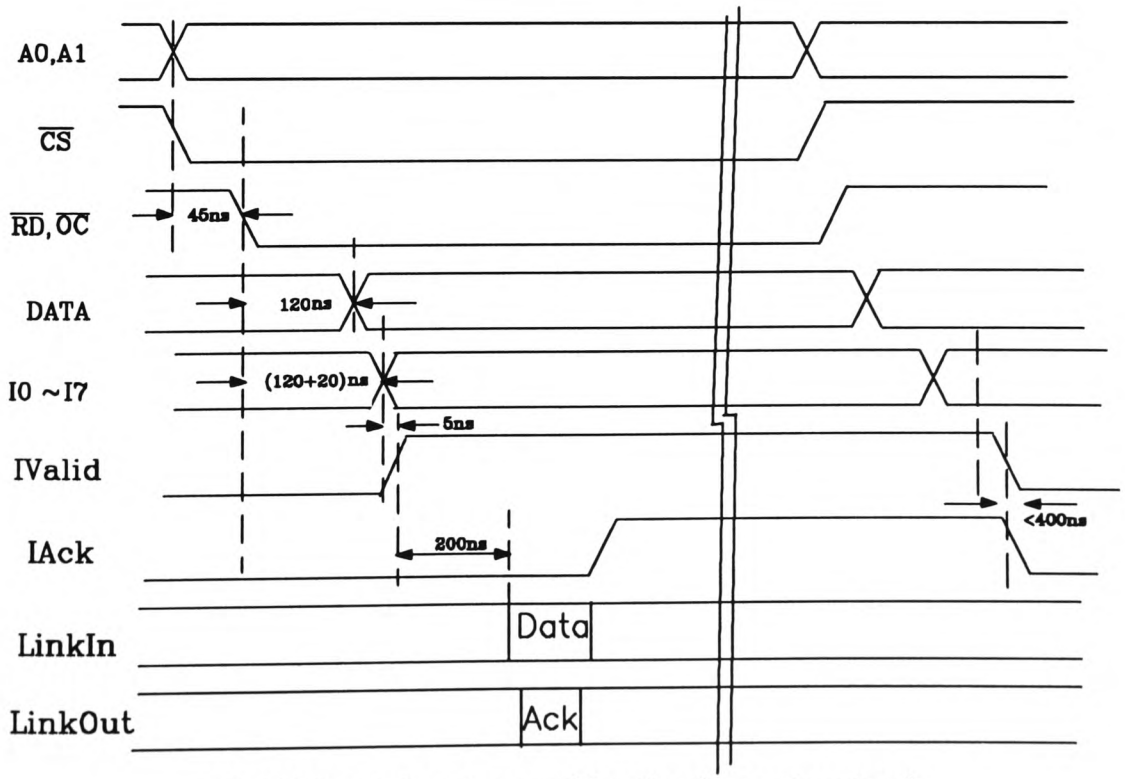


Fig.7.6 Special timing consideration for read cycle of the 'speed' process realised by the circuit of Fig.7.5

programmable interval timer, TTL buffers and logic gates. The link adaptors are both configured in 'mode 1', which functions as a parallel peripheral interface with handshaking lines. The upper link adaptor (LA1) is used for the input and output of data to the 8254 timer data bus, whereas the lower link adaptor (LA2) is used for the output of the control and address signals to the 8254 timer. To control the direction of the data flow of the data bus of the 8254 timer, a D-type latch is connected to the output bus (`Q0-Q7`) of LA1, and another D-type latch is connected to its input (`I0-I7`). The associated circuits ensure that the D-type latches will not be enabled at the same time. The control and address signals are designated as shown in the table below.

WR	RD	D-TYPE LATCH 1	D-TYPE LATCH 2
1	0	disabled	enabled
0	1	enabled	disabled
1	1	high impedance	not allowed
0	0	high impedance	not allowed

Table 7.1 Truth table for write and read cycle of speed measurement circuit

The **IValid** line is connected to the zero marker of the encoder. This, together with an appropriate `occam` program in priorized mode, could enable the transputer to be interrupted by the zero-marker. The interrupting `occam` program could then calculate the absolute position of the rotor. The **QValid** line of each link adapter is connected to their respective **QAck** lines via two TTL inverter gates in series. As with the PWM generator circuit, the propagation delays introduced by the two gates provides handshaking automatically.

7.3.2 Timing Considerations

The manufacturer's specifications of read cycle timings for the 8254 timer are more critical than those for the write cycles and the timing of the read cycle are therefore worthy of description. In the speed measurement circuit, the timer is initially programmed in the counting mode (mode 0) by the transputer 'writing' to the data bus. The write cycle is similar to that for the PWM generator circuit. During counting, the count values of the 8254 counter are read by the transputer for data logging. The timing diagram of the read cycle is shown in Fig.7.6. A time delay of at least 45 ns is required between $\overline{CS}$ and $\overline{RD}$ of

the 8254 timer. Furthermore, a delay of 5 ns is also required **DATA** and **IValid** lines by the IMSC011 link adapter chip. However, the maximum data line delay from the timer is 120 ns. Thus, allowing for a delay time of 20 ns from the D-type latch, and a safety margin of 5 ns, a delay of 150 ns is required between the $\overline{\mathbf{CS}}$ and **IValid** lines. The delays were realised by using simple RC circuits and Schmitt trigger inverter logic gates as shown in the figure.

7.3.3 occam Statements for Speed Measurement Circuit

A 5 ms sampling time is found to be commendable with induction motors having a rating of about 2 kW [Leonhard,1985]. For an induction motor with a rated speed of 1450 rev/min and for a digital encoder of resolution 2048 pulse/rev, an 8-bit counter is sufficient to measure the speed range from about 6 rev/min to 1490 rev/min. This is because:

$$2048\ (pulse/rev)*0.05(sec)*n(rev/sec) = an\ 8\text{-}bit\ word\ (\text{from } 0\text{-}255)$$

Since the range of an 8-bit word is 0 to 255, the range of the speed, n_{range}, that can be measured will be from 0 and 1494 rpm, which is determined as follows:

$$n_{range} = \frac{255}{2048\ (pulse/rev)*0.05(sec)}*60(sec/min) = 1494\ rpm$$

The resolution of the speed, n_{res}, is determined by:

$$n_{res} = \frac{1}{2048\ (pulse/rev)*0.05(sec)}*60(sec/min) = 5.85\ rpm$$

This resolution is considered to be moderate for high-performance drive system. Thus, a higher encoder resolution may be required if higher accuracy of speed is required.

With reference to the diagram of the speed measurement circuit, the format of the word to control the 8254 counter is as follows:

—	—	—	CS	RD	WR	A1	A0

Table 7.2 Word format for strobing the 8254 counter

As can be seen, the 3 MSB's are not used, and their values do not affect the addressing process. The binary and decimal values for the address of the timers are summarized in Table 7.3. Once again, due to the un-used bits, the values of the bytes for addressing the counter are not unique.

address/data (byte)	byte in binary	byte in decimal
mode 0-counter 0	00010000	16
control word for write	00001011	11
read counter 0	00000100	4

Table 7.3 Address/data for the 8254 counter

The **occam** statements to program the 8254 timer with an 8-bit word, and in event counting mode (mode 0) are given by the following:

```
SEQ
  to.timer.data.chan ! BYTE 16
  to.timer.control.chan ! BYTE 255 -- reset all signal
  to.timer.control.chan ! BYTE 11
```

To read the value from the counter and store it in the memory location '**speed.pulse**', the **occam** statements are required:

```
SEQ
  to.timer.control.chan ! BYTE 4
  from.timer.data.chan ? speed.pulse
  to.timer.control.chan ! BYTE 0 -- reset timer for next count
```

7.4 'control' PROCESS

The control process developed in the previous chapter was used for passing sample data and was mapped onto processor **T0**. The 'unused' processing power of **T0** could therefore be used to implement the vector control strategy. The control strategy used was similar to

the V-type method developed by [Hindmarsh,1985]. An off-line run of the algorithms for 'vector control' demonstrated the time required to run the basic algorithm of vector control was 59 μs. This time is well within the limit of the sampling time (5.12ms).

7.5 'current' PROCESS

The '**current**' process is responsible for the measurement of the motor current(s). With the present system, only one motor line current was monitored. The software aspect of the current process was rather simple and was facilitated by an A-to-D converter card (by Bangor University). The functional diagram of the A-to-D converter is shown in Fig.7.7. It mainly consists of a multiplexer, a sample and hold device and the standard link adapter. Since only one phase of the current has been monitored at this stage, the seven inputs are reserved for further development.

7.6 MAPPING OF PROCESSES TO PROCESSORS

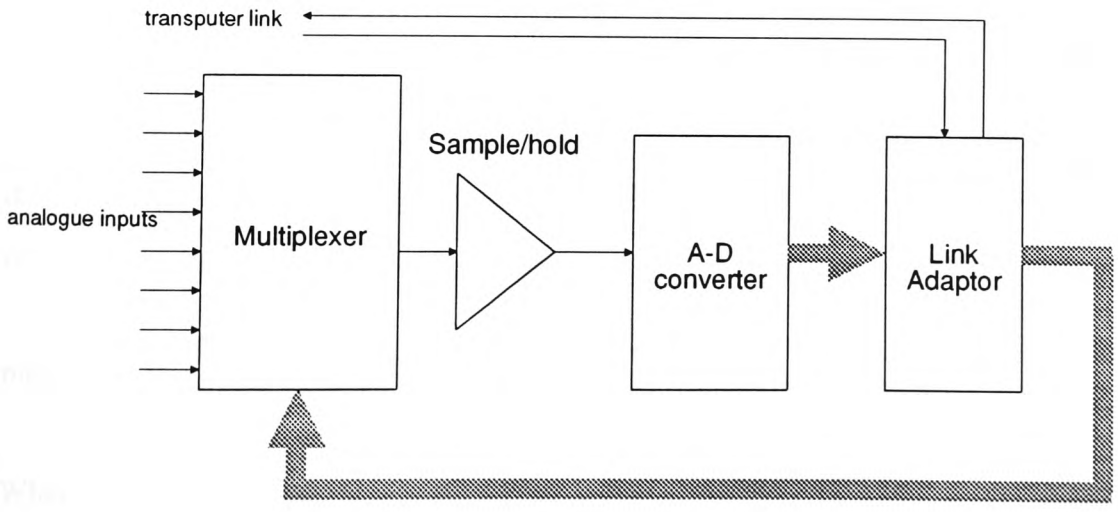


Fig.7.7 Functional diagram of A-to-D converter

The mapping of processes (tasks) to processors is briefly mentioned in chapter 6, where the process '**control**' is mapped to processor **T0** and the process '**pwm**' is mapped to processor **T1**. The task performed by process '**control**' in that case consisted of nothing no more than passing data. In the previous sections of this chapter, the four processes have been developed:

(1) The '**pwm**' process which includes the soft-start process.

(2) The '**control**' process that performs the vector control strategy and data traffic management.

(3) The '**speed**' process that is involved in the measurement of the speed of the motor by means of the encoder.

(4) The '**current**' process whose duty is to sample and process the motor current.

In addition, the '**monitor**' process has been upgraded to include the input of more data, such as that required for soft-starting. Although there are several floating point calculations in the processes '**speed**' and '**current**', it only takes up a small fraction of the computing power available when they are each mapped onto an individual processor. However, the '**speed**' process requires two links to communicate with the external counter, whilst the current process requires one for the control of the A-to-D converter. Since each transputer of the B003 board has already two links committed for inter-processor connections, only two links are available for each transputer for external communications. A direct solution is to run the '**speed**' and '**current**' processes on the two remaining transputers on B003 board, **T2** and **T3**. Thus, a large amount of computing power in each processor is reserved for further development. The topology of the transputer network and the host transputer is given in Fig.7.8, and the corresponding mapping table is shown in Table 7.4. It may be noted that not every transputer link is mapped with an **occam** channel. The program development compiler gives a warning message for any uncommitted link, which may therefore be ignored.

When a transputer network is loaded from a host computer, the code for each processor must be directed to the appropriate processor before start up of the program. The route to a processor in a network is through other processors. Thus, each processor should be able to pass on the code of other processors, before starting to run its own code. At first, the

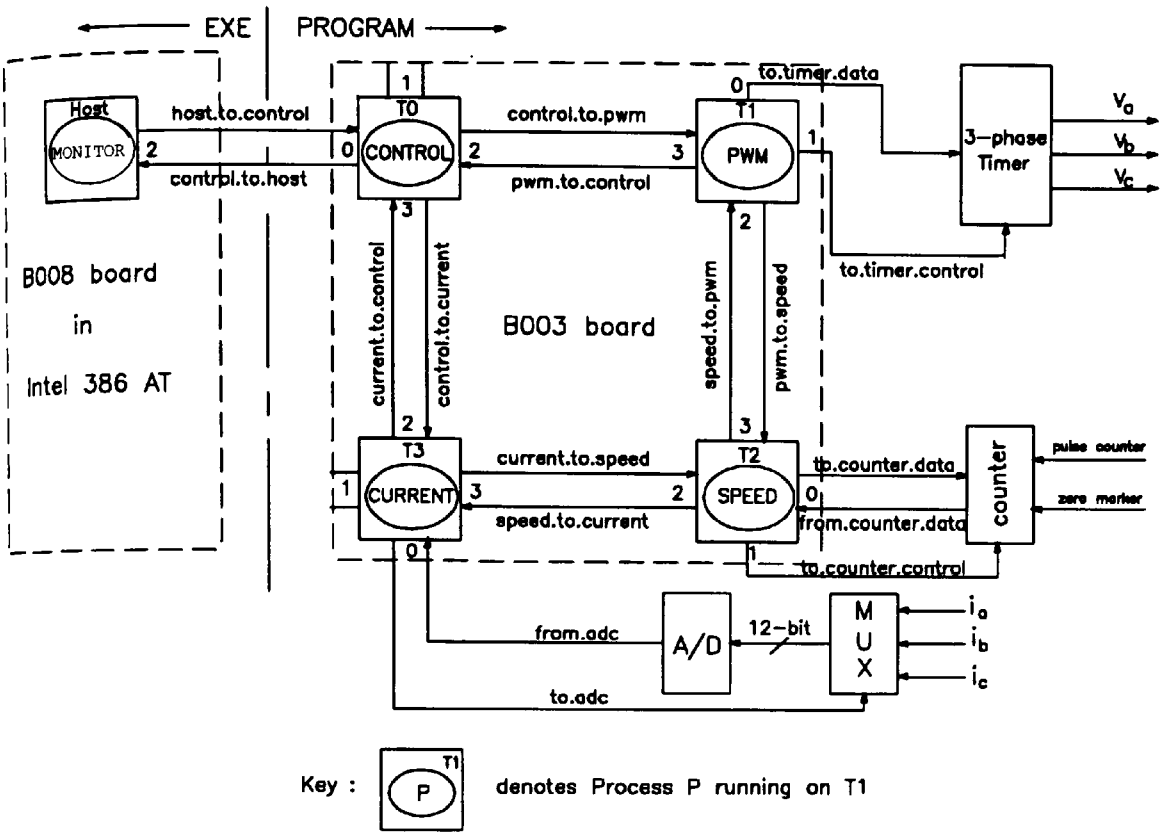


Fig.7.8 Topology of 5-transputer network

Processor	Channel Name	Link Number	Direction
0	control.to.host	0	out
	host.to.control	0	in
	control.to.pwm	2	out
	pwm.to.control	2	in
	control.to.current	3	out
1	pwm.to.control	3	out
	control.to.pwm	3	in
	pwm.to.speed	2	out
	speed.to.pwm	2	in
	to.timer.data	0	out
	to.timer.control	1	out
2	speed.to.pwm	3	out
	pwm.to.speed	3	in
	speed.to.current	2	out
	current.to.speed	2	in
	to.counter.data	0	out
	to.counter.control	1	out
3	current.to.speed	3	out
	speed.to.current	3	in
	current.to.control	2	out
	control.to.current	2	in
	to.adc	0	out
	from.adc	0	in
	to.dac	1	out

Table 7.4 Mapping table of the transputers in the network of transputers described in Fig.7.8

to pass on the code of other processors, before starting to run its own code. At first, the TDS sends a 'reset' signal to all transputers so that all are ready to 'boot from links'. The TDS then sends a packet of bootstrap data to the root transputer `T0`, and then boots and starts executing the packet. The bootstrap code initializes the transputer, and then receives a sequencing code called 'distributing loader.' The loading program loads the application code onto the root transputer, and routes the code to other processors in the network. Once the neighbouring transputers have been booted, they also contain the 'distributing loader', and can in turn route code packets to their neighbours. This continues until the whole network has been booted. The configuration description gives the information of the topology of the network. After all the transputers in the network are booted, the code packets for procedures allocated to processors in the configuration description, are exported to the network, preceded by necessary routine and loading information. Finally, the code for each processor is sent to the appropriate processor, and then each processor is ordered to start executing the loaded program.

7.7 EXPERIMENTAL RESULT

7.7.1 Experimental Setup

The experimental results were carried at light load conditions, where the load was provided by a Jay-Jay dynamometer. The DC link voltage to the inverter was 60V and DC link output current was limited to 5A. Such measures were imposed to prevent the transputer system from experiencing interference from the switching devices of the inverter. The level of interference, from both the machine and the inverter, was found to increase with the motor load current and the DC link voltage of the inverter. As the transputer is designed to operate in an electrically 'quiet' environment, it is very important to monitor the level of the radio frequency interference (RFI) commonly occurring in motor drive systems. It had been observed that, during a power off operation in which the motor was running down to standstill, communication failures between the transputers occurred when program codes were loaded to the transputer network from the host transputer in order to run the program again. The failures were believed mainly due to the size of the program codes (several Kbytes) to be downloaded to the network,

because the possibility of incorrect communication increased with the duration of the communication. The communication of the data and control signals (several bytes) from the transputer for the generation of PWM, however, suffered no communication failure problems during the operation of the drive. The experimental setup is shown in Fig.7.9.

7.7.2 Soft-start

The 'softness' of the starting process can be easily controlled in the present system by using different values of the initial modulation index, M_0, for startup. In the soft-starting process shown in Fig.7.10, a voltage boost of 'M_0=0.05 ' at a DC link supply of 60V was used. The trajectory of the motor speed is shown to be very different from that of direct-on-line starting. The motor ramps slowly during the first 2 seconds when the supply voltage is used to overcome the voltage drop across the stator winding. Fig.7.11 shows the response of the motor for an increase of voltage boost where 'M_0=0.1'. With an increase of voltage boost, the voltage drop across the stator winding is compensated for during low voltage operation and the motor steadily ramps to the reference speed from standstill. Three oscillographic pictures of the motor line current during a soft-start are shown in Fig.7.12. It may be seen that the magnitude of the current stays roughly constant throughout the acceleration period when the *v-to-f* ratio is maintained constant. The transputer computes and outputs the switching angles of the PWM waveform during each frequency incremental interval (0.25 seconds) by maintaining the *M-to-f* ratio constant during the startup period. The trajectory of the voltage phasor during the same soft-start process is shown in Fig.7.13. In this figure, the 'voltage boost' can easily be identified by the sudden 'jump' of the phasor just after start-up. Thus, it is apparent that the inclusion of the soft-start process reduces the harmful effect of the inrush current to the motor.

7.7.3 Dynamic Braking Mode

In dynamic braking, the AC motor is disconnected from the AC supply and connected to a DC supply. The way in which the induction motor can be connected to the DC supply in an inverter bridge system can easily be achieved by switching certain power switches on,

as shown in Fig.7.14, for example. It should be noted that other connections, as also shown in Fig.7.14, may give a uniform flow of current in all three phases but complicate the switching operation. The flow of direct current through the stator windings establishes a stationary magnetic field in the motor. Thus, the relative speed between the stationary magnetic and the rotating rotor becomes negative. The rotor induced voltages are therefore of reverse phase sequence such that the resultant three-phase rotor currents produce a rotating field that is moving with the rotor speed in the direction opposite to that of the rotor. A stationary rotor field is therefore set up. Since both the stator and rotor fields are stationary and the rotor flows in reverse direction, a steady negative torque is produced at all speeds. It becomes, however, zero at standstill due to zero rotor currents. During the braking process, the DC link current flowing through the stator windings was limited by the current limit (5A) set by the supply. If such a limit was not imposed, the magnitude of the current would have been determined by the supply voltage and the stator winding resistance. In practice, however, it may be necessary to avoid high DC currents flowing into the stator windings by lowering the supply voltage, or applying a 'chopper' signal to monitor the DC stator currents. The trajectory of the motor speed during a dynamic braking process when an initial speed of 600 rpm at light load is shown in Fig.7.15. A comparison of the motor speed at the same initial speed and load condition with no dynamic braking, that is to say, the supply to the motor was switched off by the turning off of all the power devices in the inverter, is illustrated in Fig.7.16. It may be seen by comparing Fig.7.15 and Fig.7.16 that the time required for the motor to reach to standstill is decreased by a factor of approximately 10 when dynamic braking is applied. Since only AC current could be measured by the current transformer used in the test system, the magnitude of the DC current (5A) during dynamic braking period appeared as zero on the oscilloscope. The occam sub-program responsible for dynamic braking monitors the speed of the motor during the dynamic braking period and causes the upper switching devices of the inverter to turn off when the motor speed falls below 5.85 rev/min (see section 7.3). In fact, in Fig.7.15, a transient can be identified in the current waveform at an instant just before the standstill condition, when a sudden change of current was caused by the removal of the stationary field set up by the stator winding during the dynamic braking period.

7.7.4 Vector Control

An on-line keystroke was provided to enable/disable the 'vector control' algorithm so that comparisons of the system performance with vector control 'enabled' and 'disabled' could be readily made. The effect on the motor speed transients was studied in detail during an 'excursion' from one reference to another reference speed. Typical response curves for motor speed and motor line current are illustrated in Fig.7.17 and Fig.7.18. It may be seen from Fig.7.17 that when system is programmed for vector control and an increase of speed of 240 rpm is demanded, then the new value of speed is achieved in approximately 0.3 s. It may also be seen that during the period of changing speed, large transients occur in the motor line current. Similarly it may be seen from Fig.7.18 that when the same speed increase in speed is demanded and vector control has not been elected, then the motor speed takes approximately 0.8 s to reach the new desired value. The transient in the motor line current, however, is much smaller than those observed for 'vector control'. Although the results show an improvement by a factor of over two when 'vector control' is applied, a smaller response time may be achieved if the power source can deliver the actual current and voltage demanded by the 'vector control' algorithm. The test results for a step decrease of 120 rpm with 'vector control' enabled and disabled are shown in Fig.7.19 and Fig.7.20 respectively. With 'vector control' enabled, the motor speed response time is shorter and less overshooting when compared with the result obtained when 'vector control' is not implemented. However, it must be stressed that when a decrease in the speed is demanded, the power converter supplying the motor must be able to absorb the kinetic energy of the rotating masses (rotor and load) during the deceleration period. Consequently, the effectiveness of the vector control scheme in a motor drive during a deceleration is greatly affected by the ability of the power source to absorb regenerative power, or dissipate the power in form of I^2R loss.

7.8 INTERIM CONCLUSION

The test results for soft-starting and dynamic braking confirmed the theoretical results described both in chapter 2 and 3 in this thesis. The experimental result of vector control also agreed with simulated results presented and described in chapter 3 and 4. However, the experimental result achieved were rather limited because of the interference experienced by the transputer network as a result of the switching operations of the power devices in the drive system. It should be stressed that a more receptive DC link power supply, or an efficient means of removing the regenerative power, will be required for further study of vector control during motor deceleration.

The experimental results obtained also demonstrated that the transputer used could accommodate all of the initial specifications and requirements of the drive system without having to resort to 'load balancing' of the transputers in the network. However, it is believed that 'load balancing', in which each processor shares approximately the same load in a network of processors, will become necessary when further control functions such as adaptive control are included in the present drive system.

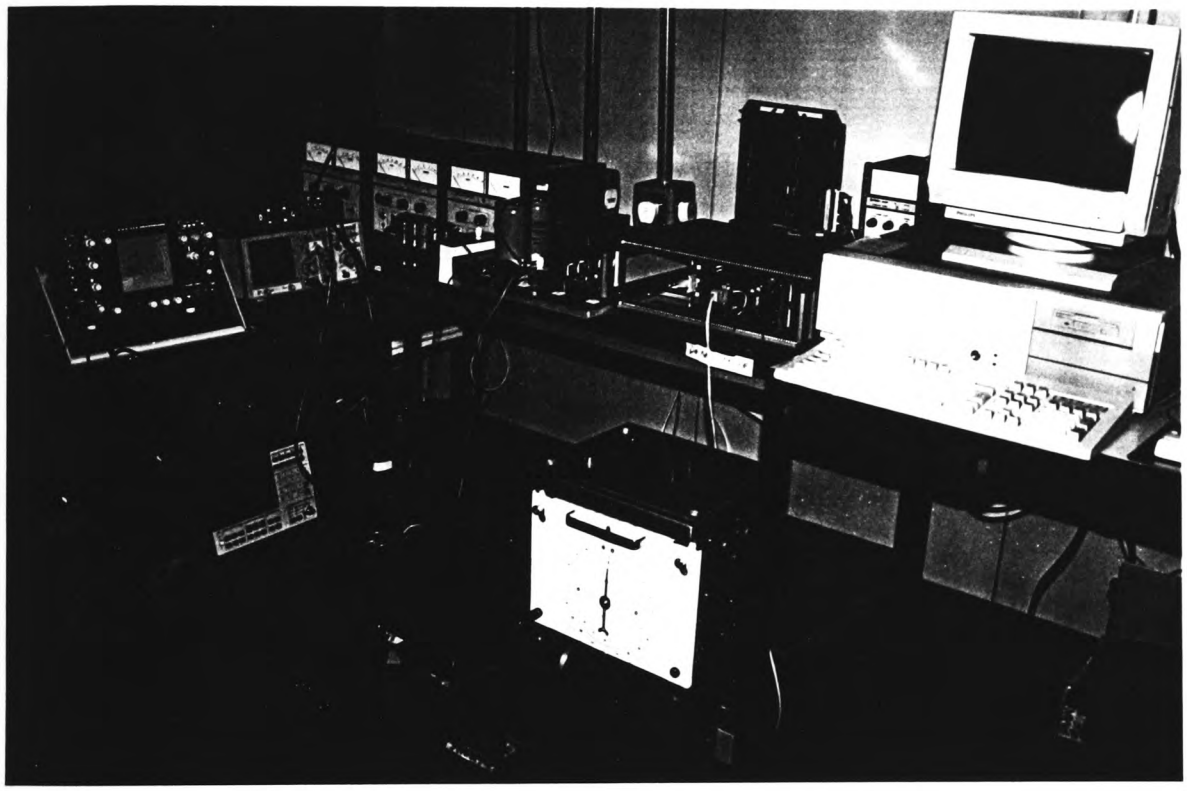
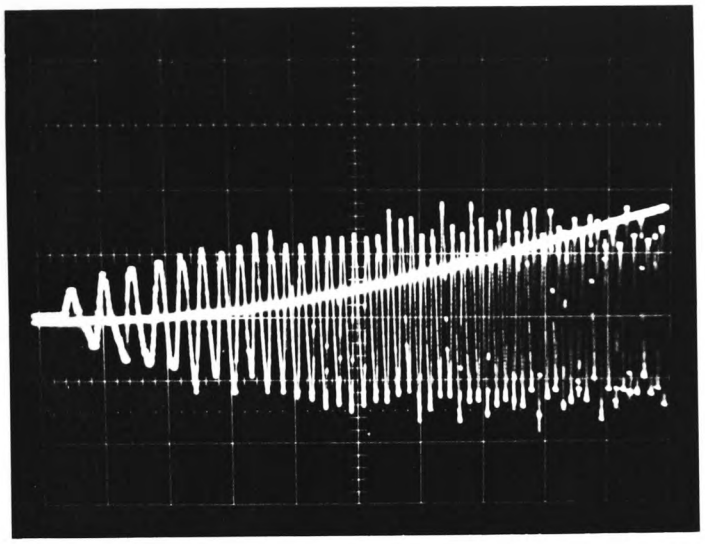


Fig.7.9 Experimental setup of the complete drive system

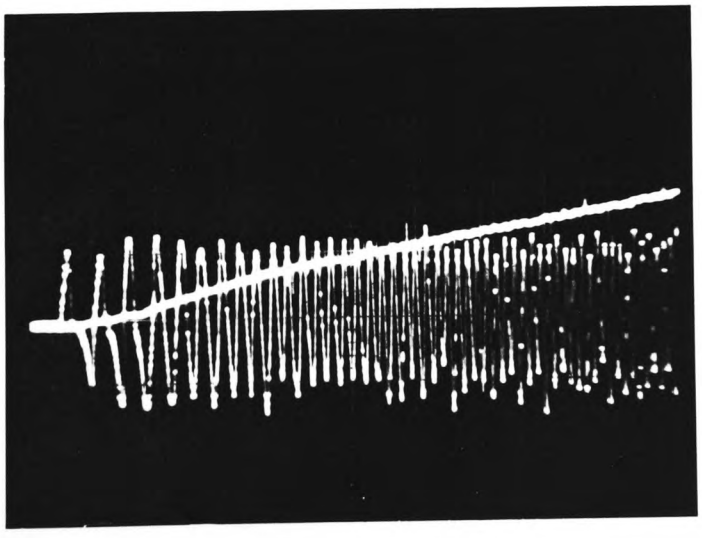


Speed (60rpm/div)

Current (0.1A/div)

Time base (0.5s/div)

Fig.7.10 Trajactories of motor speed/current during a soft-starting with voltage boost M_0 =0.05



Speed (60rpm/div)

Current (0.1A/div)

Time base (0.5s/div)

Fig.7.11 Trajactories of motor speed/current during a soft-starting with voltage boost M_0 =0.1

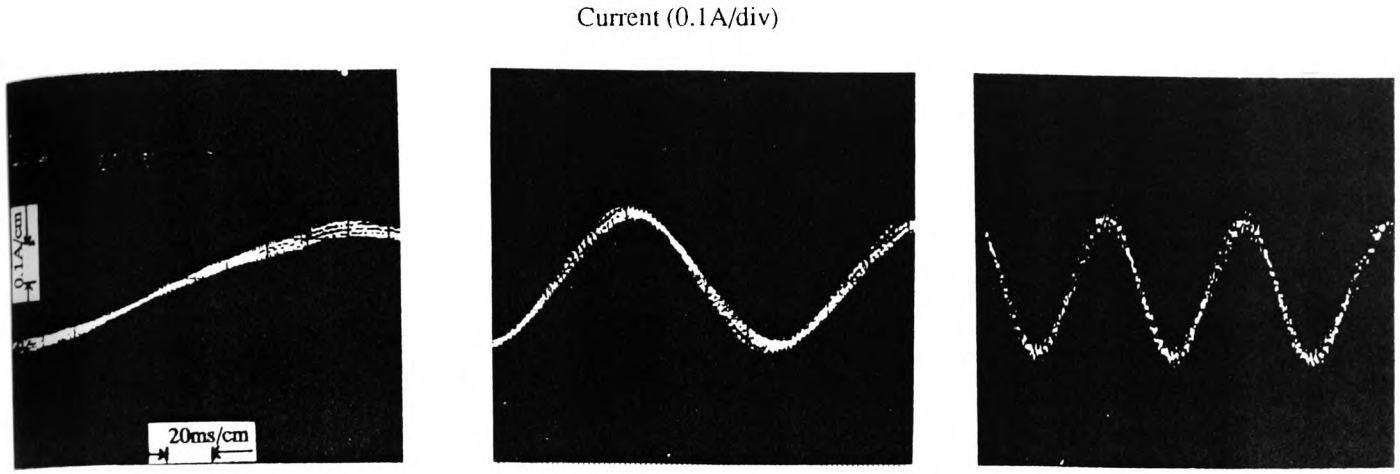


Fig.7.12 Motor line currents in soft-start process with voltage boost M_0 =0.05, modulating index at full speed=0.95

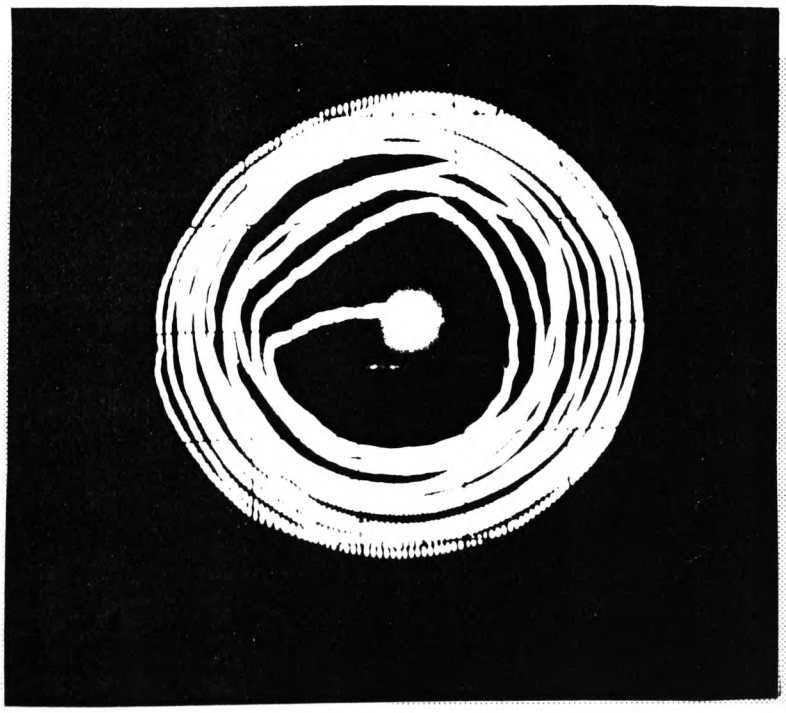


Fig.7.13 Trajactory of voltage phasor in soft-starting 'voltage boost' of M_0 =0.1

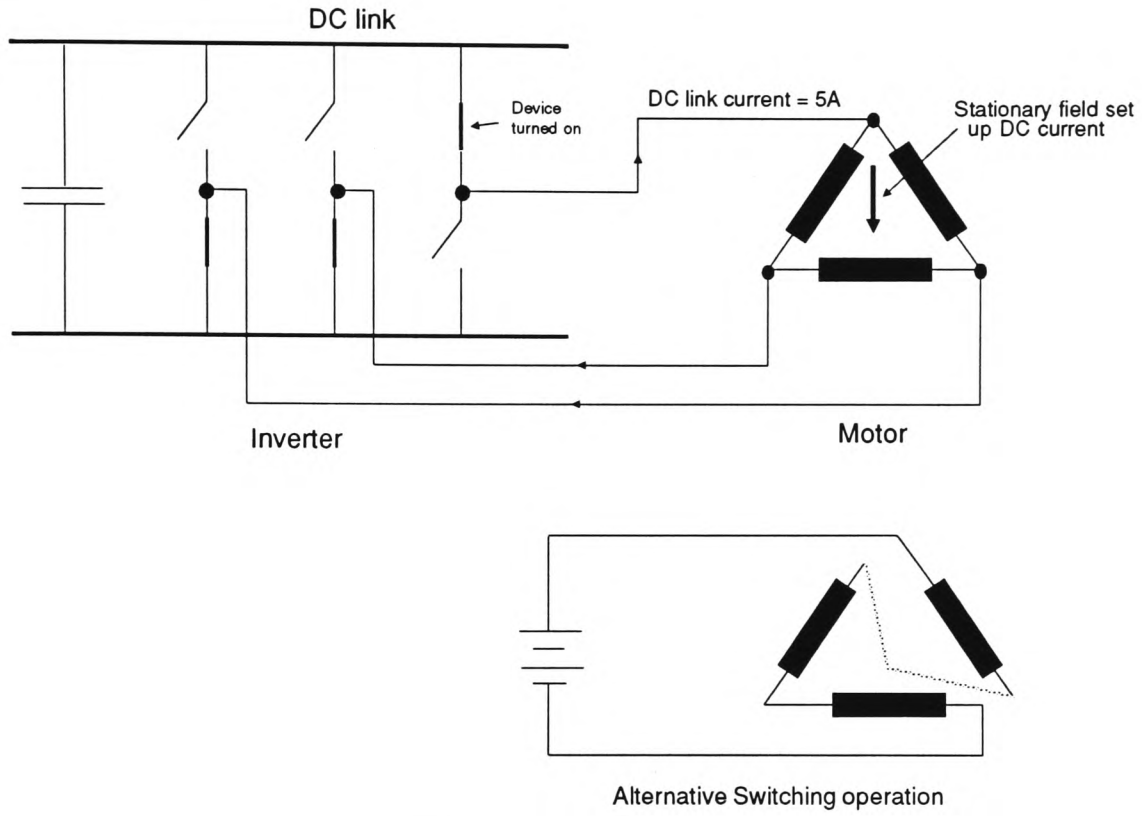
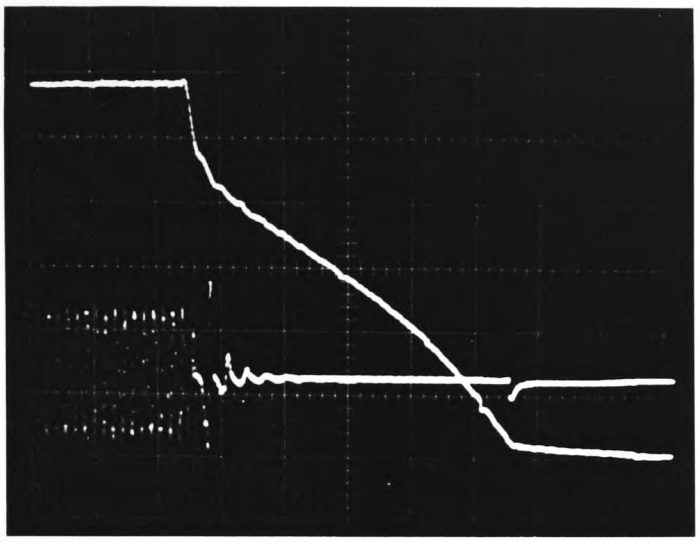


Fig.7.14 Dynamic braking process

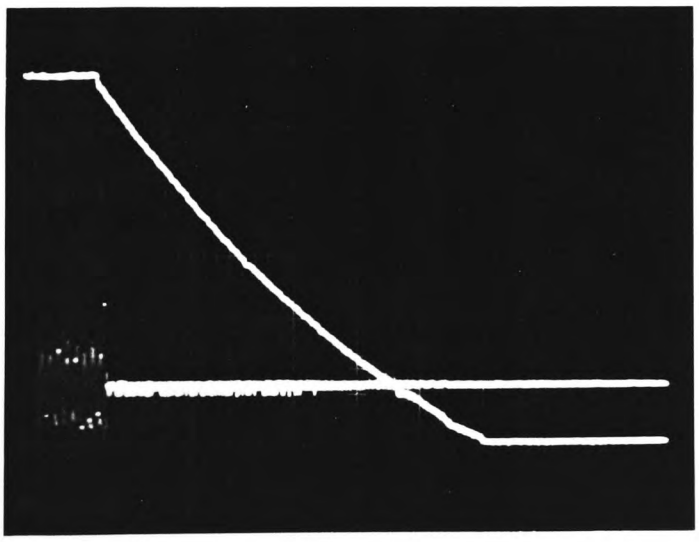


Current (0.1A/div)

Speed (100rpm/div)

Time base (0.5s/div)

Fig.7.15 Trajactories of motor speed/current during dynamic braking from an initial speed of 600 rpm

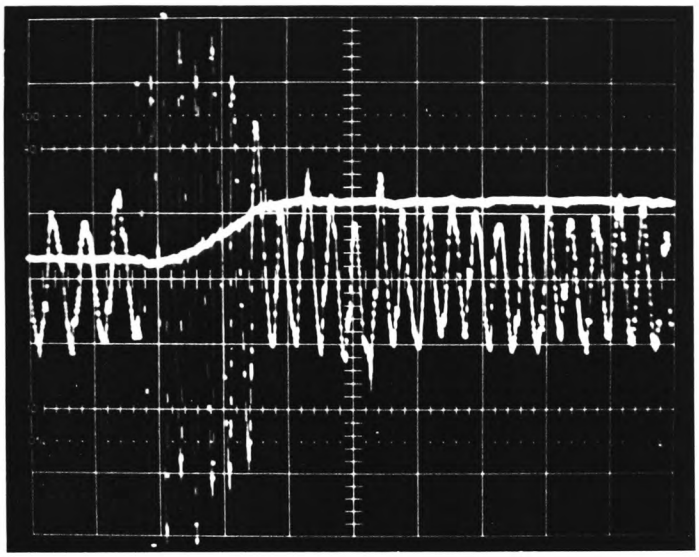


Current (0.1A/div)

Speed (100rpm/div)

Time base (5s/div) *10 times that of Fig.7.15

Fig.7.16 Trajactories of motor speed/current with power switches OFF from an initial speed of 600 rpm

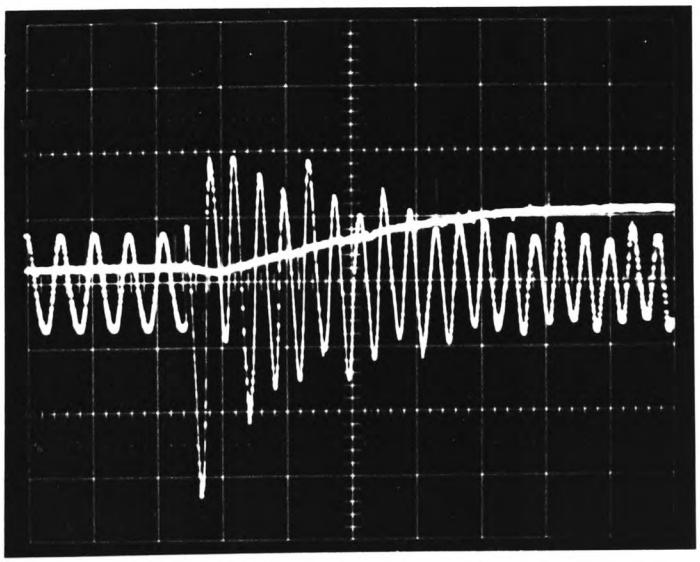


Speed (240rpm/div)

Current (0.1A/div)

Time base (0.2s/div)

Fig.7.17 Dynamic response with vector control for a step increase of 240 rpm in reference speed

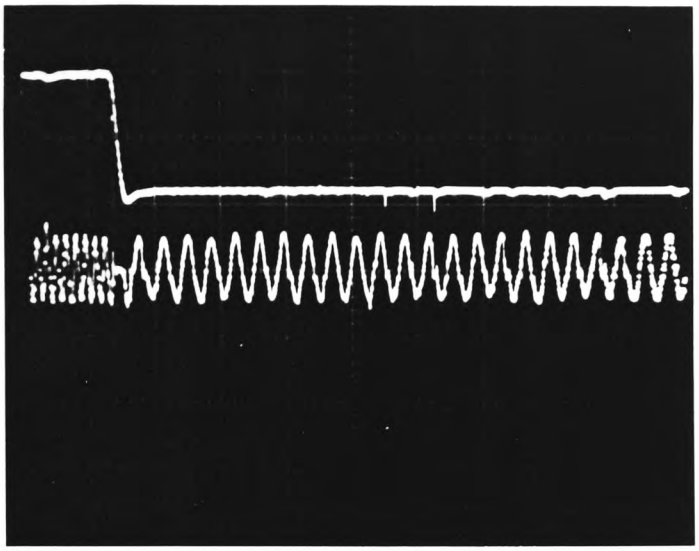


Speed (240rpm/div)

Current (0.1A/div)

Time base (0.2s/div)

Fig.7.18 Dynamic response without vector control for a step increase of 240 rpm in reference speed

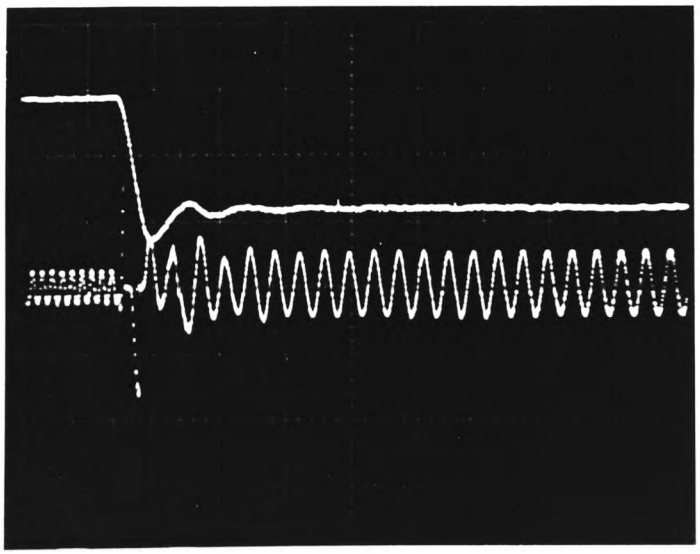


Speed (60rpm/div)

Current (0.1A/div)

Time base (0.2s/div)

Fig.7.19 Dynamic response with vector control for a step decrease of 120 rpm in reference speed



Speed (60rpm/div)

Current (0.1A/div)

Time base (0.2s/div)

Fig.7.20 Dynamic response without vector control for a step decrease of 120 rpm in reference speed

Chapter 8 : Conclusion and Further Research

8.1 AUTHOR'S CONTRIBUTION

This thesis contributes to the state-of-the-art technology of microelectronics in the control of induction motor drives as a whole, the application of parallel processing to induction motor drives by means of a transputer-based system specifically. Furthermore, the digital simulation of induction motor drives reported is thought to be amongst the first of such implementation which uses the state-space analysis technique by means of a personal computer (PC). The similar simulation investigations reported elsewhere have either been done on mini- or main-frame computers using the state-space method, or on PCs using methods that did not employ state-space analysis.

Conventional variable speed induction motor drives of similar complexity and functionality to the system described in this thesis, when implemented by a single sequential processor or a network of processors, would require the processor with very high processing speed. As the cost of a processor increases dramatically with increasing processing speed, such a single processor system becomes very costly to develop. Moreover, if such a system is developed, it would still lack the flexibility necessary for further upgrading or development of the control system. It should also be emphasized that the bus-based multiprocessor system is difficult to design and costly to develop mainly because of the problem of 'bus contention', that is to say, where two or more processors 'talk' to the bus system at the same time.

The transputer-based system reported in this thesis is superior to any sequential processor-based system for the following number of reasons. The use of the standard serial transputer link with its facility to support parallel processing has made the communications among processors and between peripherals a trivial task, when compared with a conventional multiprocessor system. Moreover, the cost of the development system's hardware, and the programming and software development system (TDS) for the transputer-based system, is low (£2000), when compared to the conventional Intel 16-bit 8096 development system (£8000), for example. The use of the occam language also greatly reduced the required programming time dramatically, as

well as guaranteeing no loss of program coding efficiency. This resulted in a considerable reduction of software development cost.

The real-time generation of regular asymmetric sampled PWM waveforms by means of the transputer internal timer and logic gates can also claim to be novel. PWM waveforms of a theoretical carrier frequency of approaching 2 MHz is realizable if an improved hardware circuit is developed and memory-mapping method (instead of the serial link method) is used to address and load the external timer circuit. Although the '4-timer method' of PWM generation is not new, its implementation of using the internal transputer hardware clock and the associated occam program is thought to be novel.

The literature suggests that the simulation of motor drive systems have been mainly realised by means of mini or main-frame computers. Such simulation studies using a personal computer is relative recent to date. Furthermore, the use of simulation techniques to develop a model drive system by means of a general purpose control software package, MATLAB, which is in fact becoming an industrial standard, is thought to be the right direction to be moving so far as the CAD design of the future 'integrated drive system', as proposed by Bose [Bose,1989], is concerned. It is noteworthy that because modern control theory trends tend to favour the use of the state-space approach, the space-state model of the drive system reported in this thesis can be easily integrated into the modern control design philosophy for digital controllers of variable speed induction motor drives.

8.2 CONCLUDING REMARKS

The initial objective of investigating the means by which the 'parallalism' inherent in the transputer and the high processing speed of the processor of the transputer that could be usefully applied to the real-time control of VVVF induction motor drives employing a number of control strategies, was achieved. For example, it has been recognised for some time that a three-phase induction motor drive possesses a considerable degree of 'parallelism', which can be exploited by a control system employing parallel processing.

The real-time control of open-loop PWM induction motor speed control was implemented by means of a novel architectural approach which proved successful, and offered a high degree of flexibility. Similarly, the known advantages of using 'vector control' to improve the dynamic response of induction motors was realised by means of a novel multi-transputer system approach. The flexibility of the approach is considerable and, with little modification to be made to include other control methods such as the adaptive control and sliding mode control. One area of the investigation, however, which presented considerable problem and was not entirely resolved, was that of RFI interference experienced by the transputers in the drive system.

The software development time of parallel processing by **occam** was considered to be relatively short when compared to programming languages such as *Ada* or *Modulus*. Moreover, **occam** is a high level language and supports the use of floating point data type operations (REAL32) with the guarantee of compilation efficiency similar to assembly programming. The software maintenance of the system developed was also greatly simplified under the integrated environment supported by the transputer development system. Therefore, it was these advantages of the transputer that had been exploited in the development of a parallel-task oriented induction motor drive system with and without vector control algorithms.

The transputer and its language **occam** present a very 'user-friendly' development environment for the induction motor drive. The development time was remarkably short when compared with conventional development methods. The use of the high level programming language **occam** resulted in program code with very respectable efficiency comparable to assembler programming.

The validity of the dynamic and steady state simulation model developed for the induction motor drive system was confirmed by the result of experimental tests. The detailed modelling of the stator flux in a vector control drive supplied with a PWM power supply, however, is extremely complex and beyond the scope of this project. The simulation results obtained demonstrated that the dynamic response of the motor could be greatly influenced by the power handling capability of the DC link.

Therefore, the many advantages of the drive system developed and reported in the thesis can be summarized as follows. The novel transputer-based system developed is characterized by its flexibility, expandability and high computing power. The experimental test rig developed satisfied the requirement of flexibility that allows the implementation of other control strategies such as adaptive control and sliding mode control without incurring considerable modification in the system hardware. The on-line interaction environment of the drive system enables the user to implement a number of control PWM parameters even though the drive system is operating under some previous instructions. Finally, software development times are made short due to advantages of the high level language **occam** and the use of floating point (REAL32) operation.

8.3 RECOMMENDATION FOR FURTHER RESEARCH

8.3.1 Overall Improvement

The present system allowed operations of the drive under light or no-load conditions. A full implementation of the vector control under full load was not possible for several reasons. The output impedance and current limit of the DC link supply used affected the response time and the amount of current required by the control algorithm to drive the motor during its full load transient response. The shielding required to prevent electromagnetic interference of the present system could be significantly improved by housing the whole transputer network in a fully shielded enclosure. This would allow the drive to be tested at loads nearer to its full rated values.

Other design criteria such as space factor, maintenance, instrumentation, fault protection and modular structure would require to be implemented before the present system could be considered a pre-production prototype. It is also suggested that [Plunkett,1985] the facilities for on-line diagnosis and evaluation of the efficiency of the drive are highly desirable. With the ample processing power available for further use, such facilities can readily be realised with the drive system developed.

8.3.2 Noise Problem

The architecture of the transputer is such that it is mainly intended to be operated in 'electrically quiet' environment. The B003 board used in the development of the drive system was an evaluation board in which the four on board T414 transputers are mounted in such as a way that the board is more for demonstration purpose than for industrial applications (see Fig.7.9). Whilst such setup may be ideal for evaluation of parallel processing where the four processors can vividly be visualized as processing in a parallel fashion, it is in fact greatly exposed to electromagnetic interferences.

Since radio frequency interference (RFI) is inevitably generated in motor drive systems, the interference must be properly monitored and suppressed if a true industrially viable drive is to design. There are many different methods for reducing the effect of RFI. The most obvious method that may be applied to the experimental drive system developed would be to house the whole drive system in a shield case, and to separate the control circuitries and boards from the motor and power converter. The implementation of such a method, however, would require the use of long and reliable transputer links for the data communication. It is believed the development of optical fibre transputer links reported by some transputer users may further facilitate the use of transputers in such industrial applications as control of electrical drives.

During the development of the project, it was found that RFI from motor or from the fluorescent light system in the laboratory would cause malfunction of the transputer system. The suppressing of this effect in a high performance drive system, which could consist of a number of processors running in parallel as well as communicating with other in its control circuit, presents a great challenge to the researcher and design engineer of electrical drives.

8.3.3 Adaptive Control

Although the relatively low speed of serial communication between the transputer and the external 8254 timer of the present system is a rather limiting feature, there is, however, a

very high reserve of computing power which could be readily exploited for the incorporation of adaptive control schemes which would further enhance the performance of the drive system. It is apparent from the algorithm of the vector control law employed that the validity of the control law depends on the accuracy of the rotor resistance value during the operation of the drive system. For example, the value of the rotor resistance may vary 50 percents under condition of prolonged operation. This variation in rotor resistance, unless compensated for, would affect the validity of the vector control algorithm, which relies on true actual values of rotor resistance. The application of adaptive control to the vector control system could identify and compensate for the variation in rotor resistance value under varying load or temperature conditions by developing an on-line and accurate model of the drive system. Such an on-line method of deducing the internal state of the plant or the system under control, commonly known as 'state observer', is illustrated in Fig.8.1 [Acarnley,1991]. The model of the plant receives the same input as the plant, and the outputs are then compared. The modelling errors are then fed back to the model for corrections of the model parameters. Thus, by keeping the modelling error within a predetermined limit, the model states can be expected to reflect the actual states of the plant.

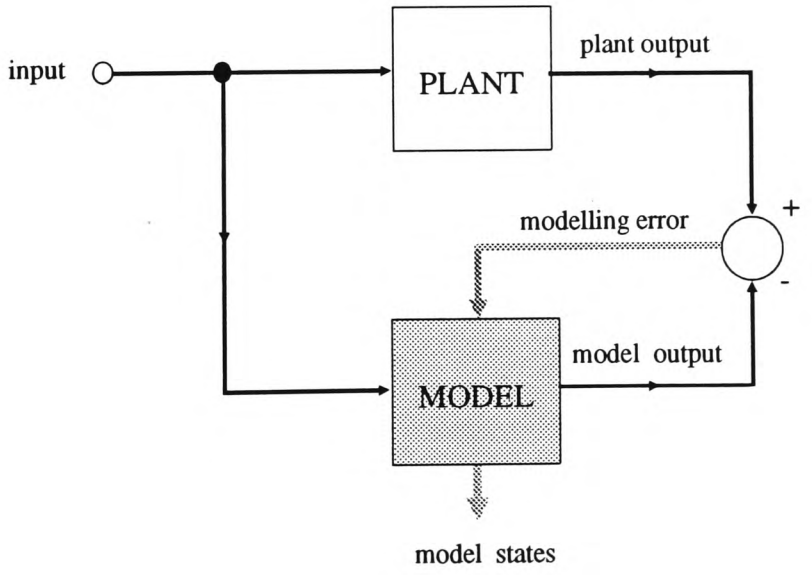


Fig.8.1 Application of adaptive control by 'State observer'

8.3.4 DC Link Supply

The output characteristic of the DC link supply can greatly influence the performance of vector control in drive systems. Since vector control aims to provide fast response time for the drive system, the transient response of the DC link power supply directly affects the performance of the drive system. Due to the output power capability of the Farnell DC power supply units, which were used for the DC link in the project described, they were only suitable for drive systems under light load or no load condition. It was found that higher loads incurred unacceptable 'dip' in the output voltage of the DC link supply, which impaired the performance of the vector control drive system. This was because the vector control strategy was designed for a fixed DC link voltage. Thus, any change of voltage, if not allowed for, affected the validity of the control scheme. A robust and stable DC link power supply capable of delivering and sourcing the transient 'burst' of power required by the vector control strategy in a command of a step change in speed, for instance, is therefore highly desirable.

8.3.5 Higher Carrier Frequency

The theoretical upper limit value of the carrier frequency for the PWM waveforms generated in the present method was 2 MHz. However, to realize a practical value near to the theoretical value would require a much quicker means of addressing the external timer. This requirement could possibly be partially met by the next generation of transputers, the T9000, which has a communication speed of ten times higher than that of the T800 transputer, and would therefore provide a ten-fold increase in the practical value of the carrier frequency. Another, and possibly better solution to the problem, is to use the 'memory-mapping' addressing technique commonly found in conventional sequential microprocessor systems. Since the normal 'off the shelf' transputer boards do not provide a direct access to the address bus of the transputer, the design of a custom decoding circuit is necessary. However, it has also been argued that, in contrast to the aspiration of generating PWM waveforms by high speed processor in real-time, the task of PWM generation should be relegated to a hardware circuitry where supervision of the host computer required is minimum and only very simple commands such as a change of modulating frequency may be allowed.

Appendices

Appendix A : An introduction to the transputer architecture and the **occam** programming language

Appendix B : MATLAB program listings

Appendix C : **occam** program listings

Appendix D : Solution to across the limit barrier of size of vector imposed by AT-MATLAB

Appendix E : Eurocard rack system

Appendix F : Motor's equivalent circuit and data

Appendix G : Circuit diagrams

Appendix H : Experimental setup for the results of Chapter 4

Appendix I : References

Appendix J : Published papers relating to this thesis

(a) "Use of transputer for pulse-width modulated (PWM) inverters", Proceedings of the 24th University Power Engineers Conference, Belfast, 1989.

(b) "A digital model for a three-phase induction motor drive using a personal computer software package", 5th IFAC/IMACS Symposium on Computer Aided Design in Control Systems, Edited by H.A. Barker, Pergamon Press, July 1991.

(c) "The transputer control of inverter induction motor drives", Proceedings of 4th European Conference on Power Electronics, Florence, Italy, 3-6 September 1991.

Development, Revolution and Evolution of the Transputer

The **inmos** transputer, launched in 1985, represented a bold attempt to revolutionize the design of microprocessor-based systems. The first member of the family was the IMS T414 32-bit transputer which had an unprecedented speed of 10 Mips and 2 Kbytes of on-chip random-access memory (RAM). The T800, introduced in 1987, was a major step forward in transputer processing power. It is essentially a revamped T414, with extra instructions, 4 Kbytes of RAM and a 1.5 Mflops (million floating point operations per second) floating point processor. The next generation of transputers, called T9000 series (or T9), are due to be launched in late 1991. The T9 transputer will provide upward compatibility and an increase of an order-of-magnitude in performance over the first-generation TX-series transputers. There is also enhanced support for operating systems in embedded applications as well as enhanced support for multiprocessing via improved interprocessor communications.

The third generation transputer family, codenamed EX, which is already in the design stage at **inmos**' Bristol design centre, is intended to augment **inmos**' strength to embedded systems by providing a powerful general-purpose microprocessor. The EX transputer will represent a significant departure from the TX and the T9 series in its architecture with inclusion of multiple data paths. The transition from TX to T9, and the complementary development of the EX, can be seen as an inevitable process of evolution in which the transputer's unique communications capabilities are merged into the mainstream of microprocessor development, which may not be the revolutionary programme envisaged by the **inmos**' founders.

Basic Architecture and Concepts

The architecture of the transputer exploits the properties of VLSI technology, that, communication between devices is much slower than communication within a device. As nearly all computer operations involve the use of memory, the transputer includes both the processor and the memory in the same integrated circuit device. Since one of the main design goals of the transputer was its implementation in multiprocessor systems, there is no support for memory management or virtual memory. Each processor has its own local memory and the processors communicate by message-passing along the links. As each transputer has its own local memory, the memory bandwidth of the system increases proportionally with the number of the transputers used. The salient features and statistics of the transputer (T800 as a typical example) are summarized as following:

(a) each transputer is a single VLSI CMOS chip,
(b) on-chip memory is inherently fast static RAM,
(c) the 32-bit transputer gives access to 4 gigabytes of off-chip memory through a 32-bit wide multiplexed address/data bus,
(d) transputer will function at a speed between 5 and 10 MIPS (Millions Instructions Per Seconds); these instructions are reduced type instructions similar to those of RISC,
(e) the process cycle has a duration of 50 ns,
(f) a transputer has 4 links for connection with other transputers; each of which can transfer data at a rate of 10 or 20 Mbits per second,
(g) the transmission failure rate on a link due to sychronisation failure is less than 0.1 FIT (less than one failure in ten thousands million devices operating hours), and
(h) it has a built-in Floating point unit (FPU).

As a high degree of concurrency is always aimed at by **occam** and therefore the transputer, the method of communication is an important aspect. The transputer uses point-to-point communication, the advantages of which, when compared with multiprocessor bus systems, are:

(a) there is no bus contention,
(b) there is no capacitive load penalty as transputers are added to the system,
(c) the communication bandwidth does not saturate as the size of the system increases and,
(d) the transputer connections can be short and local, irrespective of the size of the system.

Simplified Processor With Microcoded Scheduler

The most effective implementation of simple programs by a programmable computer is provided by a sequential processor. Thus, the transputer has a fairly conventional microcoded processor. There is a small core of about 32 instructions which are used to implement simple sequential programs. There are also specialized groups of instructions which provide facilities such as long arithmetic and process scheduling. As a process executed by a transputer may itself consist of a number of concurrent processes, the transputer has to support the **occam** programming model internally. The transputer, therefore, has a microcoded scheduler that shares the processor time between the concurrent processes. The scheduler provides two priority levels; any high priority process that can run will do so in preference to any low-priority process. Programming of I/O, interrupts and timing is standard on all transputers and conforms to the **occam** model. Therefore, a transputer is an '**occam** machine' that hosts the language in a similar way as a microprocessor is the machine of its own instruction set. **occam** is therefore called the 'assembler' of the transputer by some authors [Dowsing,1988].

The implementation of parallelism on a single transputer is achieved by time-slicing different processes. The transputer has a hardware scheduler to optimize such time-slicing. The **occam** program however, is the same whether it is run on a single or multi-transputer system. In the case where there is communication between the concurrent processes, it takes place via memory-to-memory data transfer in a single-transputer system, and via standard links (**inmos** links) for a multi-transputer system.

Transputer Internal Architecture

The CPU contains six registers which are used for execution of sequential process. The small number of registers, together with the simplicity of the instruction set, enables the processor to have a relatively simple and fast data-path and control. The functions of the six registers, shown in Fig.A.1, are:

(1) the workspace pointer which points to an area of store where local variable are kept,
(2) the instruction pointer which points to the next instruction to be executed,
(3) the operand register which is used in the formation of instruction operands and
(4) the **A**, **B** and **C** registers which form an evaluation stack, and are the sources and destinations for most arithmetic and logical operations.

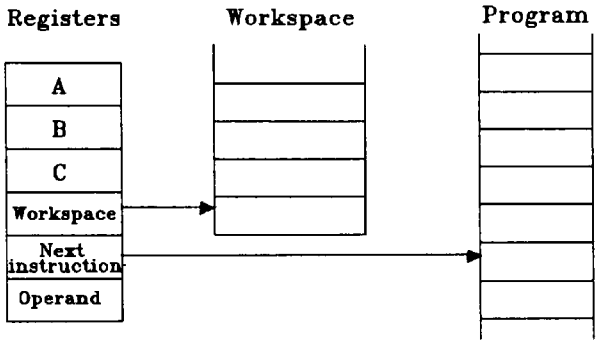


Fig.A.1 Transputer registers

Loading a value into the stack pushes **B** into **C**, and **A** into **B**, before loading **A**. Storing a value from **A** moves **B** into **A** and **C** into **B**. Expressions are evaluated on the evaluation stack, and the instructions refer to the stack implicitly. For example, the 'add' instruction adds the top two values in the stack and places the result on the top of the stack. The use of the stack removes the need for instructions to point to the location of their operands. It is shown from statistics that three registers provide an effective balance between code compactness and implementation complexity. There is no hardware mechanism to detect that more than three values have been loaded to the stack. The compiler ensures that this never happens.

Instructions

Since the transputer was designed to be programmed in a high-level language like **occam**, the transputer instruction set has been made very simple and efficient for compilation. However, the majority of the users who program in **occam** or other high level languages like C or Pascal, would not need much or any knowledge of the instruction set. The instruction set contains only a few instructions, all with the same format, chosen to give compact representation of the operations most frequently occurring in the programs. The instruction set is independent of wordlength, allowing the same microcode to be used for transputers with different wordlengths. Each instruction consists of a single byte divided into two 4-bit parts, as shown in Table A-1. The four most significant bits of the byte are a function code, and the four least significant bits are a data value.

FUNCTION		DATA	
7	4	3	0

Table A-1 Instruction format

The representation provides 16 functions, each with a data value ranging from 0 to 15. Thirteen of these are used to encode the most important functions, which include:

load constant, load local, load non-local, jump, add constant, store local
store non-local, conditional jump, load local pointer and call.

Table A-2 Main functions of the transputer instruction set

Two more of the function codes, namely prefix and negative prefix, are used to allow the operand of any instruction to be extended in length. The last function code is 'operate', which causes the operand to be interpreted as an operation on the values held in the evaluation stack. This allows up to sixteen such operations to be encoded in a single byte instruction.

Support For Concurrency

The processor of the transputer provides efficient support for the **occam** model of concurrency and communication. It has a microcoded scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. A software kernel is therefore not needed. The allocation of space to concurrent processes is performed by the **occam** compiler.

At any time, a concurrent process may be active or inactive. An active process is one that is being executed or queuing up to be executed. An inactive process is one that is ready to input/output, or waiting until a specified time. The scheduler operates in such a way that the inactive processes do not take up any processor time. The active processes waiting to be executed are held on a list. This is a linked list of process workspaces, implemented by two registers, one of which, the Front register, points to the first process on the list, whereas, the Back register points to the last. In Fig.A.2, W is executing, and U, V and X are active processes awaiting execution. Y, however, is an inactive process. This process can be represented by the following **occam** program:

```
PAR
  U
  V
  W
  X
  Y
```

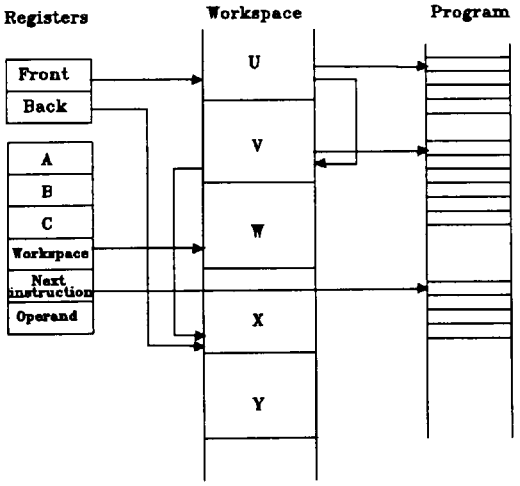


Fig.A.2 Support for concurrency

Scheduling And Priority

When an executing process is no longer able to proceed, then the scheduler must store the value of the 'next instruction' register in that process's workspace. The process pointed by the Front register then has its program pointer restored into the next instruction register. This new process will then continue its execution after the value of the Front register has been updated. If there is no prioritized process within the program, this scheduling and descheduling process works by means of a timeslicing mechanism. One timeslice is about 820 microseconds. The timeslicing mechanism is very efficient due to three factors:

(a) the scheduler is hardware,
(b) the shared registers of the transputer have been designed to minimal and
(c) there is no need to store a value from the evaluation stack when a process is suspended, nor to restore the the value from the evaluation stack when a suspended process is executed again; this is because a process cannot besuspended during the evaluation of an expression.

These factors provide the transputer with the basis of an efficient **occam**-machine. Two levels of priority are supported, namely the low priority and high priority and is implemented by the **occam** constructor **PRI PAR**. Only two components should be prioritized:

```
PRI PAR
  PAR
    ... high priority process
  PAR
    ... low priority process
```

There are two separate sets of front and back register, one for each priority queue. The scheduling algorithms for these two priority levels are quite different. First, a high priority executable process of course has preference to any low priority process. Second, once the high priority process has started executing, it will continue unless it terminates itself, stops, waits for communication, or delays. On the contrary, the low priority scheduler uses a 'round-robin' algorithm. Provided no high priority process is executable, a low priority one that is first in the waiting queue will be chosen. It will then execute until one of the following occurs:

(a) The low priority process itself terminates, stops, waits for communication, or delays.
(b) A high priority process becomes executable. This may occur if:
  1) it has been delayed and the delay time has now expired;
  2) it has been waiting to communicate with a low priority process, or a process on a different transputer, which has now reached this point of communication;
  3) it has been waiting to synchronise with the channel mapped on to the `EventIn` pin and an external interrupt on this pin has been detected
(c) It has been executing for between 1 and 2 timeslices and has reached the end of a control structure.

Thus, each low priority process has a fair share of the processing time of the transputer. The currently executing process will then be suspended at the next appropriate point when the process has finished using the stack. The high priority process will then be swapped in for execution. There is, in addition, a checking facility that imposes an upper limit upon the time it takes for the low priority process to reach a point at which it can be suspended. A typical value for such an upper limit is 40 processor cycles. Hence, the worst case real-time response can be determined.

Although the hardware scheduling scheme provides a very efficient mechanism to implement concurrency, it also imposes some inflexibilities upon the programmers. It is not possible, for example, to upgrade a low priority process temporarily for its communication with a high priority process. Thus, the high priority process in such a case is similar to a low priority. Another disadvantage is that there are only two levels of priorities.

Timers
The transputer supports two timers, one for the high priority process and one for the low priority process. The value of these timers can be read within an **occam** program by defining a TIMER variable. For high priority processes the value of the clock increments every 5 cycles of the clock input. This means that the timer will have a typical resolution of 1 microsecond and has a period of about 72 minutes on a 32-bit processor. For the low priority process, the resolution is 64 microseconds and the period is 74 hours. A typical delay statement in **occam** is as follows:

```
clock ? AFTER d
```

assuming `'clock'` has been defined as a **TIMER**.
The comparison between d and the local clock is performed by a modulo comparison to ensure no error is introduced due to overflow. With the low priority timer, a process can be delayed for up to 37 (74/2) hours. The processes that are delayed are placed on a single queue and are ordered according to the time each process should become executable again. Processes can be added to the list on the queue at any point.

Channels
The concept of channel is central to the philosophy of the transputer. A channel is used to transmit information not only between processes in the same processor but also between processes on different processors. In the former case, the transmission of data is said to be via an 'internal channel', where the channel is a memory address and communication is performed by copying data to the memory location. In the latter case, transmission of data between different processors is via an 'external channel', where the data is transmitted over a serial link. A channel is controlled by a control word in memory that holds a process identifier as well as its priority information. Eight memory locations at the lowest end of the address space are reserved as channel control words for the external channels connected to the duplex serial links. The internal channels can be defined by any other memory locations, and must first be initialized to the most negative integer.

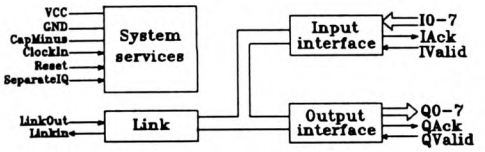
Transputer Support Devices
One of the important features, due to the novel architecture of the transputer, is that it does not require a wide range support of peripheral devices as in a conventional microprocessor. Only two kinds of support devices are commonly used in applications, the link adaptors which provide the connection between a transputer and a

conventional bus-orientated microprocessor system and the link switch which provides a programmable electronic switch in the configuration of the links between transputers.

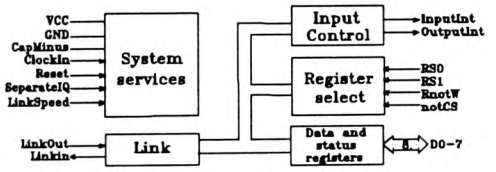
(a) Link adaptors
The link adaptor is a serial to parallel conversion device for the transputer. Two link adaptors are available from **inmos**, the IMSC011 and IMSC012 (or C011 and C012).Fig.A.3 shows the block diagrams of the devices. C011 can be programmed to Mode 1, which provides separate 8-bit parallel data I/O ports; and Mode 2 which has a single parallel I/O port. It is noted that Mode 2 of C011 is functionally identical to the C012 device.

The link adaptor chip provides interface for one byte-wide I/O devices with handshaking facility. Compared with the conventional bus-orientated interfacing, this is an attractive solution since the device will now interface to the simple, clearly defined **inmos** link standard supported by all transputer family processors, and many other families supported devices. In fact, most commercial transputer boards including, for example the **inmos** TRAM modules, provide only the link interface and no access to the memory bus. This makes the conventional memory-mapped interfacing rather complex.

(b)The IMSC004 Link Switch
The IMSC004 (or C004) link crossbar switch provides 32 inputs and 32 outputs, and can be programmed to connect in any configuration. Fig.A.4 shows the block diagram of the C004. The device is programmed by a programming link which runs at the standard rate of the transputer links (10M, 20Mbit/s). The B008 board uses the C004 link switch to program the connection of the links of the transputers.



(a) mode 0



(b) mode 1

Fig.A.3 Block diagram of link adaptor (IMSC011,IMSC012)

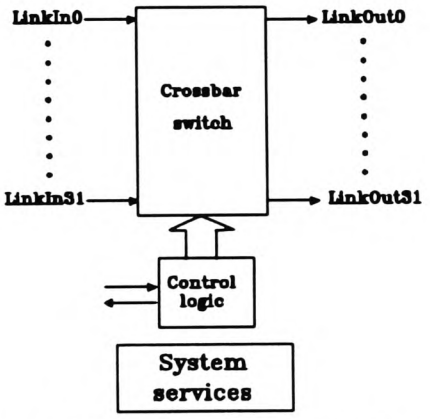


Fig.A.4 Block diagram of C004 link switch

Basics of occam

The software building block of **occam** is the 'process'. A system is designed in terms of an interconnected set of processes. Each process can be regarded as an independent unit of design. It communicates with other processes by channels. The system design is therefore hierarchically structured. Thus, at any level of design, the designer is concerned only with a small and manageable set of processes. The processes can be used to build up larger and more complex processes called 'construct'.

It is not appropiate to give a comprehesive description of **occam** here. However, the following overview explains the basic concepts of language and gives the necessary details which are required to understand the programs in this thesis.

Primitive Processes
There are three primitive processes which can combine to form larger processes and are the building block of **occam** programs.
a) Assignment

```
v := e
```

meaning expression e is assigned to variable v
b) Input

```
c ? x
```

meaning channel c inputs its value to variable x

c) Output

```
c ! y
```

meaning variable y outputs its value to channel c

Constructs
The constructs are combinations of processes. The concept of indentation is used to specify the different levels within a construct. An indentation of two spaces represents a level. There are five distinct constructs.
(a) Sequential construct **SEQ**
A sequential construct is represented by :

```
SEQ
  p1
  p2
  p3
  ...
```

The processes p1, p2, p3 and so on are executed one after the other. The construct terminates when the last process terminates. These sequential constructs are similar to conventional programming languages but are provided with performance and efficiency equivalent to that of an assembler of the latter.

(b) Parallel construct **PAR**
A parallel construct is represented by:

```
PAR
  p1
  p2
  p3
  ...
```

The processes p1, p2, p3 and so on are concurrent processes and are executed in '**PAR**allel'. The construct terminates after all the components have terminated.
This is a unique feature of **occam** that enables simple programming of concurrency inherent in real-time systems.

(c) Conditional construct **IF**
The conditional construct is represented by:

```
IF
  condition1
    p1
  condition2
    p2
...
```

The process p1 is executed if *condition1* is true, otherwise p2 is executed if *condition2* is true, and so on. Only one process is executed and then the construct terminates.

(d) The construct **WHILE**
The WHILE construct acts on a single process, re-executing it each time provided that the associated Boolean expression evaluates to **TRUE**. The construct terminates when the Boolean expression evaluates to **FALSE**. A typical **WHILE** construct takes the following form.

```
WHILE boolean.expression
  single.process
```

The *boolean.expression* takes either of the Boolean constants - **TRUE** or **FALSE**. The process single.process will be re-executed until boolean.expression has the value **FALSE**.

(e) The construct **ALT**
The **ALT** construct takes the form:

```
ALT
  G1
    P1
  G2
    P2
  ...
```

where Gi (i=1,2...) is a guard condition and Pi (i=1,2..) is a subprocess. The first ready subprocess is executed provided the guard condition is met and the process terminates.

Transputer Development System (TDS)

During the program development process it is always necessary that some operating system facilities such as access to a disk filing system, terminals and the compiler for a high-level language, be available. An example

of such a development system is the Philips Development system for the Motorola MC6800 processors. Recently, development systems which use a personal computer (PC) as a host system and a separate emulator connected to the PC, are introducted. The Ashling Development system is one example of these systems using a PC as a host and different emulators to emulate different Intel processors such as 8051 or 8096. The TDS is similar to the latter type of development system in that it also uses a PC as a host system. The host provides the environmental support to the transputer module by providing mechanical support for the board, power supply, power on reset, file servers, keyboard and screen. This transputer board provides the memory (2M bytes of RAM are required for the late version of the TDS) and the interface to the PC bus and a link to the outside world.

The TDS used was one of the first development systems developed by **inmos** itself. Several versions of the system have evolved in the past two to three years, making it a rather futile task to upgrade the system whenever a new version was released. There were some major improvements over the earliest versions. From version D700C to D700D, for example, the toolkits are much easier to access and run. The library facilities are greatly enhanced as well. The programs developed under the D700C and earlier versions were all changed so they could run under the D700D version. No attempt thereafter would be made to upgrade the programs written in the D700D version, although the D700E has been released at the time of writing this thesis. The D700D TDS, also referred as TDS2, allows **occam** programs to be written, compiled and then run from within the development system. The TDS2 comprises an integrated folding editor, a file manager, a compiler and a debugging system. Most of the development system runs on the transputer board; there is a program that runs on the host computer called a 'server' (an upgraded version called 'iserver' was developed later). The 'server' program, written in both C and Intel 8088 assembler, supports protocol down to the links, providing access to the screen, keyboard, and filing system of the host. The protocol is used by the various utilities that run on the transputer board. Programs may also be configurated to run in a target network of transputers; these may range from a single transputer on an evaluation board to networks of several hundred transputers. The code for a transputer network may be loaded directly from the Transputer Development System, through a link connected to the TDS transputer to the target network. Programs may also be placed into a file separate from the TDS, or into a ROM (Read-Only Memory), and used to load a network.

Folding Editor

The user interface is based on a full screen structured editor, exploiting the concept of folding. Folding provides a very effective method of navigating around and viewing selected parts of a large design, and yet operates on an ordinary text VDU. Folding is similar to taking a letter or document with headed paragraphs and then folding the paper so that the text of the paragraph is invisible, leaving only the heading visible. With a suitably folded program the programmer can see the structure of the program at a glance without any distracting detail. Individual sections can then be unfolded so that the detail can be worked on. The user can create and delete folds at will as the text of the program is created and edited. The editor imposes no restriction on folding, which can be nested to any depth. It also provides the editing convenience of being able to reorganize the major structure of a program simply and easily. For example, moving or copying a fold is achieved by a keystroke. Fig.A.5 shows one of the occam programs in the fold editor environment.

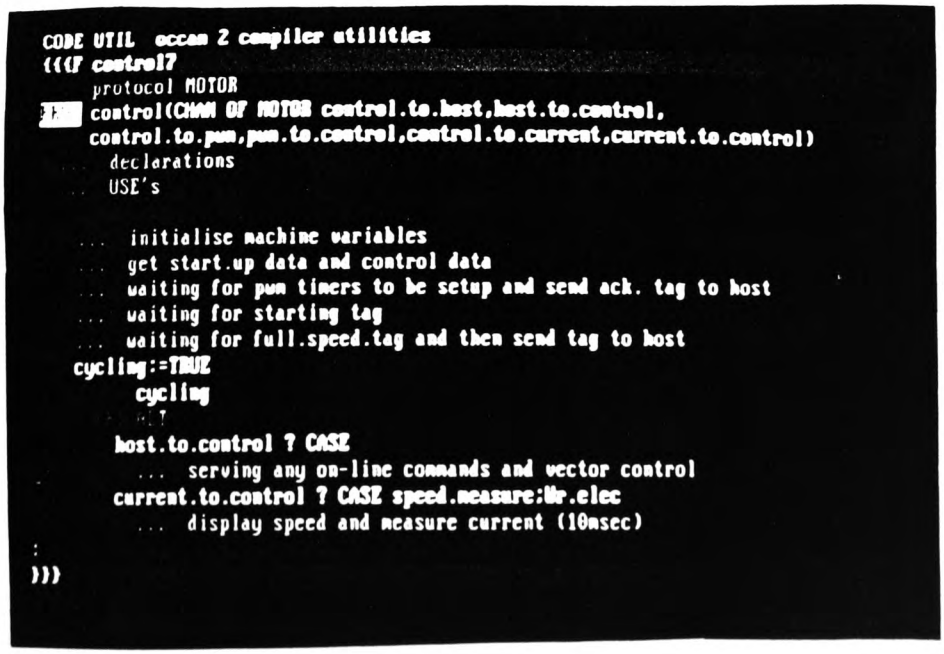


Fig.A.5 An OCCAM program in fold editor environment

Configuration

The logical development of the **occam** program can be done in a single transputer under the TDS system, such as the B008 board with the TDS system D700D. If an application program is to be run on a transputer network, the program must be expressed in a number of processes using the construct PAR. The description of information, in addition to the main program, about the link topology and the association of code to individual transputers is called '**configuration**'. The parallel processes will then be allocated to different processors. Once the logical model of the program is completed, it can be compiled and run on a single transputer. Any runtime error can be checked and corrected at this stage. Configuration in **occam** is very simple due to the internal hardware support for concurrency of the transputer. The configuration will have no logical effect on the behaviour of the program. However, overall system performance in terms of speed and fault tolerance is usually improved as a result.

Examples of Configuration

Two examples of configuration are discussed below. The first example is a typical transputer network system that comprises only of transputers. The second example shows the configuration of the transputer with a non-transputer device.

(a) A Master-Slave Task

Fig.A.6(a) shows the logical model of a master task `p` with a slave task `q`, where the master allocates work to the slave as it becomes ready. If the channel of communication between the master and slave, `p.to.q`, passes only integer type data (**INT**), then a top-level **occam** description of the program is as follows:

```
CHAN OF INT p.to.q:
  PAR
    p(p.to.q)
    q(p.to.q)
```

If the program is divided over a network of 2 T800 processors, with `p` running on processor 0 and `q` on the processor 1, the statement **PAR** should be replaced by **PLACED PAR**. In addition a **PROCESSOR** statement and a **PLACE...AT** statement are needed to indicate the type of processor and the address of the physical link used for each process. Fig.A.6(b) shows the configuration of the logical model into two processors. The solid line represents the physical link (hard channel) and the dotted line represents the **occam** channel (soft channel), which is in effect a memory location in the transputer. A physical link consists of one out-going wire and one in-coming wire, and can be configurated to two **occam** channels. The top-level **occam** program for the configuration is given in Fig.A.6(c). In the program, since `p.to.q` is common to the two processors, it is declared before **PLACED PAR**.

If the processes `p` and `q` communicate in both directions, the link should then be configurated to two **occam** channels `p.to.q` and `q.to.p`. (An **inmos** standard link cable contains two signal wires and therefore allows traffic of data in both directions.) If the data to be transferred are of mixed data types such as **REAL**, **INTEGER** and **BYTE**, a protocol should be declared for `p.to.q` and `q.to.p`. Suppose `p` always outputs data to `q` as a real and integer pair, and `q` always returns with data in a byte type, the following **PROTOCOL** statements are used to setup the `p.to.q` and `q.to.p` channels:

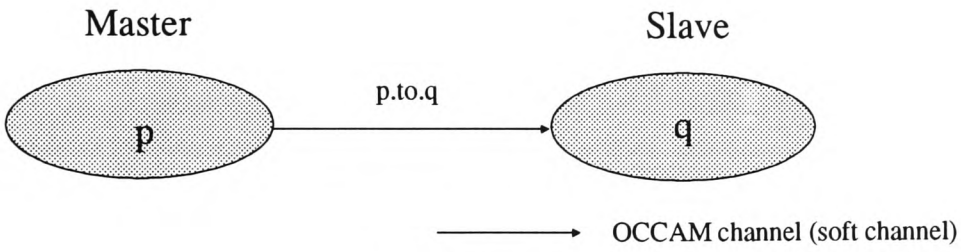
```
PROTOCOL outgoing REAL32;INT:
PROTOCOL incoming BYTE:
```

The top-level **occam** statements is given in Fig.A.6(d).

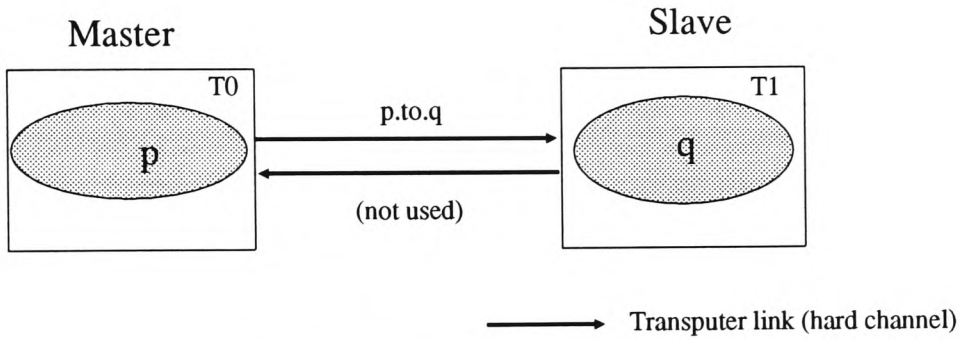
To enhance the performance of the system, the slave task is further decomposed into 4 processes (`q1, q2, q3 and q4`) and its new logical model is shown in Fig.A.7(a). If five processors are available, then a possible allocation of the processes to the processors is shown in Fig.A.7(b). In the figure, each hardware link is either configured with two logical channels (bi-directional), or with only one channel (uni-directional), or left unconfigured. The compiler will give a warning message for any unconfigured links. The top level **occam** program for the new configuration is Fig.A.7(c).

(b) An Interrupt Handler

This example of configuration shows the ease of programming in **occam** for a transputer-based interrupt handler circuit. The hardware connection assumes that a single interrupt line from an external device has been tied to the transputer's event pin, as shown in Fig.A.8(a). A signal transition on the event pin acts as a channel communication, and can be used to detect interrupts from the peripheral. The 'interrupt' channel needs only to be placed (by the **PLACE...AT** statement) at the address of the event pin in the transputer's memory map. The '?' and '!' symbols represent channel input and output operations respectively. The **occam** statements needed to define the process are shown in Fig.A.8(b). This process can then be executed in parallel with appropriate device handling process as shown in Fig.A.8(c), along with the **occam** program which top level is shown along side.



(a) Logical model



(b) Configuration of a 2-processor system

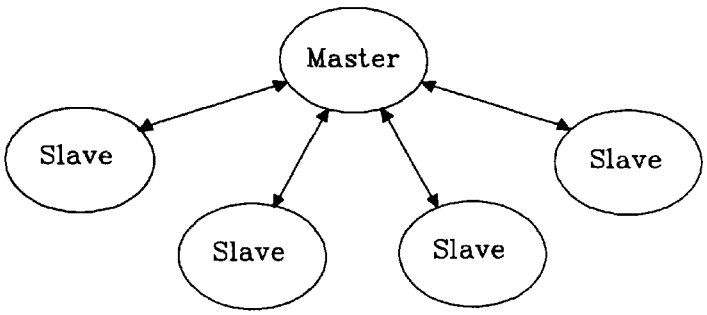
```
CHAN OF INT p.to.q:
PLACED PAR
  PROCESSOR 0 T8
    PLACE p.to.q AT link2out:
    p(p.to.q)
  PROCESSOR 1 T8
    PLACE p.to.q AT link3in:
    q(p.to.q)
```

(c) Occam program for 2-processor system at configuration level

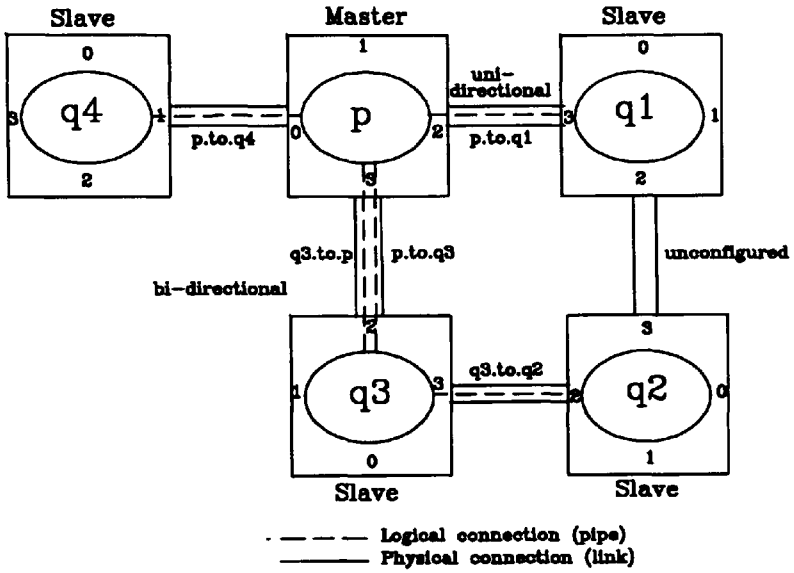
```
CHAN OF outgoing p.to.q:
CHAN OF incoming q.to.p:
  PLACED PAR
    PROCESSOR 0 T8
          PLACE p.to.q AT link2out:
          PLACE q.to.p AT link3in:
          p(p.to.q)
        PROCESSOR 1 T8
          PLACE p.to.q AT link3in:
          PLACE q.to.p AT link2out:
          q(p.to.q)
```

(d) Occam program for 2-processor system at configuration level when processes p and q communicate in both directions

Fig.A.6 A master-slave task



(a) New logical model for the master-slave task

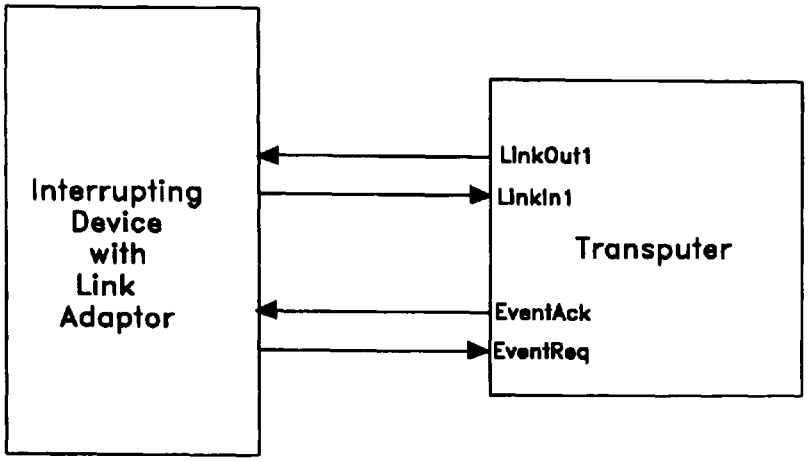


(b) A 5-processor system for the model in (a)

```
CHAN OF outgoing p.to.q3:
CHAN OF incoming q3.to.p:
CHAN OF BYTE p.to.q1,q3.to.q2,p.to.q4:
PLACED PAR
  PROCESSOR 0 T8
    PLACE p.to.q4 AT link0out:
    PLACE p.to.q1 AT link2out:
    PLACE p.to.q3 AT link3out:
    PLACE q3.to.p AT link3in:
    p(p.to.q1,q3.to.p,p.to.q3,p.to.q4)
  PROCESSOR 1 T8
    PLACE p.to.q1 AT link3in:
    q1(p.to.q1)
  PROCESSOR 2 T8
    PLACE q3.to.q2 AT link2in:
    q2(q3.to.q2)
  PROCESSOR 3 T8
    PLACE q3.to.p AT link2out:
    PLACE p.to.q3 AT link2in:
    PLACE q3.to.q2 AT link3out:
    q3(q3.to.p,p.to.q3,q3.to.q2)
  PROCESSOR 4 T8
    PLACE p.to.q4 AT link1in:
    q4(p.to.q4)
```

(c) Occam program

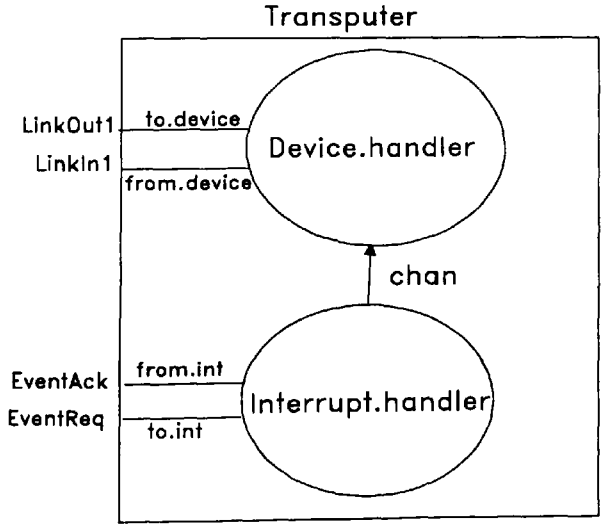
Fig.A.7 Enhancement of the master-slave task



(a) Hardware connection

```
PROC interrupt.handler (CHAN OF BYTE to.device.handler)
  CHAN OF BYTE event:
  PLACE event AT event.in:
  BYTE data.register:
  PLACE data.register AT #6000:
  WHILE TRUE
    SEQ
      event ? interrupt
      value:=data.register
      to.device.handler ! value :
```

(b) OCCAM statements to define the interrupt handling




```
CHAN OF BYTE chan:
PRI PAR
  interrupt.handler(chan)
  device.handler(chan,...)
```

(c) OCCAM model and program

Fig.A.8 Interrupt handler

Procedure for Configuration

To create an **occam** program to be run on a network of transputers, the section of **occam** program to be allocated onto a processor should be contained within a SC fold. Different SC folds can be compiled separately. The compiled SC's must then be collected into a filed fold, to which the **MAKE FOLDSET** utility is applied with the parameter set to **PROGRAM**. The **PROGRAM** foldset also contains the configuration statements which describe the inter-connections and call the required procedures on the desired processors.

PROGRAM and EXE Programs

The TDS2 recognizes two different types of program, known as **EXE** and **PROGRAM**. An **EXE** program was on the host transputer, and may access the keyboard, screen, and filing system of the host machine. A **PROGRAM**, on the other hand, runs on a network of one or more transputers, and is loaded from the host transputer via a transputer link. This link may be the network's only connection with the outside world.

Typically, when a **PROGRAM** is loaded onto a multiple transputer network, a simple **EXE** program will also be run on the transputer that monitors the output transmitted back from the **PROGRAM**, sends results to the screen, passes on any input from the keyboard and controls the TDS filing system, as required. In the master-slave task example, the `p` process may be conveniently run as **EXE** program and `qi` (`i=1` to `4`) processes run as **PROGRAM** in the networks.

The top-level system description language will be:

```
...  EXE p
...  PROGRAM qi
```

The `EXE p` fold is the **occam** program that runs on the host transputer. The **PROGRAM** `qi` fold, which runs on the transputer network, consists of the declaration of channel protocols, configuration information and the separate complied SC programs for the `qi` processes. The EXE and PROGRAM usually communicate via transputer links. The **PROGRAM** fold, when unfolded, will look like:

```
{{{  PROGRAM qi
{{{F qi
...  PROTOCOL declarations
...  SC q1
...  SC q2
...  SC q3
...  SC q4
...  configuration
}}}
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program uses state variable method to solve the machine equations%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input data for parameters of Pulse-width modulation sampled waveform %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

F  = input('Input modulating frequency in Hz (F) : ');
P  = input('No. of cycles simulated (P) : ');
Q  = input('No. of sample per cycle (Q) : ');
dt = 1/(Q*F);   %  Calculate time increment
Kf = input('Friction Coeff (Kf): ');
J  = input('Inertia of Rotational Mass: ');
V  = input('Phase Voltage (rms): ')
V  = V*sqrt(2); %  Vrms to Vm

N=(1/F)/dt

% clear initdata

!del initdata.mat

echo on
            %%%%%%%%%%%%%%%%%%%%%%%%
            % Input machine data %
            %%%%%%%%%%%%%%%%%%%%%%%%
echo off

%% Inductance in Henrys, Resistance in ohms, Voltage in Volts

M=0.268;
ls=0.01165;
lr=0.02790;
Ls=M+ls;
Lr=M+lr;
Rs=3.76;
Rr=2.571;

% Machine model is based on transformation referred to stator.
% Formation of inductance, resistance and rotational matrice.

echo on
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % generation of sine waves %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

for i=1:P*N;
  ha(i)=V*sin(i*2*pi/N);
  hb(i)=V*sin((2*pi/N)*(i+N/3));
  hc(i)=V*sin((2*pi/N)*(i+2*N/3));
end

for k=1:P*N;
  V1(k)=ha(k)-hc(k);
  V2(k)=hb(k)-hc(k);
end;

clear ha;
clear hb;
clear hc;

%initialisation of state variables at time=0
% currents, angular velocity, angular position and torque set to zero
IA=0;IB=0;Ia=0;Ib=0;Wr=0;fi=0;Wr_=0;fi_=0;Te=0;
x0=[0 0 0 0]';
echo on
            %%%%%%%%%%%%%%%%%%%%%%%%
            % Simulation begins %
            %%%%%%%%%%%%%%%%%%%%%%%%
echo off

! TM start
% start with i=2, as V(1)=[0 0] gives x1=[0 0 0 0] and will duplicate the
% previous value

for i=2:P*N;
%%%%%%%%%%%%%%%%%%%%%%%
% Impedance routine %
%%%%%%%%%%%%%%%%%%%%%%%
  a1=fi_ ;
  a2=fi_-2*pi/3;
  a3=fi_+2*pi/3;

L=[2*Ls          Ls     -2*M*cos(a3)   2*M*cos(a2);
    Ls         2*Ls     2*M*cos(a2)  -2*M*cos(a1);
-2*M*cos(a3)  2*M*cos(a2)   2*Lr         Lr      ;
 2*M*cos(a2) -2*M*cos(a1)    Lr         2*Lr     ];

Z=[   2*Rs            Rs          2*M*Wr_*sin(a3)  -2*M*Wr_*sin(a2);
      Rs            2*Rs        -2*M*Wr_*sin(a2)   2*M*Wr_*sin(a1);
 2*M*Wr_*sin(a3)  -2*M*Wr_*sin(a2)     2*Rr              Rr;
-2*M*Wr_*sin(a2)   2*M*Wr_*sin(a1)      Rr              2*Rr];

  A=-inv(L)*Z;
  B=inv(L);
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  % Digitalising the State Equation.... %
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
  [phi,gam]=c2d(A,B,dt);
  x1=phi*x0 + gam*[V1(i) V2(i) 0 0]';

  %% currents to be calculated
  %I=C*x1;
  %IA_=I(1);IB_=I(2);      %IC_=I(3);Ia_=I(4);Ib_=I(5);Ic_=I(6);
  %IA=[IA;IA_];IB=[IB;IB_];  %Ia=[Ia;Ia_];Ib=[Ib;Ib_];

dLdF=[   0           0          2*M*sin(a3)  -2*M*sin(a2);
         0           0         -2*M*sin(a2)   2*M*sin(a1);
 2*M*sin(a3)  -2*M*sin(a2)          0             0;
-2*M*sin(a2)   2*M*sin(a1)          0             0];

  Te_=-x1'*dLdF*x1;

  k0=dt*(2/J)*(Te_-Kf*Wr_);   % the factor 2 is due to no. of pole pair
  k1=dt*(2/J)*(Te_-Kf*(Wr_+k0/2));
  k2=dt*(2/J)*(Te_-Kf*(Wr_+k1/2));
  k3=dt*(2/J)*(Te_-Kf*(Wr_+k2));
  Wr_=Wr_+k0/6+k1/3+k2/3+k3/6;
  fi_= fi_+ Wr_*dt;
  Wr=[Wr;Wr_];
  Te=[Te;Te_];
 x0=x1;
        % counter i %
 [i Te_ Wr_]
 end;
! TM stop
 % save data F,dt,J,Kf,P,V1,V2,fi_,Te_,Wr_,x0
 save initdata F dt J Kf P V1 V2 fi_ Te_ Wr_ x0
 clear V1;
 clear V2;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This program uses the state variable method to solve the machine
%    equations.  Electrical and mechanical transients are found.
%    True values are used.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

echo on        %%%%%%%%%%%%%%%%%%
               % Input data  %
               %%%%%%%%%%%%%%%%%%
echo off

F  = input('Input modulating frequency in Hz (F) : ');
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    This program uses the state variable method to solve the machine
%    equations.  Electrical and mechanical transients are found.
%    True values are used.
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

echo on
            %%%%%%%%%%%%%%%
            % Input data  %
            %%%%%%%%%%%%%%%
echo off

F  = input('Input modulating frequency in Hz (F) : ');
P  = input('No. of cycles simulated (P) : ');
Q  = input('No. of sample per cycle (Q) : ');
Kf = input('Friction Coeff (Kf): ');
J  = input('Inertia of Rotational Mass: ');
V  = input('Phase Voltage (rms): ')

dt = 1/(Q*F);   %  Calculate time increment
V  = V*sqrt(2); %  Convert Vrms to Vm


N=(1/F)/dt

% clear initdata

!del initdata.mat

echo on
            %%%%%%%%%%%%%%%%%%%%%%%%%%
            % Input machine data %
            %%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

%% Inductance in Henrys, Resistance in ohms, Voltage in Volts

Lm=0.268;
ls=0.01165;
lr=0.02790;
Ls=Lm+ls;
Lr=Lm+lr;
Rs=3.76;
Rr=2.571;

% Machine model is based on transformation referred to stator.
% Formation of inductance, resistance and rotational matrice.

echo on
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % generation of sine waves %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

for i=1:P*N;
 ha(i)=V*sin(i*2*pi/N);
 hb(i)=V*sin((2*pi/N)*(i+N/3));
 hc(i)=V*sin((2*pi/N)*(i+2*N/3));
end

% d-q transformation of the 3-phase stator voltages

C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];

for k=1:P*N;
 u=C*[ha(k) hb(k) hc(k)]';u=u';
 uds(k)=u(1);uqs(k)=u(2);
end

clear ha;
clear hb;
clear hc;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Initialisation of state variables at time=0             %
% currents, angular velocity, angular position and torque set to zero %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Wr=0;fi=0;Wr_=0;fi_=0;Te=0;time_=0;time=0;
ids=0;iqs=0;idr=0;iqr=0;ids_=0;iqs_=0;idr_=0;iqr_=0;
x0=[0 0 0 0]';

echo on
            %%%%%%%%%%%%%%%%%%%%%%%
            % Simulation begins %
            %%%%%%%%%%%%%%%%%%%%%%%
echo off

L=[Ls 0 Lm 0;
   0 Ls 0 Lm;
   Lm 0 Lr 0;
   0 Lm 0 Lr];

Res=[Rs 0 0 0;
     0 Rs 0 0;
     0 0 Rr 0;
     0 0 0 Rr];

G=[ 0   0   0   0;
    0   0   0   0;
    0   Lm  0   Lr;
   -Lm  0  -Lr  0];

! TM start

for i=2:P*N;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Machine model is based on transformation referred to stator.
% Formation of inductance, resistance and rotational matrice.
% Change to matrice suitable for standard State-variable form
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  A=-inv(L)*(Wr_*G+Res),
  B=inv(L);

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Digitalising the State Equation....
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  [phi,gam]=c2d(A,B,dt);
  x1=phi*x0 + gam*[uds(i) uqs(i) 0 0]';
  ids_=x1(1);iqs_=x1(2);idr_=x1(3);iqr_=x1(4);
  ids=[ids;ids_];iqs=[iqs;iqs_];%idr=[idr;idr_];iqr=[iqr;iqr_];
  Te_=3*Lm*(idr_*iqs_-ids_*iqr_);
  k0=dt*(2/J)*(Te_-Kf*Wr_);   % the factor 2 is due to no. of pole pair
  k1=dt*(2/J)*(Te_-Kf*(Wr_+k0/2));
  k2=dt*(2/J)*(Te_-Kf*(Wr_+k1/2));
  k3=dt*(2/J)*(Te_-Kf*(Wr_+k2));
  Wr_=Wr_+k0/6+k1/3+k2/3+k3/6;
  fi_= fi_+ Wr_*dt;
  Wr=[Wr;Wr_];
  Te=[Te;Te_];
  x0=x1;
  %dT=Te_-Kf*Wr_;
  %time_ = time_ + dt;  % update time
  %time=[time;time_];
         % display values %
  [i Te_ Wr_]
 end;

! TM stop

 % save data
 save initdata
end

end
```

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
% This program simulates motor response when supplied by a PWM power source. %
% A fixed carrier frequency (1000Hz) is used.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
echo off
dt = input('time increment (msec) [0.1-0.2] : ');
dt = dt/1000;
Tmax = input('time limit for simulation (sec) : ');
N=Tmax/dt    % total no. of samples
tc=0.001;    % use a fixed carrier freq. of 1000 Hz
disp('no. of sample per carrier freq = '); s=tc/dt
Vdc = input('DC link = ');
Wrref = input('input ref.speed (rpm) = ');
f=Wrref*50/1500;
M=input('input mod.ind. = ');
Kf = input('Friction coeff. = ');
J  = 0.02; % input('Moment of inertia = ');
%
%% Inductance in Henrys, Resistance in ohms, Voltage in Volts
Lm=0.268;
ls=0.01165;
lr=0.02790;
Ls=Lm+ls;
Lr=Lm+lr;
Rs=3.76;
Rr=2.571;
%
L=[Ls 0 Lm 0;
   0 Ls 0 Lm;
   Lm 0 Lr 0;
   0 Lm 0 Lr];
%
L_=inv(L);
%
Res=[Rs 0 0 0;
     0 Rs 0 0;
     0 0 Rr 0;
     0 0 0 Rr];
G=[ 0    0    0    0;
    0    0    0    0;
    0    Lm   0    Lr;
   -Lm   0   -Lr   0];
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%           Initialisation of state variables at time=0            %
% currents, angular velocity, angular position and torque set to zero %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
idr=0;iqr=0;ids_=[];iqs_=[];idr_=[];iqr_=[];
Wr_=[];Wr=0;fi=0;Te_=[];Te=0;time=0;
Wr=0;theta=0;
x0=[0 0 0 0]';

We=2*pi*f;
            %%%%%%%%%%%%%%%%%%%%%%%
            % Simulation begins %
            %%%%%%%%%%%%%%%%%%%%%%%
p=1;
n=N/s;
for i=1:n;
   theta=theta + We*(dt*s);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % generation of pulse-width waveform            %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%% find first pulse width of the 3 phases
   A1=theta;
   B1=A1+2*pi/3;
   C1=A1-2*pi/3;
   up_a=(tc/4)*(1-M*sin(A1));
   up_b=(tc/4)*(1-M*sin(B1));
   up_c=(tc/4)*(1-M*sin(C1));
   down_a=(tc/4)*(1+M*sin(A1));
   down_b=(tc/4)*(1+M*sin(B1));
   down_c=(tc/4)*(1+M*sin(C1));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % Generation of 3-phase PWM waveforms     %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for k=1:s
    if k < (s/2)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % pulse widths of 3-phases when carrier wave is at -ve gradient
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     tt=dt*k;
     if tt  down_a
      pwma(p)=-Vdc;
     else
      pwma(p)=Vdc;
     end,
     if tt  down_b
      pwmb(p)=-Vdc;
     else
      pwmb(p)=Vdc;
     end,
     if tt  down_c
      pwmc(p)=-Vdc;
     else
      pwmc(p)=Vdc;
     end,
    else
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % pulse widths of 3-phases when carrier wave is at +ve gradient
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     kk=k-s/2;
     tt=dt*kk;
     %%%%%%%%%%%%%
     % phase a %
     %%%%%%%%%%%%%
     if tt  up_a
      pwma(p)=Vdc;
     else
      pwma(p)=-Vdc;
     end,
     %%%%%%%%%%%%%
     % phase b %
     %%%%%%%%%%%%%
     if tt  up_b
      pwmb(p)=Vdc;
     else
      pwmb(p)=-Vdc;
     end,
     %%%%%%%%%%%%%
     % phase c %
     %%%%%%%%%%%%%
     if tt  up_c
      pwmc(p)=Vdc;
     else
      pwmc(p)=-Vdc;
     end,
    end,
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% d-q transformation again for the input to model %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
    u=C*[pwma(p) pwmb(p) pwmc(p)]';u=u';
    uds(p)=u(1);uqs(p)=u(2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Machine model is based on transformation referred to stator. %
    % Formation of inductance, resistance and rotational matrice.  %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Change to matrice suitable for standard State-variable form
    A=-L_*(Wr*G+Res);
    B=L_;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Digitalising the State Equation....%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    [PHI,GAM]=c2d(A,B,dt);
    x1=PHI*x0 + GAM*[uds(p) uqs(p) 0 0]';
    Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
    % currents recorded
    ids=x1(1);iqs=x1(2);idr=x1(3);iqr=x1(4);
    ids_=[ids_;ids];iqs_=[iqs_;iqs];  %%idr_=[idr_;idr];iqr_=[iqr_;iqr];

    k0=dt*(2/J)*(Te-Kf*Wr); % the factor 2 is due to no. of pole pair
    k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
    k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
    k3=dt*(2/J)*(Te-Kf*(Wr+k2));
    Wr=Wr+k0/6+k1/3+k2/3+k3/6;
    Wr_=[Wr_;Wr];
    Te_=[Te_;Te];
    x0=x1;
    dT=Te-Kf*Wr;
    percent=p*100/N;
    %%%% display values %%%%
    disp('  percent    up_a    down_a   pwma(p)      Te      Wr(rpm)');
    disp([percent up_a down_a pwma(p) Te Wr*1500/314]);
    p=p+1;
  end;
end

save dqpwmdat
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program is an extension of the program dqpwm1.m. This allows more%
% samples in time domain to be plotted.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Inputs from dqpwmdat.mat file . Outputs (overwrite) to dqpwmdat.mat   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all;
echo on
          %%%%%%%%%%%%%%%
          % Input data  %
          %%%%%%%%%%%%%%%
echo off

% input uds,udq,f,dt,F,N
load dqpwmdat

clear Wr_    % delete previous
clear Te_    %    matrices
clear ids_
clear iqs_

Wr_=Wr;      % reset matrices
Te_=Te;
ids_=ids;
iqs_=iqs;
          %%%%%%%%%%%%%%%%%%%%%
          % Simulation begins %
          %%%%%%%%%%%%%%%%%%%%%
p=1;
n=N/s;
for i=1:n;
   theta=theta + We*(dt*s);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % generation of pulse-width waveform          %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%% find first pulse width of the 3 phases
   A1=theta;
   B1=A1+2*pi/3;
   C1=A1-2*pi/3;
   up_a=(tc/4)*(1-M*sin(A1));
   up_b=(tc/4)*(1-M*sin(B1));
   up_c=(tc/4)*(1-M*sin(C1));
   down_a=(tc/4)*(1+M*sin(A1));
   down_b=(tc/4)*(1+M*sin(B1));
   down_c=(tc/4)*(1+M*sin(C1));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %  Generation of 3-phase PWM waveforms      %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for k=1:s
    if k < (s/2)
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %  pulse widths of 3-phases when carrier wave is at -ve gradient
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     tt=dt*k;
     if tt  down_a
      pwma(p)=-Vdc;
     else
      pwma(p)=Vdc;
     end,
     if tt  down_b
      pwmb(p)=-Vdc;
     else
      pwmb(p)=Vdc;
     end,
     if tt  down_c
      pwmc(p)=-Vdc;
     else
      pwmc(p)=Vdc;
     end,
    else
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %  pulse widths of 3-phases when carrier wave is at +ve gradient
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     kk=k-s/2;
     tt=dt*kk;
     %%%%%%%%%%%%%
     %  phase a  %
     %%%%%%%%%%%%%
     if tt  up_a
      pwma(p)=Vdc;
     else
      pwma(p)=-Vdc.
     end,
     %%%%%%%%%%%%%
     %  phase b  %
     %%%%%%%%%%%%%
     if tt  up_b
      pwmb(p)=Vdc.
     else
      pwmb(p)=-Vdc.
     end,
     %%%%%%%%%%%%%
     %  phase c  %
     %%%%%%%%%%%%%
     if tt  up_c
      pwmc(p)=Vdc.
     else
      pwmc(p)=-Vdc.
     end,
    end,
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    %%% d-q transformation again for the input to model %%%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
    u=C*[pwma(p) pwmb(p) pwmc(p)]';
    uds(p)=u(1);uqs(p)=u(2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Machine model is based on transformation referred to stator.  %
    % Formation of inductance, resistance and rotational matrice.   %
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Change to matrice suitable for standard State-variable form
    A=-L_*(Wr*G+Res).
    B=L_;
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Digitalising the State Equation....%
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [PHI,GAM]=c2d(A,B,dt);
    x1=PHI*x0 + GAM*[uds(p) uqs(p) 0 0]';

    % currents recorded
    ids=x1(1);iqs=x1(2);idr=x1(3);iqr=x1(4);
    ids_=[ids_;ids];iqs_=[iqs_;iqs];  %%idr_=[idr_;idr];iqr_=[iqr_;iqr];

    Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
    k0=dt*(2/J)*(Te-Kf*Wr); % the factor 2 is due to no. of pole pair
    k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
    k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
    k3=dt*(2/J)*(Te-Kf*(Wr+k2));
    Wr=Wr+k0/6+k1/3+k2/3+k3/6;
    Wr_=[Wr_;Wr];
    Te_=[Te_;Te];
    x0=x1;
    dT=Te-Kf*Wr;
    percent=p*100/N;
    %%%% display values %%%%
    disp('  percent    up_a    down_a   pwma(p)      Te      Wr(rpm)');
    disp([percent up_a down_a pwma(p) Te Wr*1500/314]);
    p=p+1;
  end,
end

% clear and update data
!del dqpwmdat.mat
save dqpwmdat
```

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                   Program name : SINVEC3.M
% Voltage Type (V-type) Vector Control supplied by a sinusoidal power
% source. Limits on the outputs of power are imposed.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

dt = input('time increment (msec) [1]: ');
dt = dt/1000;
Tmax = input('time limit for simulation (sec) : ');
N = Tmax/dt;  % total no. of samples
ids = input('id (A) = ');
iqs_limit = input('Limit on iqs (A) = ');
Vds_limit = input('Limit on DC link (V) = ');
% Vqs is set to be the same as Vds. This is only an rough approximation, since
% Vdc link should be the practical limit, and sq(Vdc)=sq(Vds)+sq(Vqs).
Vqs_limit = Vds_limit;
Wrref= input('Reference speed (rpm) = ');
Wrref=Wrref*pi/15; % conversion from rpm to rad/s
gain = input('Gain of speed controller  = ');
Kf = input('Friction coeff. = ');
J  = input('Moment of inertia = ');

%% Inductance in Henrys, Resistance in ohms, Voltage in Volts
Lm=0.268;
ls=0.01165;
lr=0.02790;
Ls=Lm+ls;
Lr=Lm+lr;
Rs=3.76;
Rr=2.571;

% limit counters
counter1=0;counter2=0;counter3=0;

L=[Ls 0 Lm 0;
   0 Ls 0 Lm;
   Lm 0 Lr 0;
   0 Lm 0 Lr];

L_=inv(L);

Res=[Rs 0 0 0;
     0 Rs 0 0;
     0 0 Rr 0;
     0 0 0 Rr];

G=[ 0    0    0    0;
    0    0    0    0;
    0    Lm   0    Lr;
   -Lm   0   -Lr   0];

% Find reference rotor flux from given ids
Fdr=ids*Lm;

Tr=Lr/Rr;  % time constant of rotor
sigma=1-Lm^2/(Ls*Lr);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Initialisation of state variables at time=0
% currents, angular velocity, angular position and torque set to zero
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

We_=0;We=0;Wr=0;fi=0;Wr_=0;fi_=0;Te_=0;Te=0;time_=0;time=0;
theta=0;
idr=0;iqr=0;ids_=0;iqs_=0;idr_=0;iqr_=0;
x0=[0 0 0 0]';

                %%%%%%%%%%%%%%%%%%%%%%%%%%
                % Simulation begins %
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:N;

%               Vector control
Tref=(Wrref-Wr)*gain;
iqs=Tref*(Lr/(Lm*Fdr));
Vds=ids*Rs - iqs*sigma*Ls*We;
Vqs=ids*Ls*We + iqs*Rs;
if iqs  iqs_limit
  iqs=iqs_limit;disp('iqs limit reached')
  counter1=counter1+1
 else
end
if Vds  Vds_limit
  Vds=Vds_limit;disp('Vds limit reached')
  counter2=counter2+1
 else
end
if Vqs  Vqs_limit
  Vqs=Vqs_limit;disp('Vqs limit reached')
  counter3=counter3+1
 else
end

Wsl=iqs/(Tr*ids);
theta=theta + (Wsl+Wr)*dt;
We=Wsl+Wr;

%               Luk's Transformation
Vm=sqrt(Vds^2+Vqs^2);
phi(i)=atan2(Vqs,Vds);
Va=Vm*sin(theta + pi/2 + phi(i));
Vb=Vm*sin(theta + pi/2 + phi(i) + 2*pi/3);
Vc=Vm*sin(theta + pi/2 + phi(i) - 2*pi/3);

% d-q transformation again for the input to model
C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
u=C*[Va Vb Vc]';u=u';
uds(i)=u(1);uqs(i)=u(2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Machine model is based on transformation referred to stator.
% Formation of inductance, resistance and rotational matrice.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Change to matrice suitable for standard State-variable form

  A=-L_*(Wr*G+Res);
  B=L_;

                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
                % Digitalising the State Equation....
                %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  [PHI,GAM]=c2d(A,B,dt);
  x1=PHI*x0 + GAM*[uds(i) uqs(i) 0 0]';
  Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
  k0=dt*(2/J)*(Te-Kf*Wr);   % the factor 2 is due to no. of pole pair
  k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
  k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
  k3=dt*(2/J)*(Te-Kf*(Wr+k2));
  Wr=Wr+k0/6+k1/3+k2/3+k3/6;
  Wr_=[Wr_;Wr];
  Te_=[Te_;Te];
  iqs_=[iqs_;iqs];
  x0=x1;
  dT=Te-Kf*Wr;
  percent=i*100/N;
     % display values %
  disp('   percent      Vqs       Vds      phi      iqs      Te                 Wr');
  disp([percent Vqs Vds phi(i)*180/pi iqs Te Wr*15/pi]);
  %time = time + dt;  % update time
  %time_=[time_;time];
end;

Wr_=Wr_*15/pi;

save simdata
```

Appendix B-6 : Vector control with sinusoidal supply (extension program)

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                 Program name : SINVEC3_.M                                    %
% Voltage Type (V-type) Vector Control supplied by a sinusoidal power
% source. Limits on the outputs of power are imposed.                          %
% Input data from SINVEC3.M
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off

% load data
load simdata
Wr_=Wr;Te_=Te;iqs_=iqs;
% limit counters
counter1=0;counter2=0;counter3=0;

Wrref=input('Input new speed reference : ');
Wrref=Wrref*pi/15; % conversion from rpm to rad/s

             %%%%%%%%%%%%%%%%%%%%%%%%
             % Simulation begins %
             %%%%%%%%%%%%%%%%%%%%%%%%
for i=1:N;

%            Vector control
Tref=(Wrref-Wr)*gain;
iqs=Tref*(Lr/(Lm*Fdr));
Vds=ids*Rs - iqs*sigma*Ls*We;
Vqs=ids*Ls*We + iqs*Rs;
if iqs  iqs_limit
  iqs=5;disp('iqs limit reached')
  counter1=counter1+1
 else
end
if Vds  Vds_limit
  Vds=Vds_limit;disp('Vds limit reached')
  counter2=counter2+1
 else
end
if Vqs  Vqs_limit
  Vqs=Vqs_limit;disp('Vqs limit reached')
  counter3=counter3+1
 else
end

Wsl=iqs/(Tr*ids);
theta=theta + (Wsl+Wr)*dt;
We=Wsl+Wr;

%            Luk's Transformation
Vm=sqrt(Vds^2+Vqs^2);
phi(i)=atan2(Vqs,Vds);
Va=Vm*sin(theta + pi/2 + phi(i));
Vb=Vm*sin(theta + pi/2 + phi(i) + 2*pi/3);
Vc=Vm*sin(theta + pi/2 + phi(i) - 2*pi/3);

% d-q transformation again for the input to model
C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
u=C*[Va Vb Vc]';u=u';
uds(i)=u(1);uqs(i)=u(2);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Machine model is based on transformation referred to stator.
% Formation of inductance, resistance and rotational matrice.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Change to matrice suitable for standard State-variable form

  A=-L_*(Wr*G+Res);
  B=L_;

             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
             % Digitalising the State Equation....
             %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  [PHI,GAM]=c2d(A,B,dt);
  x1=PHI*x0 + GAM*[uds(i) uqs(i) 0 0]';
  Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
  k0=dt*(2/J)*(Te-Kf*Wr);   % the factor 2 is due to no. of pole pair
  k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
  k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
  k3=dt*(2/J)*(Te-Kf*(Wr+k2));
  Wr=Wr+k0/6+k1/3+k2/3+k3/6;
  Wr_=[Wr_;Wr];
  Te_=[Te_;Te];
  iqs_=[iqs_;iqs];
  x0=x1;
  dT=Te-Kf*Wr;
  percent=i*100/N;
    % display values %
  disp('   percent      Vqs      Vds      phi      iqs      Te                 Wr');
  disp([percent Vqs Vds phi(i)*180/pi iqs Te Wr*15/pi]);
  %time = time + dt;  % update time
  %time_=[time_;time];
end;

Wr_=Wr_*15/pi;

save simdata
```

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program simulates the Voltage Type (V-type) Vector Control
% supplied by a PWM  power source.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
dt = input('time increment (msec) : ');
dt = dt/1000;
Tmax = input('time limit for simulation (sec) : ');
N=Tmax/dt   % total no. of samples
tc=0.002    % use a fixed carrier freq. of 500 Hz
s=tc/dt;    % no. of sample per carrier freq.
Vdc = 300;
ids = 5     %input('id (A) = ');
Wrref= 100  %input('Reference speed = ');
gain = 0.12 %input('Gain of speed controller  = ');
Kf = 0.04 %input('Friction coeff. = ');
J  = 0.02 % input('Moment of inertia = ');
%
%% Inductance in Henrys, Resistance in ohms, Voltage in Volts
Lm=0.268;
ls=0.01165;
lr=0.02790;
Ls=Lm+ls;
Lr=Lm+lr;
Rs=3.76;
Rr=2.571;
%
L=[Ls 0 Lm 0;
  0 Ls 0 Lm;
  Lm 0 Lr 0;
  0 Lm 0 Lr];
%
L_=inv(L);
%
Res=[Rs 0 0 0;
    0 Rs 0 0;
    0 0 Rr 0;
    0 0 0 Rr];
G=[ 0    0    0    0;
    0    0    0    0;
    0   Lm    0   Lr;
  -Lm    0  -Lr    0];
%
Tr=Lr/Rr;  % time constant of rotor
sigma=1-Lm^2/(Ls*Lr);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            Initialisation of state variables at time=0          %
% currents, angular velocity, angular position and torque set to zero %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%idr=0;iqr=0;ids_=[];iqs_=[];idr_=[];iqr_=[];
We=0;Wr_=[];Wr=0;fi=0;Te_=[];Te=0;time=0;
We=0;Wr=0;theta=0;
x0=[0 0 0 0]';
%
                                        %%%%%%%%%%%%%%%%%%%%%
                                        % Simulation begins %
                                        %%%%%%%%%%%%%%%%%%%%%

p=1;
n=N/s;
for i=1:n;
   %%%%%%%%%%%%%%%%%%%%%%
   %         Vector control              %
   %%%%%%%%%%%%%%%%%%%%%%
   iqs=(Wrref-Wr)*gain;
   Vds=ids*Rs - iqs*sigma*Ls*We;
   Vqs=ids*Ls*We + iqs*Rs;
   Wsl=iqs/(Tr*ids);
   We=Wsl+Wr;
   theta=theta + We*(dt*s);
   Vm=sqrt(Vds^2+Vqs^2);
   phi=atan2(Vqs,Vds);
   M=Vm/(Vdc*0.78);
   %% set limit on M1
   if M1;M=1;
     disp('M  1');
   end,
   f=We/(2*pi);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % generation of pulse-width waveform         %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%% find first pulse width of the 3 phases
   A1=theta+phi;
   B1=A1+2*pi/3;
   C1=A1-2*pi/3;
   up_a=(tc/4)*(1-M*sin(A1));
   up_b=(tc/4)*(1-M*sin(B1));
   up_c=(tc/4)*(1-M*sin(C1));
   down_a=(tc/4)*(1+M*sin(A1));
   down_b=(tc/4)*(1+M*sin(B1));
   down_c=(tc/4)*(1+M*sin(C1));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %  Generation of 3-phase PWM waveforms    %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for k=1:s
     if k < (s/2)
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %  pulse widths of 3-phases when carrier wave is at +ve gradient
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       tt=dt*k;
       %%%%%%%%%%%%%%
       %  phase a  %
       %%%%%%%%%%%%%%
       if tt  up_a
         pwma(p)=Vdc;
       else
         pwma(p)=-Vdc;
       end,
       %%%%%%%%%%%%%%
       %  phase b  %
       %%%%%%%%%%%%%%
       if tt  up_b
         pwmb(p)=Vdc;
       else
       pwmb(p)=-Vdc;
       end,
       %%%%%%%%%%%%%%
       %  phase c  %
       %%%%%%%%%%%%%%
       if tt  up_c
         pwmc(p)=Vdc;
       else
         pwmc(p)=-Vdc;
       end,
     else
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %  pulse widths of 3-phases when carrier wave is at -ve gradient
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       tt=dt*k;
       if tt  down_a
         pwma(p)=-Vdc;
       else
         pwma(p)=Vdc;
       end,
       if tt  down_b
         pwmb(p)=-Vdc;
       else
         pwmb(p)=Vdc;
       end,
       if tt  down_c
         pwmc(p)=-Vdc;
       else
         pwmc(p)=Vdc;
       end,

           %%% d-q transformation again for the input to model %%%
           C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
```

```
            u=C*[pwma(p) pwmb(p) pwmc(p)]';u=u';
            uds(p)=u(1);uqs(p)=u(2);

            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Machine model is based on transformation referred to stator.                 %
            % Formation of inductance, resistance and rotational matrice.  %
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Change to matrice suitable for standard State-variable form
            A=-L_*(Wr*G+Res);
            B=L_;
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            % Digitalising the State Equation....%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            [PHI,GAM]=c2d(A,B,dt);
            x1=PHI*x0 + GAM*[uds(p) uqs(p) 0 0]';
            Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
            k0=dt*(2/J)*(Te-Kf*Wr); % the factor 2 is due to no. of pole pair
            k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
            k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
            k3=dt*(2/J)*(Te-Kf*(Wr+k2));
            Wr=Wr+k0/6+k1/3+k2/3+k3/6;
            Wr_=[Wr_;Wr];
            Te_=[Te_;Te];
            x0=x1;
            dT=Te-Kf*Wr;
            %time = time + dt;  % update time
            percent=p*100/N;
            %%%% display values %%%%
 disp('  percent     M      up_a     down_a  pwma(p)       Te        Wr');
 disp([percent M up_a down_a pwma(p) Te Wr]);
     end,
     p=p+1;
   end,
end
```

Appendix B-8 : Vector control with PWM supply (extension program)

```
clc
echo on
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program continues the program pwmvec_.
% supplied by a PWM  power source.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
echo off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%            load data from vecdata.mat file : including
% currents, angular velocity, angular position and torque
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

load vecdata
% clear Te_,Wr_,pwma,pwmb,pwmc,uds,uqs
Te_=[];Wr_=[];pwma=[];pwmb=[];pwmc=[];uds=[];uqs=[];

            %%%%%%%%%%%%%%%%%%%%%%%%
            % Simulation begins %
            %%%%%%%%%%%%%%%%%%%%%%%%
p=1;
n=N/s;
for i=1:n;
   %%%%%%%%%%%%%%%%%%%%%%%%%
   %         Vector control
   %%%%%%%%%%%%%%%%%%%%%%%%%
   iqs=(Wrref-Wr)*gain;
   Vds=ids*Rs - iqs*sigma*Ls*We;
   Vqs=ids*Ls*We + iqs*Rs;
   Wsl=iqs/(Tr*ids);
   We=Wsl+Wr;
   theta=theta + We*(dt*s);
   Vm=sqrt(Vds^2+Vqs^2);
   phi=atan2(Vqs,Vds);
   M=Vm/(Vdc*0.78);
   %% set limit on M1
   if M1;M=1;
     disp('M  1');
   end,
   f=We/(2*pi);
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   % generation of pulse-width waveform         %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %%% find first pulse width of the 3 phases
   A1=theta+phi;
   B1=A1+2*pi/3;
   C1=A1-2*pi/3;
   up_a=(tc/4)*(1-M*sin(A1));
   up_b=(tc/4)*(1-M*sin(B1));
   up_c=(tc/4)*(1-M*sin(C1));
   down_a=(tc/4)*(1+M*sin(A1));
   down_b=(tc/4)*(1+M*sin(B1));
   down_c=(tc/4)*(1+M*sin(C1));
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   %  Generation of 3-phase PWM waveforms    %
   %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
   for k=1:s
     if k < (s/2)
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %  pulse widths of 3-phases when carrier wave is at -ve gradient
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       tt=dt*k;
       if tt  down_a
         pwma(p)=-Vdc;
       else
         pwma(p)=Vdc;
       end,
       if tt  down_b
         pwmb(p)=-Vdc;
       else
         pwmb(p)=Vdc;
       end,
       if tt  down_c
         pwmc(p)=-Vdc;
       else
         pwmc(p)=Vdc;
       end,
     else
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       %  pulse widths of 3-phases when carrier wave is at +ve gradient
       %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
       tt=dt*(k-s/2);
       %%%%%%%%%%%%%%
       %  phase a  %
       %%%%%%%%%%%%%%
       if tt  up_a
         pwma(p)=Vdc;
       else
         pwma(p)=-Vdc;
       end,
       %%%%%%%%%%%%%%
       %  phase b  %
       %%%%%%%%%%%%%%
       if tt  up_b
         pwmb(p)=Vdc;
       else
         pwmb(p)=-Vdc;
       end,
       %%%%%%%%%%%%%%
       %  phase c  %
       %%%%%%%%%%%%%%
       if tt  up_c
         pwmc(p)=Vdc;
       else
         pwmc(p)=-Vdc;
       end,
     end,
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     %%% d-q transformation again for the input to model %%%
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     C=(2/3)*[1 -1/2 -1/2;0 -sqrt(3)/2 sqrt(3)/2;1/2 1/2 1/2];
     u=C*[pwma(p) pwmb(p) pwmc(p)]';u=u';
     uds(p)=u(1);uqs(p)=u(2);
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Machine model is based on transformation referred to stator.       %
     % Formation of inductance, resistance and rotational matrice.   %
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Change to matrice suitable for standard State-variable form
     A=-L_*(Wr*G+Res);
     B=L_;
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     % Digitalising the State Equation....%
     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
     [PHI,GAM]=c2d(A,B,dt);
     x1=PHI*x0 + GAM*[uds(p) uqs(p) 0 0]';
     Te=3*Lm*(x1(3)*x1(2)-x1(1)*x1(4));
     k0=dt*(2/J)*(Te-Kf*Wr); % the factor 2 is due to no. of pole pair
     k1=dt*(2/J)*(Te-Kf*(Wr+k0/2));
     k2=dt*(2/J)*(Te-Kf*(Wr+k1/2));
     k3=dt*(2/J)*(Te-Kf*(Wr+k2));
     Wr=Wr+k0/6+k1/3+k2/3+k3/6;
     Wr_=[Wr_;Wr];
     Te_=[Te_;Te];
     x0=x1;
     dT=Te-Kf*Wr;
     percent=p*100/N;
     %%%% display values %%%%
     if rem(p,8)==0
       clc
     else
     end
 disp('   percent     M       up_a     down_a  pwma(p)       Te        Wr');
 disp([percent M up_a down_a pwma(p) Te Wr]);
     p=p+1;
   end,
end

save vecdata
```

```
fm = input('input modulating frequency (Hz):');
M = input('input modulation index :');
r = input('input frequency ratio :');
N = input('no. of data points (power of 2) :');
E = input('No. of harmonics to be shown:');
R = round(r);
Tm = 1/fm;
fc = fm*r;
Tc = 1/fc;
fs = N*fm;
Ts = 1/fs;

echo on

% Computing pulse widths for the phase A

echo off

sum=0;
for n=1:2*R+1;
 sum=sum + Tc/2*(1+((-1)^(n-1))*(M/2)*(sin(pi*(n-1)/R)+sin(pi*n/R)));
 Tsum(n)=sum;
end

for n=1:2*R+1;
 T(n+1)=Tsum(n)+Tc/4;
end
T(1)=Tc/4;

echo on;

% Sampling of the generated PWM -- phase A

echo off;

S = 0:Ts:N*Ts;

i=2;j=1;
ha(1)=-1;

while i
  if S(i)< T(j)
    ha(i)=ha(i-1);
    i=i+1;
  else
    ha(i)=(-1)*ha(i-1);
    i=i+1;
    j=j+1;
  end
end

S=[];

echo on;

% phase B & C found by shifting 120 and 240 elect. degree from that of A

echo off;

N1_3=fix(N/3);
N2_3=2*N1_3;

for j=1:N1_3
  hb(j)=ha(N2_3+j);
end

for j=N1_3+1:N;
  hb(j)=ha(j-N1_3);
end

for m=1:N
  hab(m)=ha(m)-hb(m);
end;

echo on

%performing FFT

echo off

Ha=fft(ha);
Ma=abs(Ha);Ha=[];
Mnom=Ma/max(Ma);  Ma=[];

hold off;
t=Tm/N:Tm/N:Tm;


freq=0:1:N;
hold off;
axis([0 freq(E+1) 0 1.2]);
plot([freq(1) freq(1)],[0 Mnom(1)],'g');
hold on;
plot([freq(2) freq(2)],[0 Mnom(2)]);
for v=3:E
  if v==R+1
    plot([freq(v) freq(v)],[0 Mnom(v)],'w')
  else
    plot([freq(v) freq(v)],[0 Mnom(v)],'g')
  end
end;
end
```

```
fm = input('input modulating frequency (Hz):');
M = input('input modulation index :');
r = input('input frequency ratio :');
N = input('no. of data points (power of 2) :');
E = input('No. of harmonics to be shown:');
R = round(r);
Tm = 1/fm;
fc = fm*r;
Tc = 1/fc;
fs = N*fm;
Ts = 1/fs;

echo on

% Computing pulse widths for the phase A

echo off

sum=0;
for n=1:2*R+1;
 sum=sum + Tc/2*(1+((-1)^(n-1))*(M/2)*(sin(pi*(n-1)/R)+sin(pi*n/R)));
 Tsum(n)=sum;
end

for n=1:2*R+1;
 T(n+1)=Tsum(n)+Tc/4;
end
T(1)=Tc/4;

echo on;

% Sampling of the generated PWM -- phase A

echo off;

S = 0:Ts:N*Ts;

i=2;j=1;
ha(1)=-1;

while i
  if S(i) < T(j)
     ha(i)=ha(i-1);
     i=i+1;
  else
     ha(i)=(-1)*ha(i-1);
     i=i+1;
     j=j+1;
  end
end

S=[];

echo on;

% phase B & C found by shifting 120 and 240 elect. degree from that of A

echo off;

N1_3=fix(N/3);
N2_3=2*N1_3;

for j=1:N1_3
  hb(j)=ha(N2_3+j);
end

for j=N1_3+1:N;
  hb(j)=ha(j-N1_3);
end

for m=1:N
  hab(m)=ha(m)-hb(m);
end;

echo on

%performing FFT

echo off

Hab=fft(hab);
Mab=abs(Hab);Hab=[];
Mnom=Mab/max(Mab);  Mab=[];

hold off;
t=Tm/N:Tm/N:Tm;


freq=0:1:N;
hold off;
axis([0 freq(E+1) 0 1.2]);
plot([freq(1) freq(1)],[0 Mnom(1)],'g');
hold on;
plot([freq(2) freq(2)],[0 Mnom(2)]);
for v=3:E
  if v==R+1
     plot([freq(v) freq(v)],[0 Mnom(v)],'w')
  else
     plot([freq(v) freq(v)],[0 Mnom(v)],'g')
  end
end
```

```
PROC host ()
 PROTOCOL MOTOR
  CASE
   -- On-line command tags
   vector.control.tag
   change.reference.flux.tag
   change.reference.speed.tag
   change.carrier.freq.tag
   change.pwm.parameter.tag
   forward.tag
   reverse.tag
   dynamic.brake.tag
   -- operational tags
   full.speed.tag
   acknowledge.tag
   finish.tag
   -- machine.data ; data format : r1,r2,L1,L2,Lo,M
   machine.data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32
   -- startup data ; data format : f,m0,m,N
   start.up.data;REAL32;REAL32;REAL32;INT
   -- control data ; data format ; Vdc,Fdr,controller.gain
   control.data;REAL32;REAL32;REAL32
   -- On-line command channels
   speed.ref;REAL32
   vector.control.parameter;INT;INT;REAL32 -- data format: phi,R,m
   pwm.parameter;INT;REAL32                 -- data format: R,m
   flux,ref;REAL32
   carrier.freq;REAL32                -- (not used yet)
   -- feedback channels
   speed.measure;REAL32
   control.parameter;INT;REAL32;REAL32   -- phi,m,Wsl
   position;REAL32                    -- (not used yet)
   link.current;REAL32                 -- DC link current is read
 :
 #USE userio
 #USE userhdr
 CHAN OF MOTOR host.to.control,control.to.host:
 VAL link0in IS 4 :
 VAL link0out IS 0 :
 VAL link1in IS 5 :
 VAL link1out IS 1 :
 VAL link2in IS 6 :
 VAL link2out IS 2 :
 VAL link3in IS 7 :
 VAL link3out IS 3 :
 PLACE control.to.host AT link2in:
 PLACE host.to.control AT link2out:
 -- The no. of variables used in the host transputer is less than 4K.
 -- Therefore, the variables can be declared globally without affecting
 -- the performance of system.
 REAL32 f,fc,r,m,m0,Wr,Wr.ref.rpm,Tc,Tc.4,flux2,Wd:
 INT N,R,Tc.2,code,char,any,time.now,i:
 BOOL cycling,check.Wr.ref,check.m0,check.m,check.N,get.display.data:
 BOOL dynamic.brake:
 TIMER clock:
 REAL32 Vdc:
 REAL32 We,Wsl,Wr.rpm,Wr.elec:
 REAL32 T.ref,Fdr,ids,iqs,idr,iqr,idc,Vds,Vqs:
 REAL32 controller.gain:
 REAL32 Vm,phi.angle:
 REAL32 m.buffer:
 INT phi:
 BOOL check.Fdr,check.T.ref,check.controller.gain,check.Vdc,vector.control.flag:
 SEQ
  screen ! tt.goto; INT 15;INT 1
  write.text.line(screen," TRANSPUTER VECTOR CONTROL OF INDUCTION MOTOR (Ver.8)")
  screen ! tt.goto; INT 15;INT 2
  write.text.line(screen," ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~")
  screen ! tt.goto; INT 11;INT 3
  write.full.string(screen,"Reference speed :")
  screen ! tt.goto; INT 37;INT 3
  write.text.line(screen,"(rpm) ")
  screen ! tt.goto; INT 50;INT 3
  write.text.line(screen,"Vdc(V) :  xx.x   (V)")
  screen ! tt.goto; INT 11;INT 4
  write.text.line(screen,"Ref. rotor flux  :")
  screen ! tt.goto; INT 37;INT 4
  write.text.line(screen,"(Wb)")
  screen ! tt.goto; INT 50;INT 4
  write.text.line(screen,"Vector control : xx")
  screen ! tt.goto; INT 11;INT 5
  write.text.line(screen,"No. of steps for soft starting (1 step=0.25s) :")
  screen ! tt.goto; INT 11;INT 6
  write.text.line(screen,"Voltage boost (per unit of DC link voltage) : x.x ")
  screen ! tt.goto; INT 11;INT 7
  write.text.line(screen,"Modulation index : x.x ")
  screen ! tt.goto; INT 11;INT 8
  write.text.line(screen,"Controller gain  : x.x ")
  screen ! tt.goto; INT 11;INT 9
  write.text.line(screen,"PWM parameters:")
  screen ! tt.goto; INT 11;INT 10
  write.text.line(screen,"Carrier freq = xxx.x  (Hz)")
  screen ! tt.goto; INT 40;INT 10
  write.text.line(screen,"Mod. index = xx.x ")
  screen ! tt.goto; INT 11;INT 11
  write.text.line(screen,"Modulating freq = xxx.x (Hz)")
  screen ! tt.goto; INT 40;INT 11
  write.text.line(screen,"Freq ratio = xx.x ")
  screen ! tt.goto; INT 11;INT 13
  write.text.line(screen,"Measured speed :   xxxx  (rpm)")
  screen ! tt.goto; INT 50;INT 13
  write.text.line(screen,"DC link current : xxx   (A)")
  screen ! tt.goto; INT 11;INT 14
  write.text.line(screen,"Speed difference:  xxxx  (rpm)")
  screen ! tt.goto; INT 11;INT 16
  write.text.line(screen,"Torque angle =")
  screen ! tt.goto; INT 33;INT 16
  write.text.line(screen,"m = ")
  screen ! tt.goto; INT 44;INT 16
  write.text.line(screen,"Wsl =")
  screen ! tt.goto; INT 11;INT 18
  write.text.line(screen,"COMMAND")
  screen ! tt.goto; INT 55;INT 18
  write.text.line(screen,"ACTION")
  screen ! tt.goto; INT 11;INT 19
  write.text.line(screen,"~~~~~~~")
  screen ! tt.goto; INT 55;INT 19
  write.text.line(screen,"~~~~~~")
  screen ! tt.goto; INT 11;INT 20
  write.text.line(screen,"Type s for starting ")
  SEQ
   fc:=500.0(REAL32)
   check.Wr.ref:=TRUE
   WHILE check.Wr.ref
    SEQ
     screen ! tt.goto; INT 29;INT 3
     char:=0
     read.echo.real32(keyboard,screen,Wr.ref.rpm,char)
     IF
      (Wr.ref.rpm < (60.0(REAL32))) OR (Wr.ref.rpm > (960.0(REAL32)))
       SEQ
        screen ! tt.beep
        screen ! tt.beep
        screen ! tt.goto; INT 29;INT 3
        write.full.string(screen,"xxxxx ")
      (Wr.ref.rpm > (60.0(REAL32))) AND (Wr.ref.rpm < (960.0(REAL32)))
       check.Wr.ref:=FALSE
   f:=Wr.ref.rpm/(30.0(REAL32))  -- pole-pair is 2, f in Hz
   check.Vdc:=TRUE
   WHILE check.Vdc
    SEQ
     screen ! tt.goto; INT 60;INT 3
     char:=0
     read.echo.real32(keyboard,screen,Vdc,char)
     IF
      (Vdc <= (10.0(REAL32))) OR (Vdc > (150.0(REAL32)))
       SEQ
        screen ! tt.beep
        screen ! tt.beep
```

```
        screen ! tt.goto; INT 60;INT 3
        write.full.string(screen,"xxxx")
      (Vdc > (10.0(REAL32))) AND (Vdc < (150.0(REAL32)))
        check.Vdc:=FALSE
  check.Fdr:=TRUE
  char:=0
  WHILE check.Fdr
    SEQ
      screen ! tt.goto; INT 31;INT 4
      char:=0
      read.echo.real32(keyboard,screen,Fdr,char)
      IF
        (Fdr <= (0.1(REAL32))) OR (Fdr >= (2.0(REAL32)))
          SEQ
            screen ! tt.beep
            screen ! tt.goto; INT 31;INT 4
            write.full.string(screen,"x.xx")
        TRUE -- (Fdr > (0.0(REAL32))) AND (Fdr < (1.0(REAL32)))
          check.Fdr:=FALSE
  check.N:=TRUE
  WHILE check.N
    SEQ
      screen ! tt.goto; INT 58;INT 5
      char:=0
      read.echo.int(keyboard,screen,N,char)
      IF
        (N < 10) OR (N > 1000)
          SEQ
            screen ! tt.beep
            screen ! tt.beep
            screen ! tt.goto; INT 58;INT 5
            write.full.string(screen,"xx")
        (N >= 10) AND (N <= 1000)
          check.N:=FALSE
  check.m0:=TRUE
  WHILE check.m0
    SEQ
      screen ! tt.goto; INT 57;INT 6
      read.echo.real32(keyboard,screen,m0,char)
      IF
        (m0 >= (0.5(REAL32)))
          SEQ
            screen ! tt.beep
            screen ! tt.goto; INT 57;INT 6
            write.full.string(screen,"x.x")
        (m0 <= 0.0(REAL32))
          SEQ
            screen ! tt.beep
            screen ! tt.goto; INT 57;INT 6
            write.full.string(screen,"x.x")
        TRUE -- (m0>(0.0(REAL32)))AND(m<(0.5(REAL32)))
          SEQ
            check.m0:=FALSE
  check.m:=TRUE
  WHILE check.m
    SEQ
      screen ! tt.goto; INT 30;INT 7
      read.echo.real32(keyboard,screen,m,char)
      IF
        (m >= (1.0(REAL32)))
          SEQ
            screen ! tt.beep
            screen ! tt.goto; INT 30;INT 7
            write.full.string(screen,"x.x")
        (m <= m0)
          SEQ
            screen ! tt.beep
            screen ! tt.goto; INT 30;INT 7
            write.full.string(screen,"x.x")
        (m>m0)AND(m<(1.0(REAL32)))
          SEQ
            check.m:=FALSE
  check.controller.gain:=TRUE
  WHILE check.controller.gain
    SEQ
      screen ! tt.goto; INT 30;INT 8
      char:=0
      read.echo.real32(keyboard,screen,controller.gain,char)
      IF
        (controller.gain<(0.01(REAL32))) OR (controller.gain>(5.0(REAL32)))
          SEQ
            screen ! tt.beep
            screen ! tt.beep
            screen ! tt.goto; INT 30;INT 8
            write.full.string(screen,"xx")
        TRUE
          check.controller.gain:=FALSE
host.to.control ! start.up.data;f;m0;m;N
host.to.control ! control.data;Vdc;Fdr;controller.gain
screen ! tt.goto; INT 45;INT 20
write.text.line(screen," Sending start up data to network")
screen ! tt.goto; INT 45;INT 21
control.to.host ? CASE acknowledge.tag  -- acknowledge timer setup
screen ! tt.goto; INT 11;INT 20
write.text.line(screen,"Type s for starting                ")
SEQ
  cycling:=TRUE
  WHILE cycling
    SEQ
      keyboard ? code
      IF
        (code=115) OR (code=83)
          SEQ
            host.to.control ! forward.tag
            screen ! tt.goto; INT 45;INT 20
            write.text.line(screen,"Motor is accelerating ....        ")
            screen ! tt.goto; INT 45;INT 21
            write.text.line(screen,"Host is waiting for full.speed.tag")
            cycling:=FALSE
        TRUE
          SEQ
            screen ! tt.beep
            screen ! tt.beep
screen ! tt.goto; INT 11;INT 22
write.text.line(screen,"Type q for Quit system ")
control.to.host ? CASE full.speed.tag
screen ! tt.beep
screen ! tt.goto; INT 45;INT 20
write.text.line(screen,"Motor is running at full speed     ")
screen ! tt.goto; INT 45;INT 21
write.text.line(screen,"                                   ")
SEQ
  vector.control.flag:=FALSE
  cycling:=TRUE
  i:=0
  phi:=0
  m.buffer:=0.0(REAL32)
  Wsl:=0.0(REAL32)
  WHILE cycling
    PRI ALT
      keyboard ? code
      IF
        code=45   -- decrease m by 1%
          SEQ
            m:=m-0.05(REAL32)
            IF
              m<0.01(REAL32)
                m:=0.01(REAL32)       -- impose lower limit
              TRUE
                SKIP
            host.to.control ! change.pwm.parameter.tag
            host.to.control ! pwm.parameter;R;m
        code=43   -- increase m by 1%
          SEQ
            m:=m+0.05(REAL32)
            IF
              m>1.0(REAL32)
                m:=1.0(REAL32)       -- impose upper limit
              TRUE
```

```
            SKIP
          host.to.control ! change.pwm.parameter.tag
          host.to.control ! pwm.parameter;R;m
        code=202  -- decrease reference speed by 60 rpm
          SEQ
            Wr.ref.rpm:=Wr.ref.rpm-240.0(REAL32)
            IF
              Wr.ref.rpm<60.0(REAL32)
                Wr.ref.rpm:=60.0(REAL32)      -- impose lower limit
              TRUE
                SEQ
                  m:=m-0.06(REAL32)
                  IF
                    m<0.01(REAL32)
                      m:=0.01(REAL32)
                    TRUE
                      SKIP
            f:=Wr.ref.rpm/(30.0(REAL32))      -- Wr in rpm, 2 pole-pair
            screen ! tt.goto; INT 29;INT 3
            write.real32(screen,Wr.ref.rpm,4,1)
            r:=fc/f
            R:=INT ROUND (r)
            host.to.control ! change.pwm.parameter.tag
            host.to.control ! pwm.parameter;R;m
        code=201  -- increase reference speed by 5 rpm
          SEQ
            Wr.ref.rpm:=Wr.ref.rpm+240.0(REAL32)
            IF
              Wr.ref.rpm>960.0(REAL32)
                Wr.ref.rpm:=960.0(REAL32)  -- impose upper limit
              TRUE
                SEQ
                  m:=m+0.06(REAL32)
                  IF
                    m>1.0(REAL32)
                      m:=1.0(REAL32)
                    TRUE
                      SKIP
            f:=Wr.ref.rpm/(30.0(REAL32))      -- wr in rpm, 2 pole-pair
            screen ! tt.goto; INT 29;INT 3
            write.real32(screen,Wr.ref.rpm,4,1)
            r:=fc/f
            R:=INT ROUND (r)
            host.to.control ! change.pwm.parameter.tag
            host.to.control ! pwm.parameter;R;m
        code=203
          SEQ
            screen ! tt.goto; INT 45;INT 21
            write.text.line(screen,"Decrease flux by 0.05 Wb          ")
            Fdr:=Fdr-0.05(REAL32)
            screen ! tt.goto; INT 29;INT 4
            write.real32(screen,Fdr,4,2)
            host.to.control ! change.reference.flux.tag
            host.to.control ! flux.ref;Fdr
        code=204
          SEQ
            screen ! tt.goto; INT 45;INT 21
            write.text.line(screen,"Increase flux by 0.05 Wb          ")
            Fdr:=Fdr+0.05(REAL32)
            screen ! tt.goto; INT 29;INT 4
            write.real32(screen,Fdr,4,2)
            host.to.control ! change.reference.flux.tag
            host.to.control ! flux.ref;Fdr
        code=118
          SEQ
            vector.control.flag:=NOT(vector.control.flag)
            screen ! tt.goto; INT 66;INT 4
            IF
              (vector.control.flag)
                SEQ
                  host.to.control!vector.control.tag
                  write.text.line(screen," ON ")
              TRUE
                SEQ
                  host.to.control!vector.control.tag
                  write.text.line(screen," OFF ")
        (code=81)OR(code=113)  -- 'q' or 'Q' for Quit
          SEQ
            screen ! tt.goto; INT 45;INT 20
            write.text.line(screen," Quit system ...                ")
            screen ! tt.goto; INT 45;INT 21
            write.text.line(screen," Send finish tag to network  ... ")
            host.to.control ! finish.tag
            control.to.host ? CASE acknowledge.tag -- wait [pwm] reply
            cycling:=FALSE
        TRUE
          SEQ
            screen ! tt.beep
            screen ! tt.beep
            screen ! tt.goto; INT 45;INT 20
            write.text.line(screen,"Type Q for quiting the system !   ")
            screen ! tt.goto; INT 45;INT 21
            write.text.line(screen,"                                  ")
      TRUE & SKIP
        --control.to.host ? CASE link.current;idc  -- to be implemented
        SEQ
          control.to.host ? CASE speed.measure;Wr.elec
          IF
            (vector.control.flag)
              control.to.host ? CASE control.parameter;phi;m.buffer;Wsl
            TRUE
              SKIP
          SEQ
            i:=i+1
            IF
              (i<100)  -- display new data every 0.5 second
                SKIP
              TRUE
                SEQ
                  Wr.rpm:=Wr.elec*4.77465(REAL32)
                  Wd:=(Wr.ref.rpm-Wr.rpm)
                  SEQ
                    screen ! tt.goto; INT 28;INT 13
                    write.real32(screen,Wr.rpm,3,1)
                    screen ! tt.goto; INT 26;INT 10
                    write.real32(screen,fc,3,1)
                    screen ! tt.goto; INT 29;INT 11
                    write.real32(screen,f,2,1)
                    screen ! tt.goto; INT 53;INT 10
                    write.real32(screen,m.buffer,3,2)
                    --screen ! tt.goto; INT 53;INT 11
                    --write.real32(screen,r,2,1)
                    screen ! tt.goto; INT 28;INT 14
                    write.real32(screen,Wd,3,1)
                    screen ! tt.goto; INT 26;INT 16
                    phi:=(phi*180)/1000
                    write.int(screen,phi,4)
                    screen ! tt.goto; INT 36;INT 16
                    write.real32(screen,m.buffer,1,1)
                    screen ! tt.goto; INT 50;INT 16
                    write.real32(screen,Wsl,3,1)
                  i:=0
    screen ! tt.goto; INT 66;INT 23
    write.text.line(screen,"Strike a key")
    keyboard ? any
:
host ()
```

```
PROTOCOL MOTOR
  CASE
    -- On-line command tags
    vector.control.tag
    change.reference.flux.tag
    change.reference.speed.tag
    change.carrier.freq.tag
    change.pwm.parameter.tag
    forward.tag
    reverse.tag
    dynamic.brake.tag
    -- operational tags
    full.speed.tag
    acknowledge.tag
    finish.tag
    -- machine.data ; data format : r1,r2,L1,L2,Lo,M
    machine.data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32
    -- startup data ; data format : f,m0,m,N
    start.up.data;REAL32;REAL32;REAL32;INT
    -- control data ; data format : Vdc,Fdr,controller.gain
    control.data;REAL32;REAL32;REAL32
    -- On-line command channels
    speed.ref;REAL32
    vector.control.parameter;INT;INT;REAL32 -- data format: phi,R,m
    pwm.parameter;INT;REAL32                -- data format: R,m
    flux.ref;REAL32
    carrier.freq;REAL32                     -- (not used yet)
    -- feedback channels
    speed.measure;REAL32
    control.parameter;INT;REAL32;REAL32     -- phi,m,Wsl
    position;REAL32                         -- (not used yet)
    link.current;REAL32                     -- DC link current is read
:
PROC control(CHAN OF MOTOR control.to.host,host.to.control,
    control.to.pwm,pwm.to.control,control.to.current,current.to.control)
  REAL32 f,fc,m,m0,r:
  REAL32 We,Wsl,Wr.elec:
  REAL32 ls,lr,Ls,Lr,Lo,Rs,Rr,Lm,Tr:
  REAL32 Wr.ref.elec,T.ref,ids,iqs,idr,iqr,Vds,Vqs,controller.gain,idc:
  REAL32 Fdr:
  REAL32 Vm,phi.angle:
  REAL32 Vdc:
  REAL32 PI2:
  REAL32 phi.angle:
  INT phi:
  INT N,R:
  BOOL cycling,vector.control.flag:
  #USE userio
  #USE t4math
  #USE mathvals
  SEQ
    Rs:=3.76(REAL32)
    Rr:=2.57](REAL32)
    Lm:=0.268(REAL32)
    ls:=0.01165(REAL32)
    lr:=0.02790(REAL32)
    Ls:=Lm+ls
    Lr:=Lm+lr
    Lo:=Ls-((Lm*Lm)/Lr)
    Tr:=Lr/Rr
    PI2:=PI*2.0(REAL32)
    host.to.control ? CASE start.up.data;f;m0;m;N
    control.to.pwm ! start.up.data;f;m0;m;N
    host.to.control ? CASE control.data;Vdc;Fdr;controller.gain
    Wr.ref.elec:=((2.0(REAL32))*PI)*f
    fc:=500.0(REAL32)        -- fixed carrier freq. used
    r:=fc/f
    R:=INT ROUND (r)
    pwm.to.control ? CASE acknowledge.tag  -- timer already setup signal
    control.to.host ! acknowledge.tag
    host.to.control ? CASE forward.tag
    control.to.pwm ! forward.tag
    pwm.to.control ? CASE full.speed.tag
    control.to.host ! full.speed.tag
    current.to.control ? CASE speed.measure;Wr.elec  -- newly added
    vector.control.flag:=FALSE
    cycling:=TRUE
    WHILE cycling
      PRI ALT
        host.to.control ? CASE
          vector.control.tag
            vector.control.flag:=NOT(vector.control.flag)
          change.pwm.parameter.tag
            SEQ
              host.to.control ? CASE pwm.parameter;R;m
              Wr.ref.elec:=(((2.0(REAL32))*PI)*fc)/((REAL32 ROUND (R)))--w=2pif
              control.to.pwm ! change.pwm.parameter.tag
              control.to.pwm ! pwm.parameter;R;m
          change.reference.flux.tag
            host.to.control ? CASE flux.ref;Fdr
          finish.tag
            SEQ
              control.to.pwm ! finish.tag
              pwm.to.control ? CASE acknowledge.tag
              control.to.current ! finish.tag
              current.to.control ? CASE acknowledge.tag
              control.to.host ! acknowledge.tag
              cycling:=FALSE
        current.to.control ? CASE speed.measure;Wr.elec
          -- the current is sampled and measured at the same rate as the speed
          --current.to.control ? CASE link.current;idc --to be implemented later
          REAL32 m.buffer
          SEQ
            control.to.host ! speed.measure;Wr.elec
            IF
              (vector.control.flag)
                SEQ
                  --The forcing equations for the vector control are to be performed at
                  --a sampling interval (1ms by Leonard) which is generated by the
                  --host transputer. Note that the angular speeds derived from vector control
                  --are in elec. rad/s.
                  T.ref:=(Wr.ref.elec-Wr.elec)*controller.gain
                  ids:=Fdr/Lm
                  iqs:=(T.ref*Lr)/(Lm*Fdr)
                  Wsl:=iqs/(Tr*ids)          -- Wsl in elec. rad/s
                  We:=Wsl+Wr.elec            -- We in elec. rad/s
                  f:=We/PI2                  -- f=We/2*pi
                  R:=INT ROUND (fc/f)
                  Vds:=(ids*Rs)-((iqs*Lo)*We)
                  Vqs:=((ids*Ls)*We)+(iqs*Rs)
                  Vm:=SQRT((Vds*Vds)+(Vqs*Vqs))
                  m.buffer:=Vm/Vdc
                  --1000 samples over PI (radians
                  phi.angle:=ABS(ATAN(Vqs/Vds))
                  phi:= INT ROUND ((phi.angle*(1000.0(REAL32)))/(PI)
                  control.to.host ! control.parameter;phi;m.buffer;Wsl
                  --phi:=0    --***** changed ??
                  IF
                    (m.buffer>0.95(REAL32))
                      m.buffer:=0.95(REAL32)
                    (m.buffer<0.0(REAL32))
                      m.buffer:=m0
                    TRUE
                      SKIP
                  control.to.pwm ! vector.control.tag
                  control.to.pwm ! vector.control.parameter;phi;R;m.buffer
              TRUE
                SKIP
```

```
PROTOCOL MOTOR
  CASE
    -- On-line command tags
    vector.control.tag
    change.reference.flux.tag
    change.reference.speed.tag
    change.carrier.freq.tag
    change.pwm.parameter.tag
    forward.tag
    reverse.tag
    dynamic.brake.tag
    -- operational tags
    full.speed.tag
    acknowledge.tag
    finish.tag
    -- machine.data ; data format : r1,r2,L1,L2,Lo,M
    machine.data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32
    -- startup data ; data format : f,m0,m,N
    start.up.data;REAL32;REAL32;REAL32;INT
    -- control data ; data format : Vdc,Fdr,controller.gain
    control.data;REAL32;REAL32;REAL32
    -- On-line command channels
    speed.ref;REAL32
    vector.control.parameter;INT;INT;REAL32 -- data format: phi,R,m
    pwm.parameter;INT;REAL32                -- data format: R,m
    flux.ref;REAL32
    carrier.freq;REAL32                     -- (not used yet)
    -- feedback channels
    speed.measure;REAL32
    control.parameter;INT;REAL32;REAL32     -- phi,m,Wsl
    position;REAL32                         -- (not used yet)
    link.current;REAL32                     -- DC link current is read
:
PROC pwm (CHAN OF MOTOR pwm.to.control,control.to.pwm)
  #USE userio
  #USE mathvals
  #USE t4math
  CHAN OF BYTE to.timer.data.chan:
  CHAN OF BYTE from.timer.data.chan:
  CHAN OF BYTE to.timer.control.chan:
  CHAN OF BYTE from.timer.control.chan:
  REAL32 sinA,sinB,sinC,Tc,f,fc,r,m,m0,Tc.4,Wr:
  REAL32 Tc.ramp,Tc.2.ramp,Tc.4.ramp,f.ramp,m.ramp,f.step,m.step,f0,fc:
  [5000]REAL32 s:
  INT tempA,tempB,tempC:
  INT ta,tb,tc,ta.hi,ta.lo,tb.hi,tb.lo,tc.hi,tc.lo:
  INT any.char,time.now,Tc.2,i,R,R.2:
  INT soft.start.time,soft.start.time.ref:
  INT j,j.k,N:
  BOOL cycling:
  TIMER clock:
  INT soft.start.time:
  INT phi:
  REAL32 Wc,Vdc:
  VAL link0in IS 4 :
  VAL link0out IS 0 :
  VAL link1in IS 5 :
  VAL link1out IS 1 :
  PLACE to.timer.data.chan AT 0:
  PLACE from.timer.data.chan AT 4:
  PLACE to.timer.control.chan AT 1:
  PLACE from.timer.control.chan AT 5:
  SEQ
    SEQ
      j:=1
      SEQ i=1 FOR 501
        SEQ
          s[j]:=SIN(((0.001(REAL32))*PI)*(REAL32 ROUND (i-1)))
      WHILE (j<=500)
        SEQ
          s[501+j]:=s[501-j]
          j:=j+1
      k:=1
      WHILE (k<=1000)
        SEQ
          s[1001+k]:=-s[k+1]
          s[2001+k]:=s[k+1]
          s[3001+k]:=-s[k+1]
          k:=k+1
      k:=1
      WHILE (k<995)
        SEQ
          s[4001+k]:=s[k+1]
          k:=k+1
    control.to.pwm ? CASE start.up.data;f;m0;m;N
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 50 -- (Timer0,Mode1,2-byte,binary)
    to.timer.control.chan ! BYTE 235
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- (initialise timer0, first byte)
    to.timer.control.chan ! BYTE 232  --timer0
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- (initialise timer0, last byte)
    to.timer.control.chan ! BYTE 232  --timer0
    to.timer.control.chan ! BYTE 255
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 114 -- (Timer1,Mode1,2-byte,binary)
    to.timer.control.chan ! BYTE 235
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- initialise timer1
    to.timer.control.chan ! BYTE 233
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- initialise timer1
    to.timer.control.chan ! BYTE 233
    to.timer.control.chan ! BYTE 255
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 178 -- (Timer2,Mode1,2-byte,binary)
    to.timer.control.chan ! BYTE 235  --control
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- initialise timer2 (first byte)
    to.timer.control.chan ! BYTE 234
    to.timer.control.chan ! BYTE 255
    to.timer.data.chan ! BYTE 2 -- initialise timer2 (last byte)
    to.timer.control.chan ! BYTE 234
    to.timer.control.chan ! BYTE 255
    pwm.to.control ! acknowledge.tag
    fc:=500.0(REAL32)
    r:=fc/f
    Tc:=1.0E+6(REAL32)/(f*r)  --in microseconds
    Tc.2:= INT ROUND (Tc/(2.0(REAL32)))
    Tc.2:= Tc.2 - 2  -- allow for 2 us required by return of loop
    Tc.4:= (Tc/(4.0(REAL32)))
    R:=INT ROUND (r)
    f0:=1.0(REAL32)
    f.step:= (f-f0)/(REAL32 ROUND (N))
    m.step:= (m-m0)/(REAL32 ROUND (N))
    j:=0
    f.ramp:=f0
    m.ramp:=m0
    soft.start.time:= INT ROUND (1000000.0(REAL32)/Tc)--Tc is in usec(real)
    -- 0.5 sec per step
    control.to.pwm ? CASE forward.tag
    WHILE ( f.ramp < f)
      SEQ
        j:=0
        f.ramp:=f.ramp + f.step
        m.ramp:=m.ramp + m.step
        R:=INT ROUND (fc/f.ramp)
        R.2:=2 TIMES R
        WHILE ( j <= soft.start.time)  --start.soft.time
          SEQ
            PRI PAR
              SEQ
                clock ? time.now
                SEQ
                  IF
                    i> (R.2)
                      i:=0
                    TRUE
```

```
            SKIP
          IF
            (i REM 2) = 0
              SEQ
                tempA:= ((1000 TIMES i)/R)
                tempB:= tempA + 1333
                tempC:= tempA + 666
                sinA:= s[tempA+1]
                sinB:= s[tempB+1]
                sinC:= s[tempC+1]
                ta:=INT ROUND((Tc.4*((1.0(REAL32))-(m.ramp*sinA))))
                tb:=INT ROUND((Tc.4*((1.0(REAL32))-(m.ramp*sinB))))
                tc:=INT ROUND((Tc.4*((1.0(REAL32))-(m.ramp*sinC))))
                ta.lo:= (ta/\#00FF)
                ta.hi:= (ta>8)
                tb.lo:= (tb/\#00FF)
                tb.hi:= (tb>8)
                tc.lo:= (tc/\#00FF)
                tc.hi:= (tc>8)
                to.timer.data.chan ! BYTE ta.lo  -- load data
                to.timer.control.chan ! BYTE 0
                to.timer.control.chan ! BYTE 23
                to.timer.data.chan ! BYTE ta.hi  -- load data
                to.timer.control.chan ! BYTE 0
                to.timer.control.chan ! BYTE 23
                to.timer.data.chan ! BYTE tb.lo  -- load data
                to.timer.control.chan ! BYTE 1
                to.timer.control.chan ! BYTE 23
                to.timer.data.chan ! BYTE tb.hi  -- load data
                to.timer.control.chan ! BYTE 1
                to.timer.control.chan ! BYTE 23
                to.timer.data.chan ! BYTE tc.lo  -- load data
                to.timer.control.chan ! BYTE 2
                to.timer.control.chan ! BYTE 23
                to.timer.data.chan ! BYTE tc.hi  -- load data
                to.timer.control.chan ! BYTE 2
                --to.timer.control.chan ! BYTE 23
                to.timer.control.chan ! BYTE 255
                to.timer.control.chan ! BYTE 247
            TRUE
              SEQ
                tempA:= ((1000 TIMES i)/R)
                tempB:= tempA + 1333
                tempC:= tempA + 666
                sinA:=s[tempA+1]
                sinB:=s[tempB+1]
                sinC:=s[tempC+1]
                ta:=INT ROUND((Tc.4*((1.0(REAL32))+(m.ramp*sinA))))
                tb:=INT ROUND((Tc.4*((1.0(REAL32))+(m.ramp*sinB))))
                tc:=INT ROUND((Tc.4*((1.0(REAL32))+(m.ramp*sinC))))
                ta.lo:= (ta/\#00FF)
                ta.hi:= (ta>8)
                tb.lo:= (tb/\#00FF)
                tb.hi:= (tb>8)
                tc.lo:= (tc/\#00FF)
                tc.hi:= (tc>8)
                to.timer.data.chan ! BYTE ta.lo  -- load data
                to.timer.control.chan ! BYTE 224
                to.timer.control.chan ! BYTE 247
                to.timer.data.chan ! BYTE ta.hi  -- load data
                to.timer.control.chan ! BYTE 224
                to.timer.control.chan ! BYTE 247
                to.timer.data.chan ! BYTE tb.lo  -- load data
                to.timer.control.chan ! BYTE 225
                to.timer.control.chan ! BYTE 247
                to.timer.data.chan ! BYTE tb.hi  -- load data
                to.timer.control.chan ! BYTE 225
                to.timer.control.chan ! BYTE 247
                to.timer.data.chan ! BYTE tc.lo  -- load data
                to.timer.control.chan ! BYTE 226
                to.timer.control.chan ! BYTE 247
                to.timer.data.chan ! BYTE tc.hi  -- load data
                to.timer.control.chan ! BYTE 226
                --to.timer.control.chan ! BYTE 247
                to.timer.control.chan ! BYTE 31
                to.timer.control.chan ! BYTE 23
            i:=i+1
          clock ? AFTER  time.now PLUS Tc.2
        SEQ
          SKIP
      j:=j+1
m:=m.ramp
pwm.to.control ! full.speed.tag
i:=0
cycling:=TRUE
R.2:=2 TIMES R
phi:=0
WHILE cycling
  PRI ALT
    control.to.pwm ? CASE
      vector.control.tag
        SEQ
          control.to.pwm ? CASE vector.control.parameter;phi;R;m
          R.2:=2 TIMES R
      change.pwm.parameter.tag
        SEQ
          control.to.pwm ? CASE pwm.parameter;R;m
          R.2:=2 TIMES R
      finish.tag
        SEQ
          pwm.to.control ! acknowledge.tag
          cycling:=FALSE
    TRUE & SKIP
      PRI PAR
        SEQ
          clock ? time.now
          SEQ
            IF
              i>(R.2)
                i:=0
              TRUE
                SKIP
            IF
              (i REM 2) =0
                SEQ
                  tempA:= ((1000 TIMES i)/R) + phi
                  tempB:= tempA + 1333
                  tempC:= tempA + 666
                  sinA:=s[tempA+1]
                  sinB:=s[tempB+1]
                  sinC:=s[tempC+1]
                  ta:=INT ROUND((Tc.4*((1.0(REAL32))-(m*sinA))))
                  tb:=INT ROUND((Tc.4*((1.0(REAL32))-(m*sinB))))
                  tc:=INT ROUND((Tc.4*((1.0(REAL32))-(m*sinC))))
                  ta.lo:= (ta/\#00FF)
                  ta.hi:= (ta>8)
                  tb.lo:= (tb/\#00FF)
                  tb.hi:= (tb>8)
                  tc.lo:= (tc/\#00FF)
                  tc.hi:= (tc>8)
                  to.timer.data.chan ! BYTE ta.lo  -- load data
                  to.timer.control.chan ! BYTE 0
                  to.timer.control.chan ! BYTE 23
                  to.timer.data.chan ! BYTE ta.hi  -- load data
                  to.timer.control.chan ! BYTE 0
                  to.timer.control.chan ! BYTE 23
                  to.timer.data.chan ! BYTE tb.lo  -- load data
                  to.timer.control.chan ! BYTE 1
                  to.timer.control.chan ! BYTE 23
                  to.timer.data.chan ! BYTE tb.hi  -- load data
                  to.timer.control.chan ! BYTE 1
                  to.timer.control.chan ! BYTE 23
                  to.timer.data.chan ! BYTE tc.lo  -- load data
                  to.timer.control.chan ! BYTE 2
                  to.timer.control.chan ! BYTE 23
                  to.timer.data.chan ! BYTE tc.hi  -- load data
                  to.timer.control.chan ! BYTE 2
                  --to.timer.control.chan ! BYTE 23
                  to.timer.control.chan ! BYTE 255
```

```
      to.timer.control.chan ! BYTE 247
    TRUE
      SEQ
        tempA:= ((1000 TIMES i)/R) + phi
        tempB:= tempA + 1333
        tempC:= tempA + 666
        sinA:=s[tempA+1]
        sinB:=s[tempB+1]
        sinC:=s[tempC+1]
        ta:=INT ROUND((Tc.4*((1.0(REAL32))+(m*sinA))))
        tb:=INT ROUND((Tc.4*((1.0(REAL32))+(m*sinB))))
        tc:=INT ROUND((Tc.4*((1.0(REAL32))+(m*sinC))))
        ta.lo:= (ta/\#00FF)
        ta.hi:= (ta>8)
        tb.lo:= (tb/\#00FF)
        tb.hi:= (tb>8)
        tc.lo:= (tc/\#00FF)
        tc.hi:= (tc>8)
        to.timer.data.chan ! BYTE ta.lo  -- load data
        to.timer.control.chan ! BYTE 224
        to.timer.control.chan ! BYTE 247
        to.timer.data.chan ! BYTE ta.hi  -- load data
        to.timer.control.chan ! BYTE 224
        to.timer.control.chan ! BYTE 247
        to.timer.data.chan ! BYTE tb.lo  -- load data
        to.timer.control.chan ! BYTE 225
        to.timer.control.chan ! BYTE 247
        to.timer.data.chan ! BYTE tb.hi  -- load data
        to.timer.control.chan ! BYTE 225
        to.timer.control.chan ! BYTE 247
        to.timer.data.chan ! BYTE tc.lo  -- load data
        to.timer.control.chan ! BYTE 226
        to.timer.control.chan ! BYTE 247
        to.timer.data.chan ! BYTE tc.hi  -- load data
        to.timer.control.chan ! BYTE 226
        --to.timer.control.chan ! BYTE 247
        to.timer.control.chan ! BYTE 31
        to.timer.control.chan ! BYTE 23
    i:=i+1
  clock ? AFTER time.now PLUS Tc.2
SEQ
  SKIP
```

```
PROTOCOL MOTOR
  CASE
    -- On-line command tags
    vector.control.tag
    change.reference.flux.tag
    change.reference.speed.tag
    change.carrier.freq.tag
    change.pwm.parameter.tag
    forward.tag
    reverse.tag
    dynamic.brake.tag
    -- operational tags
    full.speed.tag
    acknowledge.tag
    finish.tag
    -- machine.data ; data format : r1,r2,L1,L2,Lo,M
    machine.data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32
    -- startup data ; data format : f,m0,m,N
    start.up.data;REAL32;REAL32;REAL32;INT
    -- control data ; data format : Vdc,Fdr,controller.gain
    control.data;REAL32;REAL32;REAL32
    -- On-line command channels
    speed.ref;REAL32
    vector.control.parameter;INT;INT;REAL32 -- data format: phi,R,m
    pwm.parameter;INT;REAL32                -- data format: R,m
    flux.ref;REAL32
    carrier.freq;REAL32                     -- (not used yet)
    -- feedback channels
    speed.measure;REAL32
    control.parameter;INT;REAL32;REAL32     -- phi,m,Wsl
    position;REAL32                         -- (not used yet)
    link.current;REAL32                     -- DC link current is read
:
PROC speed(CHAN OF MOTOR speed.to.current,current.to.speed)
  #USE userio
  CHAN OF BYTE to.timer.data.chan:
  CHAN OF BYTE from.timer.data.chan:
  CHAN OF BYTE to.timer.control.chan:
  CHAN OF BYTE from.timer.control.chan:
  BOOL cycling:
  VAL link0in IS 4 :
  VAL link0out IS 0 :
  VAL link1in IS 5 :
  VAL link1out IS 1 :
  VAL link2in IS 6 :
  VAL link2out IS 2 :
  VAL link3in IS 7 :
  VAL link3out IS 3 :
  PLACE to.timer.data.chan AT 0:
  PLACE from.timer.data.chan AT 4:
  PLACE to.timer.control.chan AT 1:
  PLACE from.timer.control.chan AT 5:
  SEQ
    SEQ  -- programming counter 0
      to.timer.data.chan ! BYTE #10
      to.timer.control.chan ! BYTE #FF
      to.timer.control.chan ! BYTE #0B
      to.timer.control.chan ! BYTE #FF
      to.timer.data.chan ! BYTE #0
      to.timer.control.chan ! BYTE #08
      to.timer.control.chan ! BYTE #FF
    cycling:=TRUE
    WHILE cycling
      PRI ALT
        current.to.speed ? CASE finish.tag --finish signal from host via T3
          SEQ
            speed.to.current ! acknowledge.tag
            cycling:=FALSE
        TRUE & SKIP
          SEQ
            PAR   --provoke 64 microsecond timer
              REAL32 Wr.elec:
              TIMER clock:
              BYTE speed.pulse:
              INT time.now,speed.unscaled:
              SEQ
                clock ? time.now
                to.timer.control.chan ! BYTE #04
                from.timer.data.chan ? speed.pulse
                to.timer.control.chan ! BYTE #FF
                speed.unscaled:= 255 - (INT (speed.pulse))
                --Wr(elec):= (4pi/2048)*(n/p)  n=no.of pulses,p=sampling period
                Wr.elec:=1.19842(REAL32)*(REAL32 ROUND (speed.unscaled))
                speed.to.current ! speed.measure;Wr.elec -- send Wr.elec
                --clear counter for next sampling
                to.timer.data.chan ! BYTE #FF
                to.timer.control.chan ! BYTE #08
                to.timer.control.chan ! BYTE #FF
                clock ? AFTER time.now PLUS 80 -- T(sampling)=5.12ms
              SEQ
                SKIP
```

```
PROTOCOL MOTOR
  CASE
   -- On-line command tags
   vector.control.tag
   change.reference.flux.tag
   change.reference.speed.tag
   change.carrier.freq.tag
   change.pwm.parameter.tag
   forward.tag
   reverse.tag
   dynamic.brake.tag
   -- operational tags
   full.speed.tag
   acknowledge.tag
   finish.tag
   -- machine.data ; data format : r1,r2,L1,L2,Lo,M
   machine.data;REAL32;REAL32;REAL32;REAL32;REAL32;REAL32
   -- startup data ; data format : f,m0,m,N
   start.up.data;REAL32;REAL32;REAL32;INT
   -- control data ; data format : Vdc,Fdr,controller.gain
   control.data;REAL32;REAL32;REAL32
   -- On-line command channels
   speed.ref;REAL32
   vector.control.parameter;INT;INT;REAL32 -- data format: phi,R,m
   pwm.parameter;INT;REAL32                -- data format: R,m
   flux.ref;REAL32
   carrier.freq;REAL32                     -- (not used yet)
   -- feedback channels
   speed.measure;REAL32
   control.parameter;INT;REAL32;REAL32     -- phi,m,Wsl
   position;REAL32                         -- (not used yet)
   link.current;REAL32                     -- DC link current is read
:
PROC current(CHAN OF MOTOR current.to.control,control.to.current,
             current.to.speed,speed.to.current)
  CHAN OF BYTE from.adc,to.adc:
  REAL32 Wr.elec,idc:
  BYTE idc.lo.byte,idc.hi.byte:
  BOOL cycling:
  VAL link0in IS 4 :
  VAL link0out IS 0 :
  PLACE from.adc AT 4:
  PLACE to.adc AT 0:
  SEQ
    cycling:=TRUE
    WHILE cycling
      PRI ALT
        control.to.current ? CASE finish.tag    -------- check here
          SEQ
            current.to.speed ! finish.tag   -- pass to T2
            speed.to.current ? CASE acknowledge.tag -- wait T2 to reply
            current.to.control ! acknowledge.tag -- send ack. back to host
            cycling:=FALSE
        speed.to.current ? CASE speed.measure;Wr.elec
          SEQ
            current.to.control ! speed.measure;Wr.elec
            -- current will also be sampled at the same rate (5.12ms) as speed
            SKIP
            -- Channel 1 is used for sampling for idc.
            SEQ
              to.adc ! BYTE 1
              from.adc ? idc.lo.byte
              from.adc ? idc.hi.byte
              idc:=REAL32 ROUND (((INT idc.lo.byte)\/((INT idc.hi.byte)<<8))-2048)
              idc:=(idc*10.0(REAL32))/2048.0(REAL32)
              -- Only the dc link current is measured. Since the current transformer (CT)
              -- has a conversion factor of 0.1 V/A, the actual current is 10 times the
              -- value
              idc:=idc*10.0(REAL32)
              current.to.control ! link.current;idc
```

The size of the matrix is limited by 8088, and the limited for display is 4044. This imposes a severe constraint on the number of samples over a simulation run may well extend this number. Since the minimum frequency is application-dependent at least 10 times the maximum frequency in the model is used. The total simulation time is divided into internvals a batch file is used. The simulation programme is run in these intervals. The matrix is then sampled a regualr interval by a created function called **redm(x,p)**, which reduces the size of matrix **x** by factor of **p**.

The program listing of **redm(x,y)** is as follows:

```
function X=redm(Y,p)
% reduce matrix y by p times
n=fix(length(Y)/p);
for i=1:n;
  X(i)=Y(i*p);
end
```

All the circuits, except for the inverter, were designed with a Eurocard format and were mounted inside a cased Eurocard rack system. The case of the rack was earthed to protect the circuits inside from radio frequency interference (RFI) which is commonly found in motor drive applications. A backplane printed circuit board (PCB) that can hold up to 16 Eurocards was constructed. This back board provides the interconnections between the cards and connection points for the power supplies (+Vcc, -Vcc) and the system ground. The Eurocard rack system provides a very convenient and industrially standard way for further hardware development. Only six out of the sixteen Eurocard spaces have been used in this project. The point-to-point connection of the transputer with other circuits simplifed interfacing and the development of the prototype system.

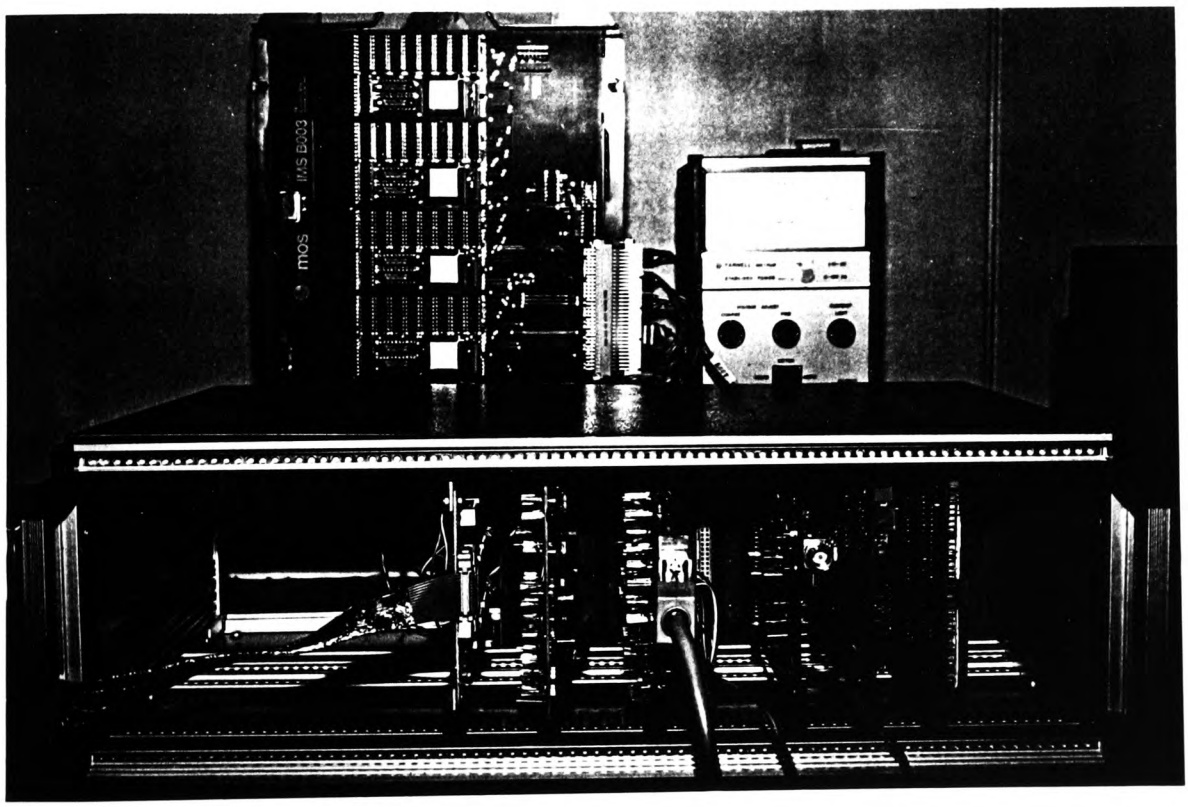


Fig.E Eurocard rack system hosting the interfacing circuits

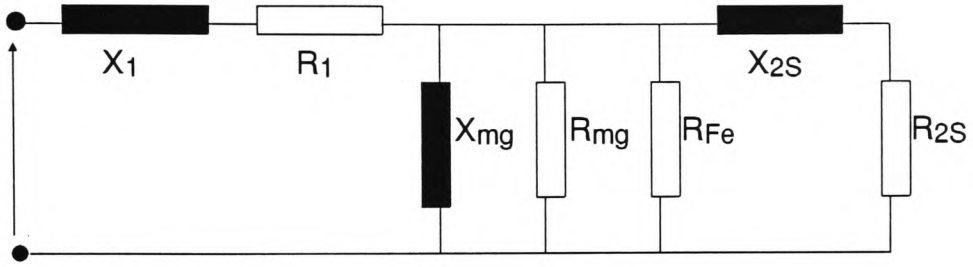
(a) Specification of the induction motor under investigation at rated operation :

— 2.2kW, 3-Phase, 4 Pole, 240V (Δ-connected), slip 4.91% (50Hz)

— Encoder (1024 pulse/rev) fitted at non-drive end for vector control

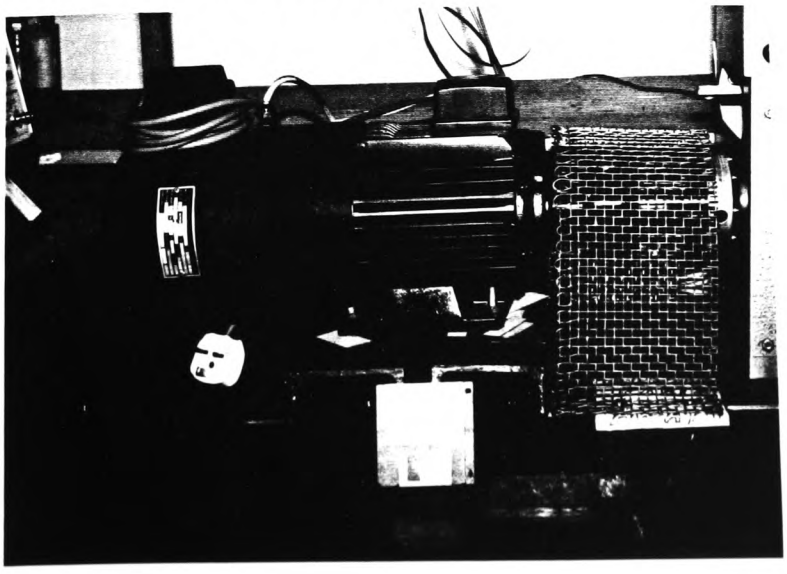
— Forced ventilation

(b) Machine equivalent circuit :

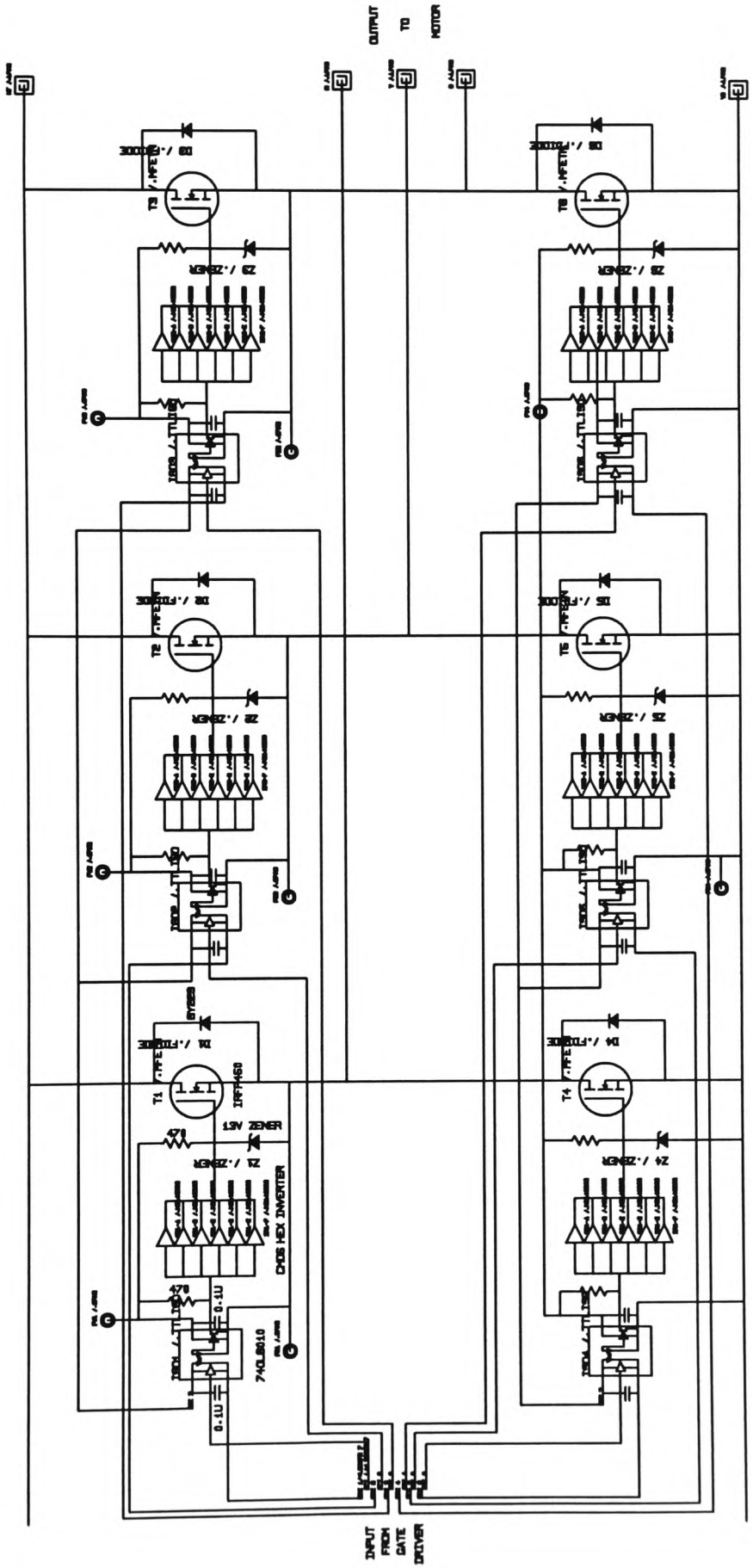


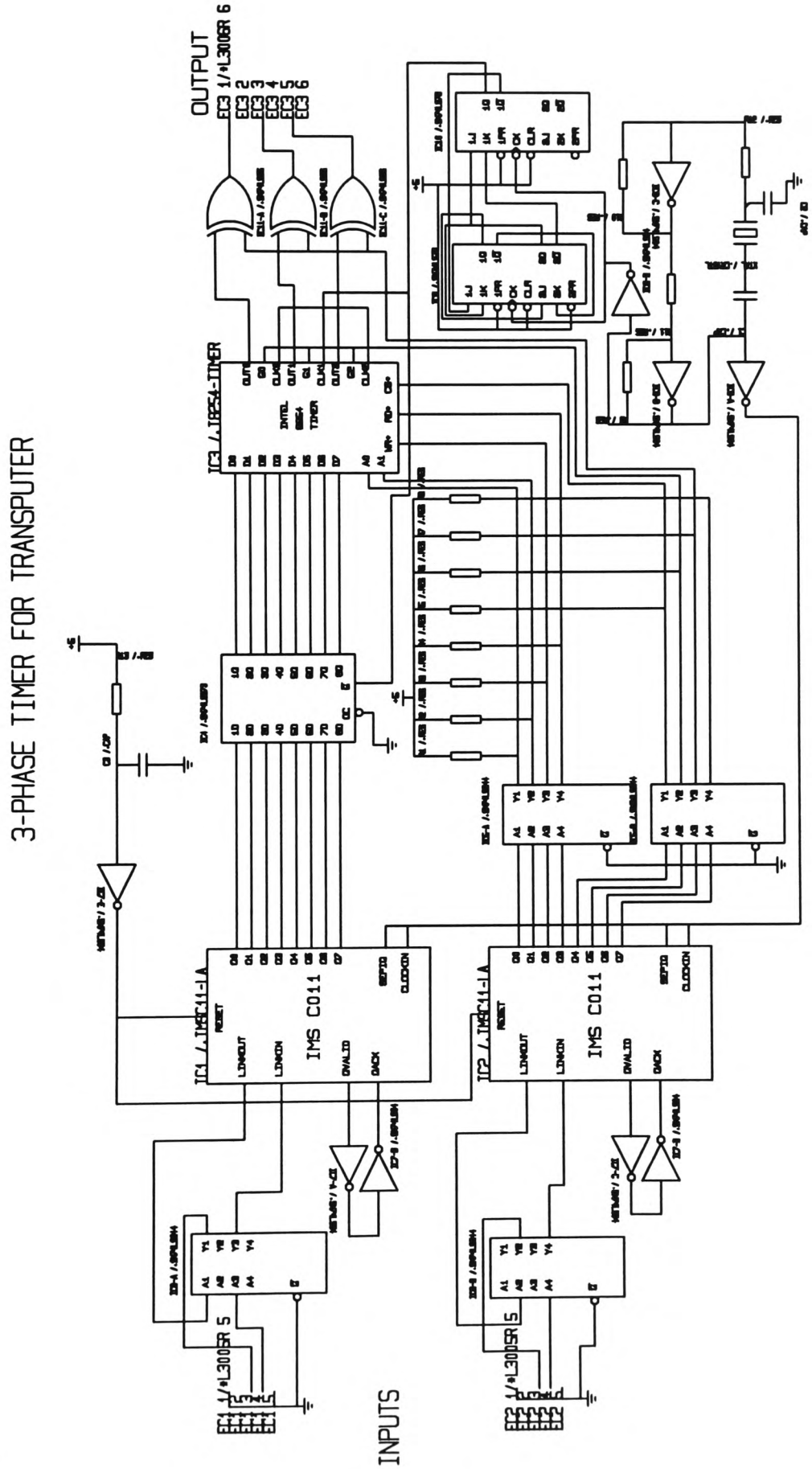
Stator leakage inductance, X_1	: 3.661 Ω
Stator resistance, R_1	: 3.76 Ω
Magnetising reactance, X_{mg}	: 84.2 Ω
Equivalent resistance for mechanical losses,R_{mg}	: 1387 Ω
Equivalent resistance for for iron losses, R_{Fe}	: 1153 Ω
Leakage inductance (referred to stator), X_{2S}	: 8.765 Ω
Rotor resistance, R_{2S}	: 2.571 Ω

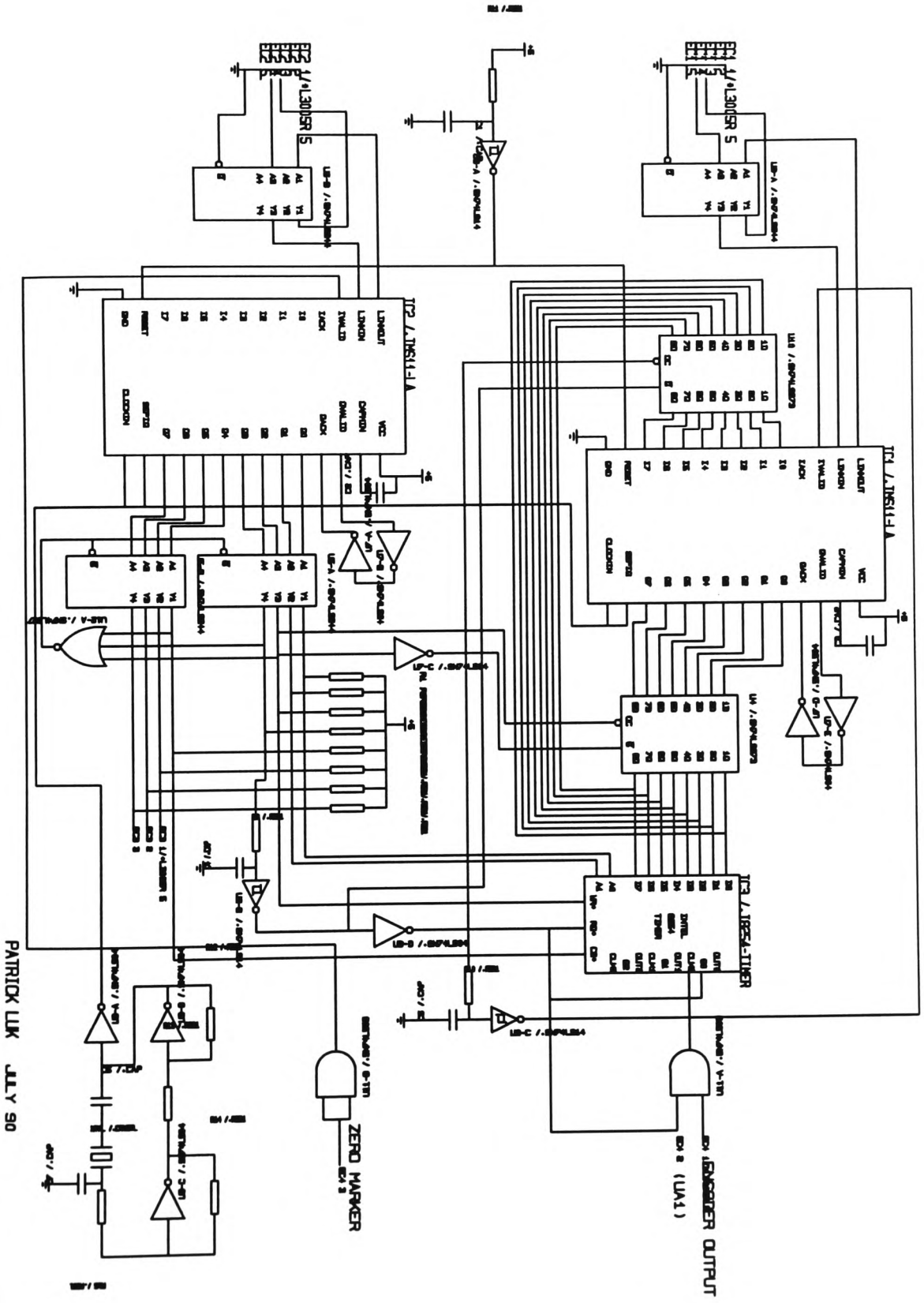
(c) Slide of the induction motor under investigation



CIRCUIT DIAGRAM OF 3-PHASE, FULL BRIDGE HEXFET INVERTER




3-PHASE TIMER FOR TRANSPUTER
OUTPUT
INPUTS
IMS C011
IMS C011


INTERFACING CIRCUIT FOR SPEED FEEDBACK
PATRICK LUK JULY 90
ZERO MARKER
ENCODER OUTPUT

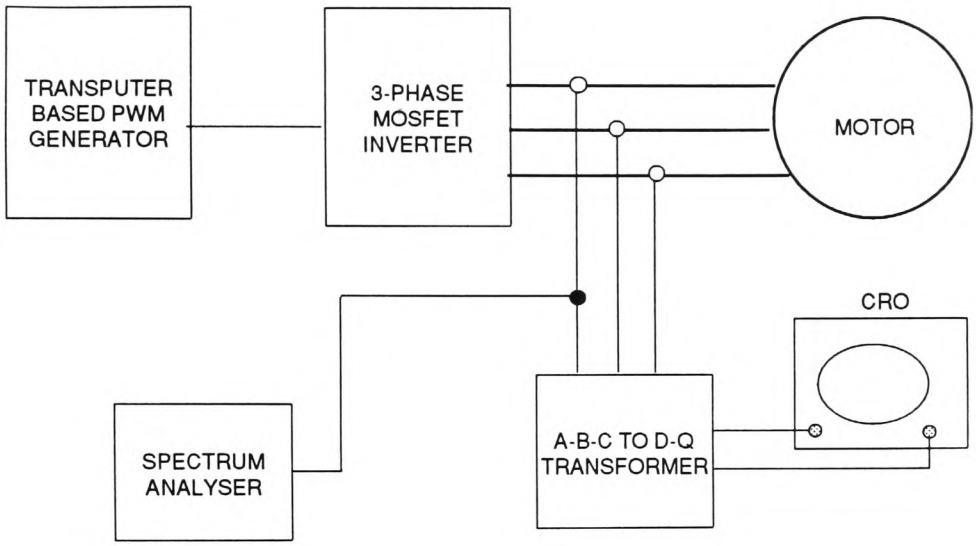


Fig.H1 Block diagram of experimental setup for results of chapter 4

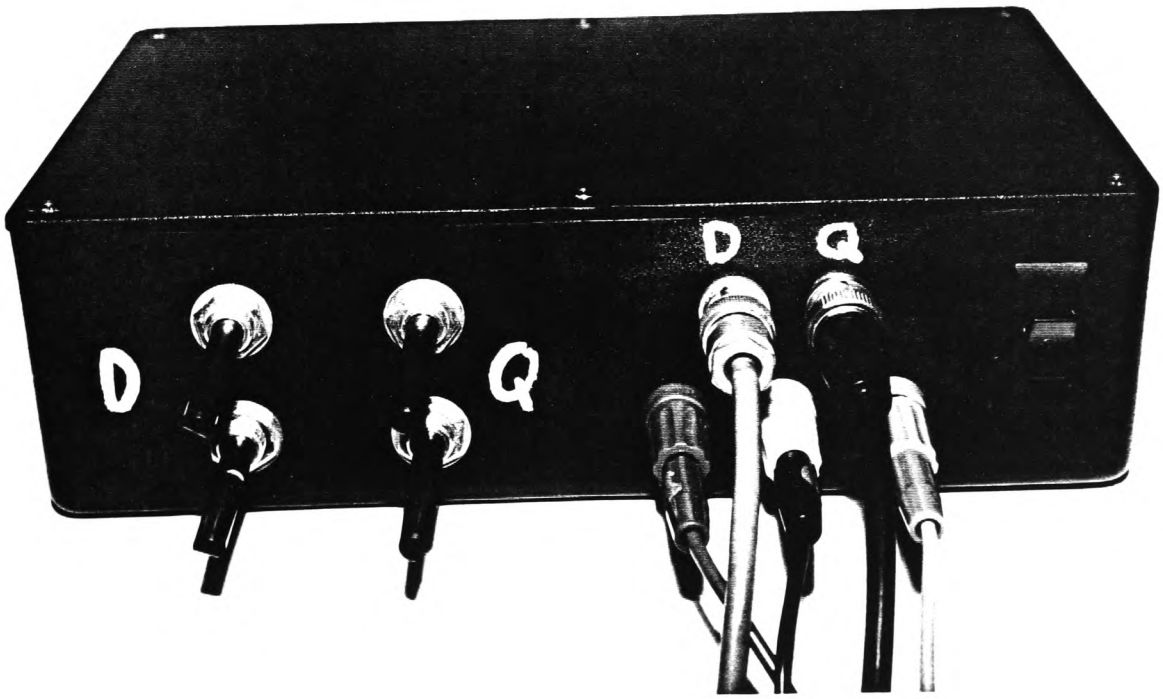


Fig.H2 The *A-B-C* to *D-Q* transformer employing operational amplifiers and *RC* circuit

[Acarnley,1987] Acarnley, P.P.; Finch, J.W. and Atkinson, D.J., 1987. "Field-orientation in AC Drives: prospects," *IEE EMDA'87*, London.

[Acarnley,1991] Acarnley, P.P.; Chai, H. and Atkinson, D.J., "Induction Motor Parameter Estimation Using On-Line Spectral Analysis," *Proceedings EPE'91, 4th European Conference on Power Electronics*, Vol.3 pp.326-331, Firenze, 3-6 Sept. 1991.

[Acharya,1986] Acharya, S.; Shakhawat, S.; Shepherd, W.; Rao, U.M. and Ng, Y.M., "Microprocessor-Based PWM Inverter Using Modified Regular Sampling Techniques," *IEEE Transactions on Industry Applications*, Vol.IA-22, No.2, March/April 1986.

[Adkins,1975] Adkins, B. and Harley, R.G., "The General Theory of Alternating Current Machines: Appliction to Practical Problems," *Chapman and Hall Ltd.*, 1975.

[Akamatsu,1982] Akamatsu, M.; Ikeda, K.; Tomei, H. and Yano S., "High Performance IM Drive by Coordinate Control Using a Controlled Current Inverter," *IEEE Trans. Ind. App.* Vol.IA-18, No.4, July/August 1982, pp.382-392.

[Al-hosini,1985] Al-hosini, F., "A noval drive circuit for power MOSFETs" *PCI. October 1985 Proceedings*, pp.33-44.

[Asher,1990] Asher, G.M. and Sumner, M., "Parallelism and transputer for real-time high-performance control of AC induction motors," *IEE Proceeding*, Vol.137, Pt.D, No.4, July 1990.

[Bedford,1972] Bedford,R.E.;Nene,V.D.,"Complex Frequency-Domain Analysis of Inverter-Fed Induction Machines," *IEEE Trans. on Ind. App.*, IA-8, No.3, 269-277, 1972.

[Blaschke,1972] Blaschke, F., "The Principle of Field Orientation as Applied to the New TRANSVECTOR Closed Loop Control System for Rotating Field Machines," *Siemens Rev.*, 1972, p.217.

[Bose,1981] Bose, B.K., "Adjustable Speed AC Drive Systems," *IEEE Press Selected Reprint Series, IEEE Industry Applications Society*, 1981.

[Bose,1984] Bose, B.K., "Scalar Decoupled Control of Induction Motor," *IEEE Transactions on Industry Applications*, Vol.IA-20, No.1, January/February 1984.

[Bose,1986] Bose, B.K., "Power Electronics and AC Drives," *Prentice-Hall*, 1986.

[Bose,1989] Bose, B.K., "Power Electronics - An Emerging Technology," *IEEE Transactions on Industrial Electronics*, Vol.36, No.3, August 1989.

[Bowes,1981] Bowes, S.R. and Mount, M.J., "Microprocessor control of PWM inverters," *IEE Proc.*, Vol.128, Pt.B, No.6, November 1981.

[Bowes,1983] Bowes, S.R.; Mech, M.I. and Clare, J., "Steady- state Performance of PWM Inverter Drives," *IEE Proceedings*, Vol.130, Pt.B, No.4, July 1983.

[Bowes,1985] Bowes, S.R. and Midoun, A., "Suboptimal switching strategies for microprocessor-controlled PWM inverter drives," *IEE Proceedings*, Vol.132, Pt.B, No.3, May 1985.

[Bowes,1988] Bowes, S.R. and Clare, J.C., "Computer-aided design of PWM power electronic variable-speed drives," *IEE Proc.*, Vol.135, Pt.B, No.5, September 1988.

[Boys,1985] Boys, J.T. and Walton, S.J.,. "A loss minimised sinusoidal PWM inverter," *IEE Proceedings*, Vol.132, Pt.B, No. 5, September 1985.

[Boys,1990a] Boys, J.T. and Walton, S.J., "Dynamic flux controlled AC drive," *IEE Proceedings*, Vol.137, Pt.B, No.4, July 1990, pp.259-263.

[Boys,1990b] Boys, J.T. and Handley, P.G., "Harmonic analysis of space vector modulated PWM waveforms," *IEE Proceedings*, Vol.137, Pt.B, No.4, July 1990, pp.197-204.

[Buja,1980] Buja, G.S., "Optimum Output Waveforms in PWM Inverters," *IEEE Transactions on Industry Applications*, Vol.IA-16, No.6, November/December 1980.

[Buja,1985] Buja, G.S. and De Nardi, P., "Application of a Signal Processor in PWM Inverter Control," *IEEE Transactions on Industrial Electronics*, Vol.IE-32, No. 1, February 1985.

[Chan,1987] Chan, C.C. and Lo, W., "Control Strategy of PWM Inverter Drive System for Electric Vehicles," *IEEE Transactions on Industrial Electronics*, Vol.IE-34, No.4, Nov.1987.

[Chan,1988] Chan, C.C. and Ng C.W., "Adaptive AC Propulsion System for Electric Vehicles," *Electric Vehicle Association of Canada*, Harbour Castle Westin Toronto, Ontario, Canada, November 1988.

[Chan,1990] Chan, C.C. and Wang H., "An Effective Method for Rotor Resistance Identification for High-Performance Induction Motor Vector Control," *IEEE Trans. Ind. Electronics*, Vol.37, No.6, December 1990, pp.477-482.

[Cogan,1985] Cogan, A. and Blanchard, R.A., "Future Trends in Semiconductor Switching," *PCI*, October 1985 Proceedings, pp.194.

[Davies,1991] Davies, T., "Industrial high fidelity - the practical flux vector drive," *Design Products & Applications*, January 1991, pp.33-37.

[Dewan,1970] Dewan, S.B. and Kankam, M.D.,"A Method for Harmonic Analysis of Cycloconverters," *IEEE Trans. on Ind. and Gen. App.*, IGA- 6, No.5, 455-62, 1970.

[Dowsing,1988] Dowsing, R.D., " Introduction to Concurrency Using OCCAM," *Van Nostrand Reinhold (International)*, 1988.

[Du,1991] Du, T.; Brdys, M.A. and Taufig, J.A., "Analytical Parameter Sensitivity Expressions for a Field Oriented Induction Motor Drive," Conf. proceedings. *EPE'91, 4th Conference on Power Electronics and applications*, Vol.3, pp.624-628, Florence, Italy, July, 1991.

[Dwyer,1979] Dwyer, E., "A Lookup Table Based Microprocessor Controller for a Three Phrase PWM Inverter," *IEEE / IECI Proc. 5th Ann. Conference Industrial and Control Applications of Microprocessors*, March 19-21, 1979.

[Econews,1990] *ECONEWS No.53*, Newspaper of the Green Party, Oct/Nov. 1990.

[Emerald,1990] Emerald, P.R., ""Merged" Technologies: Present Capabilities and Value for Applicaitons in High-Voltage and "Smart Power" Circuitry," *IEEE Transactions on Industry Applications*, Vol.26, No.6, November/December 1990.

[Enjeti,1990] Enjeti, P.N.; Ziogas, P.D. and Lindsay, J.F., "Programmed PWM Techniques to Eliminate Harmonics: A Critical Evaluation," *IEEE Transactions on Industry Applications*, Vol.26, No.2, March/April 1990.

[Gabriel,1980] Gabriel, R.; Leonhard, W. and Nordby, C.J., "Field-Oriented Control of a Standard AC Motor Using Microprocessors," *IEEE Transactions on Industry Applications*, Vol.IA-16, No.2, March/April 1980.

[Graces,1980] Graces, L., "Parmeter adaptation for the speed controlled static ac drives with squirrel cage induction motor," *IEEE Trans. Industry Applications*, vol. IA-16, no. 2 Mar./Apr. 1980.

[Grant,1983] Grant, D.A.; Houldsworth, J.A. and Lower, K.N., "A New High-Quality PWM AC Drive," *IEEE Transactions on Industry Applications*, Vol.IA-19, No.2, March/April 1983.

[Grant,1987] Grant, D.A. , Application note 967, International Recifier HEXFET Designer's Manual, 1987.

[Haneda,1989] Haneda, H. and Nagao, A., "Digitally Controlled Optimal Position Servo of Induction Motors," *IEEE Transactions on Industrial Electronics*, Vol.36, No.3, August 1989.

[Harashima,1985] Harashima, F.; Kondo, S.; Ohnishi, K.; Kajita, M. and Susono, M., "Multimicroprocessor-Based Control System for Quick Response Induction Motor Drive," *IEEE Trans. Ind. App.*, Vol.IA-21, No.4, May/June 1985, pp.602-609.

[Hasse,1968] Hasse, K., "Zum Dynamischen Vehalten der Asynchronmachine bei Betrieb Mit Variabler Standerfrequenz and Standerspannung," *ETZ-A, Bd. 89, H4*, pp.77, 1968.

[Hasse,1969] Hasse, K., "Zur Dynamik drehzahlgeregelter Antriebe mit stromrichtergespeisten Asynchron-Kurzschlusslaufermaschinen," Diss. TH Darmstadt, 1969.

[Hindmarsh,1982] Hindmarsh, J., "Work Examples in Electric Machines and Drives," Pergamon Press, 1982.

[Ho,1986] Ho, E.Y.Y. and Sen, P.C., "Digital Simulation of PWM Induction Motor Drives for Transient and Steady-State Performance," *IEEE Transactions on Industrial Electronics*, Vol.IE-33, No.1, February 1986.

[Hotlz,1983a] Holtz, J, and Stadtfeld, S., "A Predictive Controller for the stator Current Vector of AC Machines Fed from a Switched Voltage Source," *International Power Electronics Conference IPEC, Tokyo*, 1983, pp.1665-1675.

[Hotlz,1983b] Holtz, J. and Stadtfeld, S., "Field-Oriented Control by Forced Motor Currents in a Voltage Fed Inverter Drive," *3. IFAC Symposium, Control in Power Electronics and Electrical Drives, Lausanne*, 1983, pp.103-110.

[Holtz,1985] Holtz, J. and Stadtfeld, S., "A PWM Inverter Drive System with On-Line Optimized Pulse Patterns," *First European Conference on Power Electronics and Applications, EPE, Brussels*, 1985, pp.3.21-3.25.

[Holtz,1988] Holtz, J. and Bube, E., "Field Oriented Asynchronous Pulse-Width Modulation for High Performance AC Machine Drives Operating at Low Switching Frequency," *IEEE Industry Applications Society Annual Meeting, Pittsburgh*, 1988, pp.412-417.

[Houldsworth,1984] Houldsworth,J.A. and Grant,D., "The use of harmonic distortion to increase the output of a three-phase PWM inverter," *IEEE Trans. Ind. Appl. Soc., vol. IA-20, pp.1224-1228, Sept./Oct. 1984.*

[Howe,1987] Howe, C.D. and Moxon, B., " How to program parallel processors," *IEEE Spectrum*, September, 1987.

[Inmos,1988] Transputer Development System, inmos, *Prentice Hall*, 988

[Intel,1987] Intel's Microsystem Components Handbook, Peripherals Vol.II, 1985.

[IR,1987] International Recifier HEXFET Designer's Manual, 1987.

[Ito,1983] Ito, T.; Yamaguchi, T.; Ueda, R.; Mochizuki, T. and Takata, S., "Analysis of Field Orientation Control of Current Source Inverter Drive Induction Motor Systems," *IEEE Trans. Ind. App.*, Vol.IA-19, No.2, March/April 1983, pp.206-209.

[Jacovides,1973a] Jacovides, L; "Analysis of Induction Motor Drives with a Nonsinusoidal Supply Voltage Using Fourier Analysis," *IEEE Trans. on Ind. App., IA-9, No.6, 741-7*, 1973a.

[Jacovides,1973b] Jacovides, L.J., "Analysis of a Cycloconverter-Induction Motor Drive System Allowing for Stator Current Discontinuities," *IEEE Trans. on Ind. App.*, IA-9, No. 2, 206-215, 1973b.

[Jayne,1976] Jayne, M.G., "The Application of Communication Principles to Pulse-Width Modulated Inverters," *PhD thesis, CNAA* , 1976.

[Jayne,1977], Jayne, M.G.; Bowes, S.R. and Bird, B.M., "Developments in PWM Inverters", *Proceedings of 2nd IFAC Symposium on Control in Power Electronics, Dusseldorf*, 1977.

[Jardan,1969] Jardan, K.R.; Devan, S.B. and Slemon, G.R., "General Analysis of Three-Phase Inverters," *IEEE Trans. on Ind. and Gen. App.*, IGA-5, No. 6, 672-9, 1969.

[Jones,1984] Jones, B.L. and Brown, J.E., "Electric variable speed drives", IEE Proc ., vol.131, pt.A, no.7, September 1984, pp.516-558.

[Kaimoto,1982] Kaimoto, M.; Hashi, M.; Yanase, T. and Nakano, T., "Performance Improvement of Current Source Inverter-Fed Induction Motor Drives," *IEEE Transactions on Industry Applications*, Vol.IA-18, No.6, November/December 1982.

[Kassakian,1979] Kassakian, J.G., "Simulating Power Electronic Systems - A New Approach," *Proc. IEEE, 67*, October, 1979, pp.1428-1439.

[Kazmierkowski,1985] Kazmierkowski, M.P. and Kopcke, H., "A Simple Control System for Current Source Inverter-Fed Induction Motor Drives," *IEEE Transactions on Industry Applications*, Vol.IA-21, No.4 May/June 1985.

[Kazmierkowski,1991] Kazmierkowski, M.P. and Sulkowski, W., "A Novel Vector Control Scheme for Transistor PWM Inverter-Fed Induction Motor Drive," *IEEE Tran. Ind. Electronics*, Vol.38, No.1, February 1991, pp.41-47.

[Khanniche,1988] Khanniche, M.S.; Belaroussi, M. and Sethuraman, S.K., "An Alogrithm for Generating Optimised PWM for Real Time Micro Control Applictions," *Third Internation Conference on Power Electronics and Variable-Speed Drives*, Conference Publication, No.291, July 1988, pp.386-389.

[Kirschen,1985] Kirschen, D.S.; Novotny, D.W. and Lipo, T.A. "On-Line Efficiency Optimization of a Variable Frequency Induction Motor Drive," *IEEE Trans. on Ind. App.*, Vol.IA-21, No.4, May/June 1985, pp.610-616.

[Koyama,1986] 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 Transactions on Industry Applications*, Vol.IA-22, No.3 May/June 1986.

[Kubo,1985] Kubo, K.; Watanabe, M.; Ohmae, T. and Kamiyama, K., "A Fully Digitalized Speed Regulator using Multimicroprocessor System for Induction Motor Drives," *IEEE Trans. Ind. App.*, Vol.IA-21, No.4, July/August 1985, pp.1001-1008.

[Kusko,1987] Kusko,A, "Brushless DC Motors Using Unsymmetrical Field Magnetisation," *IEEE Transactions on Industry Applications*, Vol.IA-23, No.2, March/April 1987, pp.319-326.

[Leonard,1896] Leonard,H.W.,"Volts versus ohms," *AIEE Trans.,Vol.13,1886*,pp.337-399.

[Leonhard,1975] Leonhard, W., 1975, "Introduction to AC-Motor Control Using Field Coordinates," *Simposio Sulla Evoluzione Nella Dinamica Delle Macchine Elettriche Rotanti, Tirrenia*, 370.

[Leonhard,1985] Leonhard, W., "Control of Electrical Drives," *Spronger-Verlag*, 1985.

[Lipo,1975] Lipo, T.A. and Turnbull, F.G., "Analysis and Comparison of Two Types of Square-Wave Inverter Drives," *IEEE Transactions on Industry Applications*, Vol.IA-11, No.2, March/April 1975.

[Lipo,1984] Lipo, T.A. and Consoli, A., "Modeling and Simulation of Induction Motors with Saturable Leakage Reacances," *IEEE Transactions on Ind. Applications*, Vol.IA-20, No.1, January/February 1984, pp.180-189.

[Liu,1989] Liu, C.H.; Hwu, C.C. and Feng, Y.F., "Modelling and Implementation of a Microprocessor-Based CSI-Fed Induction Motor Drive Using Field-Oriented Control," *IEEE Transactions on Industry Applications*, Vol.25, No.4, July/August 1989.

[Lorenz,1986] Lorenz, R.D., "Tuning of Field-Oriented Induction Motor Controllers for High-Performance Applications," *IEEE Transactions on Industry Applications*, Vol.IA-22, No.2, March/April 1986.

[Lorenz,1990a] Lorenz, R.D. and Divan, D.M., "Dynamic Analysis and Experimental Evaluation of Delta Modulators for Field-Oriented AC Machine Current Regulators," *IEEE Transactions on Industry Applications*, Vol.26, No.2, March/April 1990.

[Lorenz,1990b] Lorenz, R.D. and Lawson, D.B., "Flux and Torque Decoupling Control for Field-Weakened Operation of Field-Oriented Induction Machines," *IEEE Transactions on Industry Applications*, Vol.26, No.2, March/April 1990.

[Lorenz,1990c] Lorenz, R.D. and Lawson, D.B. "A Simplified Approach to Continuous on-line Tuning of Field-Oriented Induction Machine Drives," *IEEE Trans. on Ind. App.*, May/June 1990, pp.420.

[Lorenz,1990d] Lorenz, R.D. and Novotny, D.W. "Saturation Effects in Field-Oriented Induction Machines," *IEEE Trans. on Ind. App.*, Vol.26, No.2, March/April 1990, pp.283-289.

[Loveday,1982] Loveday, G., "Essential Electronics - an A-Z guide," Pitman, 1982.

[Luk,1989a] Luk, C.K.P., "Microprocessor-based 3-Phase Brushless Permanent Magnet Drive System," *MPhil. Thesis, University of Sheffield*, 1989.

[Luk,1989b] Luk, C.K.P.;Jayne,M.G.;Rees,D., "Use of the transputer for pulse-width modulated (PWM) inverters," *Proceedings of the 24th University Power Engineers Conference*, 1989.

[Luk,1991a] Luk, C.K.P.;Jayne,M.G.;Rees,D., "A Digital Model for a Three-phase Induction Motor Drive using a Personal Computer (PC) Software Package," *5th IFAC Symposium on Computer Aided Design in Control Systems*, Swansea, UK, Pergamon Press, 1991.

[Luk,1991b] Luk, C.K.P.; Jayne,M.G.; Rees,D., "The Transputer Control of Variable Speed Induction Motor Drives," *Proceedings EPE'91, 4th European Conference on Power Electronics*, Vol.1 pp.574-579, Firenze, 3-6 Sept. 1991.

[MATLAB,1990] PC-MATLAB User's Guide, The MathWorks, Inc., 1990.

[Matsuo,1985] Matsuo, T. and Lipo, T.A., "A Rotor Parameter Identification Scheme for Vector-Controlled Induction Motor Drives," *IEEE Transactions on Industry Applications*, Vol.IA-21, No.4, May/June 1985.

[Miller,1989] Miller, T.J.E., "Brushless Permanent-Magnet and Reluctance Motor Drives," *Oxford Science Publicaitons*, 1989.

[Miyazaki,1984] Miyazaki, M.; Kuroiwa, A. and Kudor, T., "High Performance Vector Control Drive System by Microprocessor-Based Current Source Inverter," *IECON '84*, pp.821-826.

[Moynihan,1991] Moynihan, J.F.*et al*, "Indirect Phase Current Detection for Field Oriented Control of a Permanent Magnet Synchronous Motor Drive. *Proceedings EPE'91, 4th European Conference on Power Electronics*, Vol.3 pp.641-646, Firenze, 3- 6 Sept. 1991.

[Nagase,1984] Nagase, H.; Matsuda, Y.; Ohnishi K.; Ninomiya, H. and Koike, T., "High-Performance Induction Motor Drive System Using PWM Inverter," *IEEE Trans. Ind. App.*, Vol.IA-20, No.6, Nov/Dec.,1984, pp.1482-1489.

[Nandakumar,1991] Nandakumar, M.; Baliga, et al, *IEEE EDL-12.*, pp227- 229,1991.

[Nayak,1976] Nayak,P.H. and Hoft,R.G., "Computer-Aided Steady-State Analysis of Thyristor DC Drives on Weak Power Systems," *IEEE Conf.Rec, 1976 11th Annu. Meet. Ind. App. Soc.*, 835-47.

[Novotny,1975] Novotny,D.W.,"Switching Function Representation of Polyphase Inverters," *Conf. Rec. 1975 10th Annu. Meet. Ind. App. Soc.*, 823-31.

[Ohnishi,1986] Ohnishi, K.; Ueda, Y. and Miyachi, K., "Model Reference Adaptive System Against Rotor Resistance Variation in Induction Motor Drive," *IEEE Trans Ind. Electronics*, Vol.IE-33, No.3, August 1986, pp.217-223.

[Plunkett,1985] Plunkett, A.B.; Kliman, G.B. and Boyle, M.J., "Digital Techniques in the Evaluation of High-Efficiency Induction Motors for Inverter Drives," *IEEE Transactions on Industry Applications*, Vol.IA-21, No.2, March/April 1985.

[Profumo,1991] Profumo, F. *et al* ., "Comparsion of Universal Field Oriented (UFO) Controller in Different Reference Frames," *Proceedings EPE'91, 4th European Conference on Power Electronics*, Vol.4 pp.689-695, Firenze, 3- 6 Sept. 1991.

[Robertson,1969] Robertson, S.D.T. and Hebbar, K.M., "A Digital Model for Three-Phase Induction Machines," *IEEE Transactions on Power Apparatus and Systems*, Vol,PAS-88, No.11, Novermber 1969.

[RS,1990] RS Data library, 9085, 1990.

[Sabanovic,1986] Sabanovic, A. and Bilalovic, F., "Sliding mode control of ac drives," in *Proc. IEEE/IAS, Ann. Mtg.*, Oct. 1986, pp.50-55.

[Say,1985] Say, M.G., "Alterative Current Machines," 5th Edition, Pitman, 1985.

[Schang,1990] Schang, L. Behrens, T.,"A first course in electrical and computer engineering- with MATLAB program and experiment", Addision-Wesley Publishing Company 1990.

[Sethuraman,1988] Sethuraman, S.K. and Waheed, M.A., "A Single- Chip Microcontroller Based Real-Time PWM Inverter," *Third International Conference on Power Electronics and Variable-Speed Drives*, Conference Publication No.291, July 1988, pp.390-397.

[Siliconix,1984] Siliconix Incoporated, "MOSPOWER Applications Handbook," 1984.

[Steven,1983] Steven,R.E. "Electrical Machines and Power Electronics," Van Nostrand Reinhold (UK) Co. Ltd.

[Stevens,1985] Stevens,M.J., "Digital Control of High Frequency Pulse-width Modulated Inverters," *PhD thesis*, August 1985, Bristol University.

[Sugimoto,1987] Sugimoto, H. and Tamai, S., "Secondary Resistance Identification of an Induction-Motor Applied Model Reference Adaptive System and Its Characteristics," *IEEE Transactions on Industry Applications*, Vol. IA-23, No.2, March/April 1987.

[Sumner,1988] Sumner,M. and Asher, G.M., "PWM inverter control using the INMOST transputer," *Parallel Processing in Control, the transputer and other architectures*, Peter Pergrinus Ltd., 1988, pp.158-182.

[Tait,1984] Tait,A.J.; "Rapid Simulation of Induction Motors Using a Microprocessor System," Ph.D Thesis, August 1984, Edinburgh University.

[Takahashi,1985] Takahashi, I. and Mochikawa, H., "A New Control of PWM Inverter Waveform for Minimum Loss Operation of an Induction Motor Drive," *IEEE Transactions on Industry Applications*, Vol.IA-21, No.4, May/June 1985.

[Takahashi,1986] Takahashi, I. and Mochikawa, H., "Optimum PWM Waveforms of an Inverter for Decreasing Acoustic Noise of an Induction Motor," *IEEE Transactions on Industry Applications*, Vol.IA-22, No.5, September/October 1986.

[Takeuchi,1968] Takeuchi,T.J.,"Theory of SCR Circuit and Application to Motor Control," Tokyo Electrical Engineer College Press, Tokyo, 1968.

[Varnovitsky,1983] Varnovitsky, M., "A Microcomputer-Based Control Signal Generator for a Three-Phase Switching Power Inverter," *IEEE Transactions on Industry Applications*, Vol.IA-19, No.2, March/April 1983.

[Webster,1991] Webster, M.R.; Levy, D.C.; Harley, D.R. and Woodward, D.R. et al., "A Development System for High Performance Controllers for AC Drives Using Transputer and Parallel Processing," *Proceedings EPE'91, 4th European Conference on Power Electronics*, Vol.1 pp.580-583, Firenze, 3-6 Sept. 1991.

[Williams,1982] Williams, A.B., "The Application of Microprocessors to Pulse-Width-Modulated Inverters," *PhD thesis, CNAA*, 1982.

[Zach,1985] Zach, F.C.; Martinez, R.; Keplinger S. and Seiser, A., "Dynamically Optimal Switching Patterns for PWM Inverter Drives (for Minimization of the Torque and Speed Ripples)," *IEEE Transactions on Industry Applications*, Vol.IA-21, No.4, July/August 1985.

OF TRANSPUTER FOR PULSE-WIDTH MODULATED (PWM) INVERTERS

P.Luk, M.G.Jayne, D.Rees

Technic Of Wales, U.K.

INTRODUCTION

ural sampling and regular asymmetric sam-ng processes are commonly used in pulse-th modulated (PWM) inverter drives and nterruptible power supplies. With the ad-t of today's microprocessors and rocontrollers which are of much higher puting and control power compared with se in the past, the implementations of e sampling processes have gradually been nged from analogue to digital means. ever, most of the literature available s that microprocessor-based systems using e methods for PWM generation (1) are not formed in real-time. The most common tech-ue is to compute the switching angles ff-line', for a particular set of ameters, usually the frequency of the ulating wave, the frequency ratio and the ulation index. The values are then stored memory as a look-up table. The roprocessor simply generates the pulse-th modulated wave by clocking out the cor-ponding values in the table for the given ues of parameters input. The disadvantages such a system is its inflexibility and er-utilisation of the microprocessor.

recently introduced transputer, by INMOS, brought concerns in various areas of l-time control due to its fast speed and ilities to support parallel processing . Although the work associated with AC or control to date, is elementary, it is lieved that the concept of parallel cessing can be exploited in the inherent llelism which exists within a high per-mance AC motor drive system, when advanced trol strategies such as adaptive control needed to be incorporated. The applica-n of transputers is strongly expected to tinue in this area.

is paper presents the initial work com-ted on the application of the INMOS IMS 14 transputer for implementation of the tural sampled and regular asymmetric pled PWM processes. For the natural pled process, a digital-to-analogue con-ter (DAC) was used to produce the modulat-g and carrier waveforms. The comparison of e two waveforms was delegated to a hardware parator, which outputed the modulated veform. For the regular asymmetric sampled cess, a timer circuit was used to output e PWM waveform directly. The programs were th written in OCCAM 2. The experimental sults of the regular sampled asymmetric PWM e verified by the simulation of the veform.

THE TRANSPUTER AND OCCAM

g.1 shows the architecture of the IMS T414 ansputer. It consists of a 32-bit proces-or, 2 Kbytes on-board memory and 4 hardware channel links. The on-board memory provides fast local memory access and helps to solve the problems associated with processor-to-memory bottlenecks. The hardware channels provide point-to-point data communications at a basic rate of 20 Mbytes per second. Fig.2 shows how a pipeline or an array of transputers can be connected by the hardware links of each individual transputer to perform parallel processing.

OCCAM is a low-level concurrent programming language which is also developed by INMOS. It is closely related to the transputer. The transputer was actually designed to enable optimal implementations of OCCAM with respect to concurrency and communications. A distinct feature of OCCAM is the inclusion of an instruction called PAR (abbreviated for PARallel), which allows the programmer to express parallelism in program code.

3. TWO COMMON METHODS OF PULSE-WIDTH MODULATED (PWM) SAMPLING PROCESS

Natural sampling

It is a well-establised sampling method in communication engineering . Early developments of PWM power inverter have been centered on this method because it is intrinsically analogue and can easily be implemented by a analogue comparator. Fig.3 shows the generation of natural sampled PWM. The switching edges of the width-modulated pulses can only be expressed by a transcendental equation as follows (3),

$$t_1-t_2 = Tc/2*[1+M/2*(\sin wt_1+\sin wt_2)] \quad --(1)$$

where Tc is the carrier period,
M is the modulating index, and
w is the angular frequency of the modulating wave.

The solving of eqn.(1) by a digital computer usually involves a number of iterations and it is therefore unrealistic to calculate the switching pulses directly in real-time. Thus, the switching angles for different voltages are usually computed 'off-line' by a computer and the values stored in a look-up table. Comparison of the carrier and modulating waves can either be done by hardware or software means, which are described fully in ref(4).

Regular Asymmetric Sampling

This method of sampling has also been well-recognised in the communication industry before being applied to PWM inverters. The generation of the modulated pulses is shown in Fig.4. Unlike natural sampling, it is inherently a digital method in which the magnitude of regularly spaced samples of the

modulating wave determines the pulsewidth. The width of the n^{th} pulse of a regular sampled PWM waveform, which was used in the OCCAM program to generate the experimental results in this paper, is defined as (4),

$$T_n=(Tc/2)*\{1+(-1)^{n+1}*(M/2)*(\sin A_n+\sin A_{n+1})\} \quad --(2)$$

where $A_n=pi*n/R$, and R is the frequency ratio of the carrier wave to modulating wave.

4. SYSTEM HARDWARE

The transputer system used was the INMOS B004 board hosted by a IBM-AT compatible. Fig.5 shows the hardware schematic diagram for the generation of PWM by natural sampling. Standard INMOS link adaptors were used for external interfacing. TTL buffers were used for link-to-link connections. The buffers, although introduce typical TTL propagation delays, do not affect the speed of communication. Extra protection is also provided for the link on one end should the link on the other end becomes faulty. The multi-channel 12-bit D-to-A converter receives 2 bytes of data in succession , firstly the least significant bit (LSB) , then the most significant bit (MSB). The first 4 bits of the MSB are ignored. This is then followed by a byte word for selecting to which channel the previous 12-bit data is output. The handshaking of data communication is achieved by simple hardware TTL gates. The modulating and carrier waves are inputed to a common analogue differential comparator, which then outputs the natural sampled modulated wave.

Fig.6 shows the hardware schematic for regular asymmetric sampling process. Two link adaptors are used for communications with the programmable timer. One is for data and the other is for control signals. Buffering and handshaking are similar to the previous circuit. The timer outputs the modulated wave directly by timing out the loaded value ,which corresponds to the pulse-width, in its counter register.

5. THE OCCAM PROGRAMS

The OCCAM programs were developed under the transputer development system (TDS) IMSD700D. The program source codes were created by a special full screen structured editor which uses the concept of 'folding up' parts of the program. This provides a very efficient method of navigating around and viewing parts of a large program (5). Fig.7a shows the OCCAM source program at its top level for the natural sampled process . The symbol '...' represents a fold which contains more program lines within. The program layout, as may be seen, serves the purpose of flowcharting. Fig.8a shows the OCCAM source program for regular asymmetric sampled process.

6. SIMULATION OF REGULAR ASYMMETRIC SAMPLING USING PC-MATLAB

The experimental result of the regular asymmetric sampling was compared with that from the simulation of the waveform by a software package called PC-MATLAB. The package provides built-in Fast Fourier Transform (FFT) routines for spectrum analysis and facilities for graphics. Number of samples used for simulation was 1K (1024).

7. EXPERIMENTAL RESULTS AND DISCUSSIONS

The modulating and carrier wave of the natural sampled PWM wave and the frequency spectrum is shown in Fig.7, with modulating frequency(f)=40Hz, frequency ratio(r)=6 and modulation index(m)=1.0. This is compared with the regular asymmetric sampled PWM having the same values of f, r and m , as shown in Fig.8. It is seen that the frequency spectrum for regular asymmetric sampled PWM wave is superior to that of the natural sampled. This is in agreement with much early works which have been done in this area (6). The simulated result for asymmetric sampling is found in close agreement with the experimental result.

8. CONCLUSION AND FUTURE WORK

The results of an initial investigation on the application of transputer in the generation of pulse-width modulated waveforms by two commonly used sampling methods have been presented with verified experimental results. A single-transputer system (B004 board) has been used to generate the results. It is hoped that more work will be focused on using a multi-transputer system in the future, so that inherent concurrency within the system can be realized. Fig.9 shows the generation of a regular sampled, asymmetric, 3-phase PWM wave in parallel, using the 'PAR' instruction, and the results of from the execution of the program. Although the program has been developed in a single-transputer system, it can easily be configured to run in a multi-transputer system. The use of the PAR instruction is also expected to be extended to in a later stage of the project, which is intended to incorporate adaptive control into the system.

(The trademarks belonging to INMOS Ltd. and IBM are acknowledged.)

9. REFERENCES

[1] Tez, E.S., 1988, "Motonic Chip Set: High Performance Intelligent Controller For Industrial Variable Speed A.C. Drives", IEE Conference on "Power Electronics and Variable-speed Drives", Conference Publication Number 291.

[2] Jones, D.I., 1987, "Parallel Processing for Real-timer Control Systems", Proc. IEE Workshop on "Parallel Processing in Control - the Transputer and other Architectures", UCNW, Bangor, 1987.

[3] Bowes, S.R., 1975, "New sinusoidal pulsewidth-modulated inverter", Proc.IEE,122 (11),pp.1279-1285.

[4] Bowes, S.R., 1981, " Microprocessor Control of PWM Inverter", Proc.IEE, Vol.128, Pt.B, No.6, Nov.,1981.,pp.293-305.

[5] INMOS Limited, 1988, " Transputer Development System", Prentice Hall.

[6] Jayne, M.G., 1983, "A Microprocessor-based Control Strategy For PWM Inverters", Proceedings of 3rd IFAC Symposium on Control in Power Electronics and Electrical Drives, Lausanne, Switzerland, September, 1983.

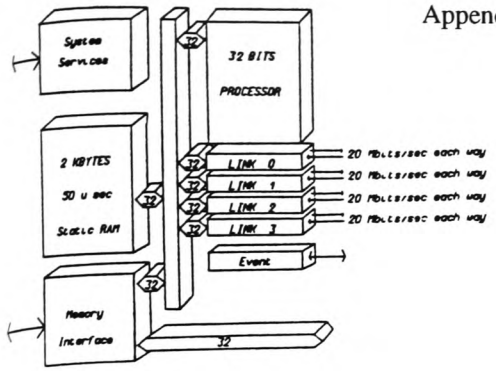


FIG.1 - TRANSPUTER ARCHITECTURE (IMS T414)

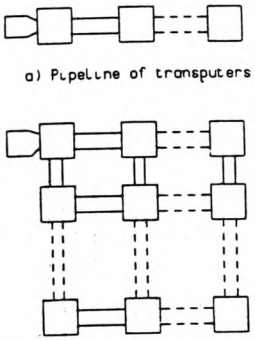


FIG.2 - TRANSPUTERS CONNECTED BY LINKS FOR PARALLEL PROCESSING

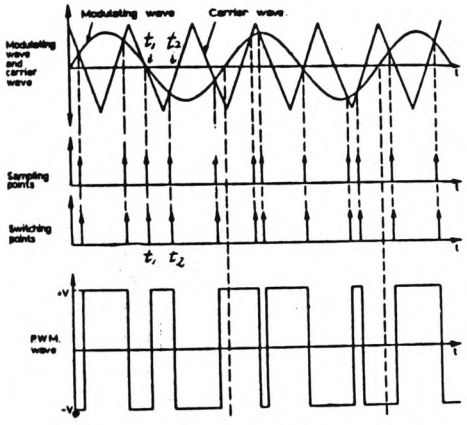


FIG.3 - NATURAL SAMPLED PWM PROCESS

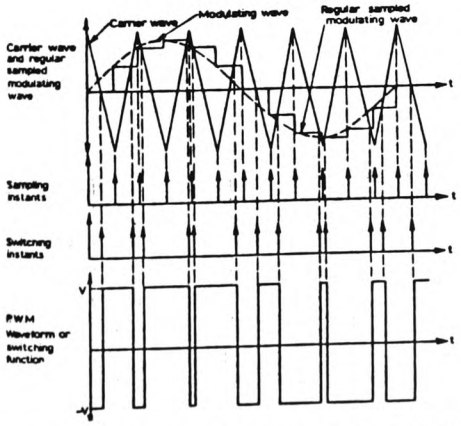


FIG.4 - REGULAR ASYMMETRIC SAMPLED PWM PROCESS

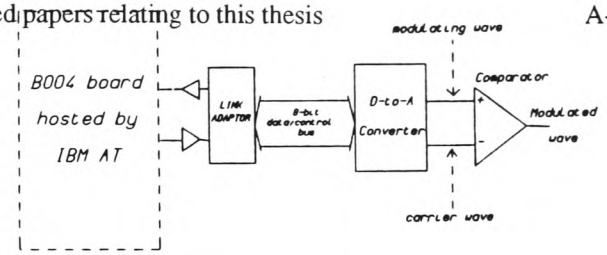


FIG.5 - SCHEMATIC FOR NATURAL SAMPLING

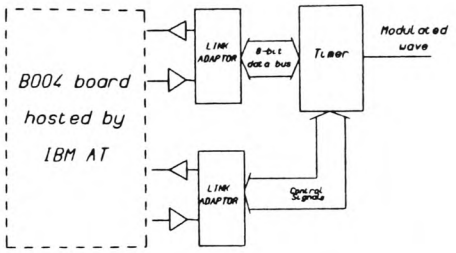
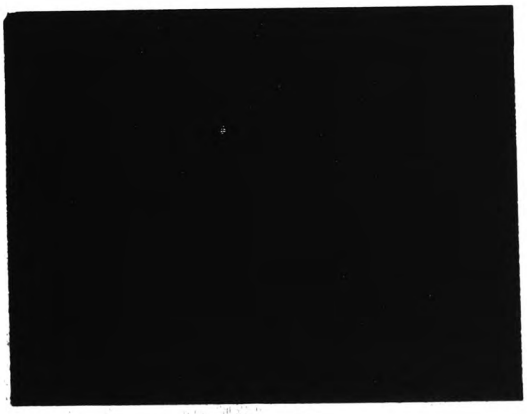


FIG.6 - SCHEMATIC FOR REGULAR ASYMMETRIC SAMPLING

a)

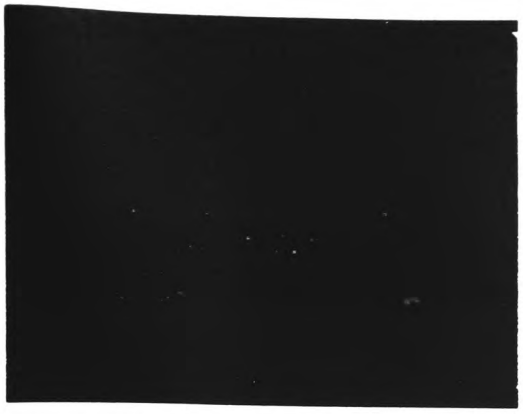


b)

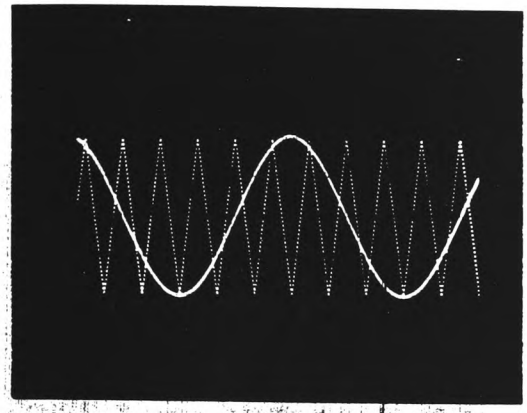
FIG.9 FUTURE WORK

a) OCCAM PROGRAM USING 'PAR' INSTRUCTION

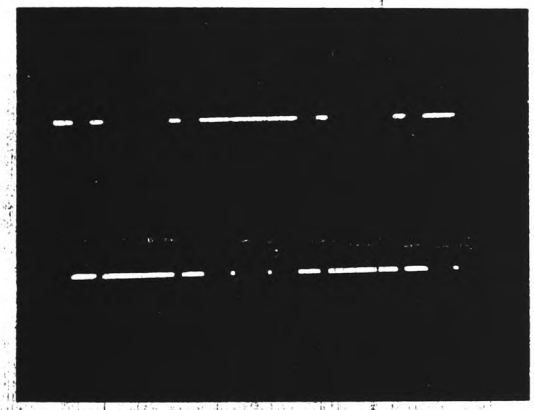
b) EXECUTION OF PROGRAM



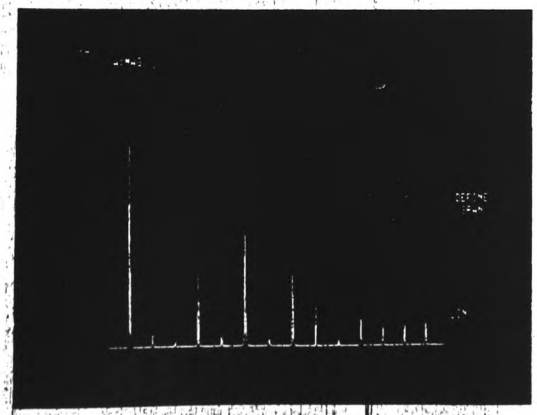
a) OCCAM Source Program



b) Carrier and Modulating Wave



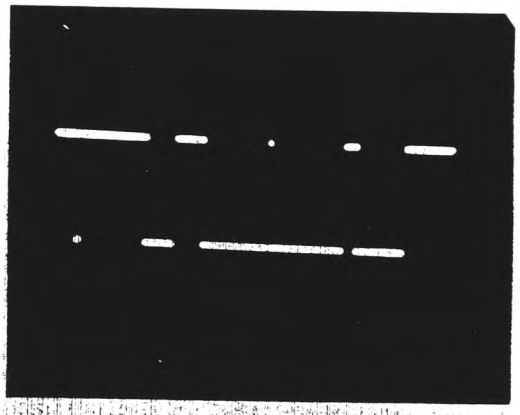
c) Pulse-width Modulated Wave



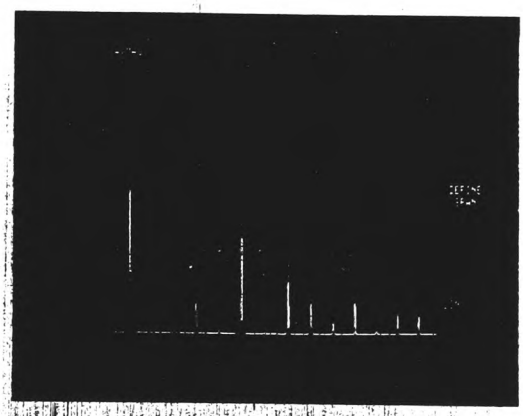
d) Frequency Spectrum

FIG.7 - NATURAL SAMPLING , f=40 Hz, m=1.0, r=6.0

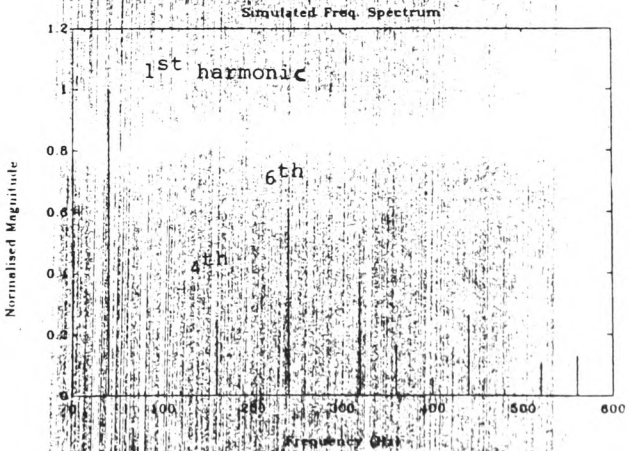
a) OCCAM Source Program



b) Pulse-width Modulated Wave



c) Frequency Spectrum



d) Frequency Spectrum (by Simulation)

FIG.8 - REGULAR ASYMMETRIC SAMPLING , f=40 Hz, m=1.0, r=6.0

A DIGITAL MODEL FOR A THREE-PHASE INDUCTION MOTOR DRIVE USING A PERSONAL COMPUTER (PC) SOFTWARE PACKAGE

C.K.P.Luk , M.G.Jayne, D.Rees, D.W.Schaper

Department of Electronics and Information Technology, Polytechnic of Wales,
Treforest, Pontypridd, Mid Glamorgan CF37 1DL, U.K.

Abstract. A discrete state variable method is used to develop a digital model for a three-phase induction motor drive. It is shown that the computation required in solving the fifth-order non-linear differential equation of the motor is greatly simplified by the exclusive use of matrices. The model can be adapted to the study of steady state as well as dynamic performance of the machine when fed from an inverter. Pulse-width -modulated (PWM) waves are used to test the model which is based on *d-q* transformations of the motor equations. The validity of the model is confirmed by experimental results obtained from a transputer-based PWM inverter drive system.

Keywords. a.c.motors, discrete systems,electric drives, Fourier transforms, matrix algebra, modelling, power converter, simulation, transputer.

INTRODUCTION

The operational advantages of using PWM techniques for the control of induction motors over alternative methods are reflected by the considerable research effort that has been described in the literature over recent years. An essential part of these developments has been the requirement for an efficient and accurate model of the motor in order to examine the performance of a diversity of inverter control strategies. Such a model, results in the general solution of a non-linear, fifth-order differential equation with an input structure which is non-sinusoidal (Bose, 1986). This, together with the subsequent Fourier analysis of various waveforms, requires both good mathematical knowledge and a considerable amount of computing power. Because the majority of a.c. machines in the past were supplied with sinusoidal or near sinusoidal waveforms, frequency domain methods were preferred as system analysis tools (Jacovides, 1973; Murphy, 1976; Jain, 1984). With the number of induction motor drives fed from non-sinusoidal variable frequency supplies increasing, there has been an obvious incentive to use a time domain analysis where all quantities are expressed explicitly instead of the sum of infinite series, as required in the case of frequency domain methods. The classic paper presented by *Lipo* (Lipo, 1977) has shown that even the simulation of a quasi-square inverter in the time domain requires very intensive computation. It has been further demonstrated that (Bowes and Clare, 1983), although accuracy is gained by using time-domain methods, the mathematical complexity involved may dissuade an engineer from adopting such an approach.

In an attempt to demonstrate that the state variable method is one of the most useful means to model a non-linear, multiple-input multiple-output (MIMO) motor model, the authors used a discrete state-variable method which involved only matrix manipulations. With the facilities provided by a personal computer (PC) software package which gives access to a library of subroutines for matrix manipulations, considerable computational simplification is achieved. It is also believed that such a digital model in state-variable form, provides a basis for further development of a modern real-time digital controller in which state-variable methods are commonly used.

MODELLING OF POWER CONDITIONING STAGE

The power conditioning stage is achieved by a 3-phase pulse-width moduated (PWM) inverter. There are numerous PWM generation schemes available (Jayne, Bowes and Bird, 1977). However, only the regular sampled asymmetric modulated PWM waveform is discussed here due to its wide popularity and ease of generation by a digital computer. The method described below for the generation of a discrete PWM waveforms from a continuous one applies to any PWM scheme with known switching angles.

Computation Of Switching Time

The analytical expression for the pulse-width of the n^{th} pulse of the phase *a* of a 3-phase regular sampled asymmetric modulated PWM waveform, is given by (Bowes and Clements, 1982),

$$t_a(n)=(Tc/2)*[1+(-1)^{n+1}*(M/2)*(\sin a_n+\sin a_{n+1})] \qquad (1)$$

where Tc is the period of the carrier wave, M is the modulation index, $a_n=\pi n/R$, and R is the frequency ratio of the carrier wave to the modulating wave. The corresponding pulse-widths for phase *b* and *c* can be found by replacing a_n by b_n and c_n in eqn.(1) respectively, where $b_n=a_n-120°$ and $c_n=a_n-240°$.

It is found that in the real-time generation of a three phase PWM waveform, a more efficient method is to modify eqn.(1) to :

$$t_a'(n) = \begin{cases} Tc/4*[1-M*\sin a_n] & \text{, for n is odd} \qquad (1a) \\ Tc/4*[1+M*\sin a_n] & \text{, for n is even} \qquad (1b) \end{cases}$$

These equations were used in both the simulation and in the generation of the real-time PWM waveforms described in the paper at a later stage.

Discretizing the PWM Waveforms

The choice of sampling period,T_s, is basically influenced by two factors - accuracy and computational time. The higher the sampling frequency, the closer is the discretized model to its continuous analogue model, and the longer becomes the

computational time. On the other hand, the sampling period should be small compared with the dominant time constant of the system. The continuous PWM waveform described by eqn.(1a) and (1b) is discretized by means of the sampling process illustrated in Fig.1(a). The fictitious carrier waveform is shown here for reference. $S(n)$ is the sampling pulse in discrete time domain and $h_a(n)$ is the resulting discretized PWM waveform. Fig.1(b) shows the program written in the simulation programming language (discussed later) for the discretization of the PWM waveform. In the program, S and h_a are matrix vectors. The variable p indicates the corresponding element in the matrix. The first two lines in the program are used to generate the sampling pulse chain matrix vector S. The number of samples (N) over half of the carrier cycle ($Tc/2$) can be chosen by the user. The following sections of the program describe the sampling process. It is evident from Fig.1(a) that, if the pulse-width of the PWM waveform is less than T_s, that pulse may not be sampled. For the regular sampled asymmetric modulated PWM waveform, and in fact most of the PWM schemes, there is no switching at the instants corresponding to the apexes of the (fictitious) carrier waveform if the modulating index is less than 1. Thus, to ensure that those pulses with pulse-width less than the sampling period are sampled, it is necessary not only to have an integral number of pulses per half carrier cycle, but also to synchronize the sampling pulse chain with the apexes of the carrier wave. It was found that in this investigation, if the number of samples taken was 10 per half carrier cycle, a good compromise between accuracy and computational time was acheived.

MODELLING OF MACHINE

The operation of a three phase induction motor, for purposes of analysis, is basically the interaction between two sets of coils with relative motion as shown in Fig.2(a). The associated equations are represented in compact matrix form as,

$$v = Ri + \omega_r G i + L\, di/dt \qquad (2)$$

where R, L and G are $6x6$ square matrices of the winding coefficients; v and i are 6-element column matrices; and ω_r is the rotor speed.

To improve computational efficiency, eqn.(2) is usually transformed into a d-q axis fixed either on the stator or the rotor or rotating in synchronism with the applied voltages. The transformation referred to the stator is chosen and can be divided into two steps as shown in Fig.2(b)and 2(c).

The first step involves the three-phase to two-phase transformation with no change of space-frame, as shown in Fig.2(b). The current transformation from frame (A,B,C) to (D,Q,Γ) for the stator , and (a,b,c) to (α,β,χ) for rotor, can be expressed in matrix forms as,

$$i_{DQ\Gamma} = D\, i_{ABC} \qquad (3)$$

$$i_{\alpha\beta\chi} = D\, i_{abc} \qquad (4)$$

Similar relationships are true for voltages. The second step involves the change of space-frame for the rotor, as shown in Fig.2(c). The current transformation from frame (α,β,χ) to (d,q,χ), can be expressed in matrix forms as,

$$i_{dq\chi} = S\, i_{\alpha\beta\chi} \qquad (5)$$

Combining (4) and (5) gives,

$$i_{dq\chi} = S\, i_{\alpha\beta\chi} = S\, D\, i_{abc}$$

$$or, \qquad i_{dq\chi} = C\, i_{abc} \qquad (6)$$

Similar relationships are true for the voltages.The matrices D, S and C are as defined in the Appendix.

The motor equation in d-q form referred to the stator may therefore be expanded as,

$$\begin{bmatrix} v_{ds} \\ v_{qs} \\ v_{dr} \\ v_{qr} \end{bmatrix} = \begin{bmatrix} R_s+L_s p & 0 & pM & 0 \\ 0 & R_s+L_s p & 0 & pM \\ pM & \omega_r M & R_r+L_r p & \omega_r L_r \\ -\omega_r M & pM & \omega_r L_r & R_r+L_r p \end{bmatrix} \cdot \begin{bmatrix} i_{ds} \\ i_{qs} \\ i_{dr} \\ i_{qr} \end{bmatrix} \qquad (7a)$$

If this equation is put into the form of eqn.(2), the order of the matrices will be reduced by 2 as a result of the d-q transformations. Thus, R,L and G become $4x4$ matrices, and v,i are 4-element column matrices. The electromagnetic torque T_e is given by,

$$T_e = n\, i\, G\, i^T \qquad (7b)$$

where n is the number of pole-pairs and i^T is the transpose of i. Eqn.(7a) and Eqn.(7b) may be combined together to form a fifth-order equation,

$$\begin{bmatrix} v_{ds} \\ v_{qs} \\ v_{dr} \\ v_{qr} \\ T_e \end{bmatrix} = \begin{bmatrix} R_s+L_s p & 0 & pM & 0 & 0 \\ 0 & R_s+L_s p & 0 & pM & 0 \\ pM & \omega_r M & R_r+L_r p & \omega_r L_r & 0 \\ -\omega_r M & pM & \omega_r L_r & R_r+L_r p & 0 \\ i_{qr}M & i_{dr}M & i_{dr}L_r & -i_{qr}L_s & Jp+R_F \end{bmatrix} \cdot \begin{bmatrix} i_{ds} \\ i_{qs} \\ i_{qr} \\ i_{dr} \\ \omega_r \end{bmatrix} \qquad (8)$$

where J is the moment of inertia of the rotating mass, and R_F is a mechanical coefficient representing dissipation due to friction and windage.

DEVELOPMENT OF THE SIMULATION

Simulation Software Package *PC-MATLAB* [1]

The *PC-MATLAB* is a high-level user-friendly programming software package suitable for the design and analysis of control systems. As the abbreviated name (*MAT*rix *LAB*oratory) implies, the software handles data exclusively in matrix form and is hosted by a PC with a maths co-processor. A subroutine library consisting of many useful matrix functions such as Fourier Transforms and the solution of the state-transition matrix in exponential form. It can also import programs written in C or Fortran. The graphic output facility contains standard output format such as Hewlett Packard Graphic Language (HPGL) so that output data can be transferred with great ease.

Solution of State Variable Equation

Digitizing the state variable equation. The machine equation described by eqn.(7a) can be written in standard state-variable form as,

$$\dot{x} = Ax + Bu \qquad (9)$$

[1] *PC-MATLAB* User's Guide, 1988.

where x is the state variable of the transformed current vector and u is the PWM input vector to the d-q frame; and A, B are as defined in the Appendix.

The discrete form of eqn.(9) is given by,

$$x(n+1) = \Phi x(n) + \Gamma u(n) \qquad (10)$$

where, $\Phi = e^{AT_s} = I + AT_s + (1/2!)A^2T_s^2 + \ldots$

$$\Gamma = \int_0^{T_s} e^{A\tau}\, d\tau\, B$$

Since the PWM input $u(nT_s)$ over the period, $nT_s < \tau < (n+1)T_s$, is constant, then,

$$\Gamma = [\, I T_s + (1/2!)AT_s^2 + \ldots \,]\, B$$

The transformation from eqn.(9) to (10) is achieved by a command called *c2d* (continuous to discrete) provided by *MATLAB*. The sampling time T_s is chosen to be the same as in the discretization of the PWM waveform.

Initial condition. The inherent symmetry of the induction motor voltage and current waveforms when supplied from a balanced source can be used to find the initial conditions of eqn.(10). Depending on the symmetry of the waveform, one of the following relations may be used (Bowes and Clare, 1983).

For waveforms with no half-wave symmetry,

$$x(t+T/3) = S_1\, x(t) \qquad (11a)$$

And for waveforms with half-wave symmetry,

$$x(t+T/6) = S_2\, x(t) \qquad (11b)$$

where T is the period of the waveform, and S_1 and S_2 are as defined in the Appendix.

Thus, for a half-wave symmetrical model, if the p^{th} sample is at T/6, then from eqn.(10), the following equations may be obtained.

$$x(1) = \Phi x(0) + \Gamma u(0) \qquad (12a)$$

$$\vdots$$

$$x(i) = \Phi x(i-1) + \Gamma u(i-1) \qquad (12b)$$

$$\vdots$$

$$x(p) = \Phi x(p-1) + \Gamma u(p-1) \qquad (12c)$$

By multiplying equations (12a), ...(12b), ...(12c), with Φ^{p-1},... Φ^{p-i},...,I respectively, and through the process of summation and elimination the following equation for $x(0)$ can be obtained.

$$x(0) = [S_2 - \Phi^p]^{-1} \sum_{i=0}^{p-1} \Gamma \Phi^i u(p-1-i) \qquad (13)$$

Once the initial vector $x(0)$ is obtained, the complete solution for the state equation of the motor currents can then be found as a matter of routine by means of eqn.(10). Fig.3 shows one of the phase currents with the PWM voltage waveform at the sampling points over one-sixth of its cycle. These sampled currents $I(0)$,$I(i)$,$I(p)$ can be found by inverse transformation of the discrete state variables $x(0)$,...$x(i)$,...$x(p)$ by means of eqn.(3).

Spectrum Analysis of the Waveforms

Two Fourier transform commands are provided by *PC-MATLAB* - the discrete Fourier Transform (*dft*) and the fast Fourier Transform (*fft*). Quantities represented in the time domain stored in a row or column matrix T can be transformed to the frequency domain and stored in another matrix, F. Thus,

$$F = fft(T) \quad or \quad F = dft(T)$$

The number of samples to be used in the *fft* should be equal to 2^n (where n is an integer value). A matrix with a number of terms less than 2^n will be filled up with zeros before the transform is performed. The number of samples chosen is restricted by the condition required to synchronize the sampling pulses with the apexes of the carrier waveform (see Fig.1a). This means that if the frequency ratio R is not equal to 2^n itself, then the number of samples N, which is equal to the number of sample per period of carrier wave, say k, times the frequency ratio R, cannot be a 2^n number either. In such cases, k should be chosen such that kxR is near to 2^n to avoid unacceptable errors due to the filled zeros in the matrix. Alternatively, the *dft* may be used at the expense of an increase in the computational time. In this paper, the *dft* was used for values of R not equal to 2^n.

SIMULATION RESULTS

PWM Inverter Voltage Waveforms

The common method of controlling the induction motor by keeping the ratio of voltage-to-frequency (or modulation index to frequency) constant is illustrated in Fig.4 and 5. It may be noted that at higher frequency operations, low frequency ratios (see Fig.5) are used to limit the switching frequency and associated inverter losses. At lower frequency operations, the modulation index is small and voltage pulses are widely separated, resulting in high harmonic distortion. Therefore, higher frequency ratios (see Fig.4) are used to improve the harmonic content of the inverter output voltage . The process of changing from one pulse number per cycle to another is commonly known as 'gear changing'.

Motor Current and Voltage Waveforms

The transformed PWM stator voltages (v_{ds},v_{qs}) and stator currents (i_{ds},i_{qs}) in the d-q frame are shown in Fig.6. It can be shown that v_{ds} represents the line voltage and v_{qs} is a scaled version of the phase voltage (Bowes and Clements, 1982). By plotting the q-quantities against the d-quantities, the locus of the phasor voltage or current waveforms can be obtained. Fig.7 shows the locus of the voltage phasor for the PWM inverter at a frequency ratio (R) of 21. It should be noted that as the frequency ratio increases, the locus of the phasor will approach a circle. Fig.8 shows the typical motor phase currents, which were found by inverse transformation of i_{ds} and i_{qs}.

Spectrum Analysis Of Current And Voltage Waveforms

The harmonic spectrum of the PWM inverter voltage waveforms illustrated in Fig.4 and Fig.5 are shown in Fig.9 and Fig.10 respectively. It can be seen that, by comparing Fig.9 and Fig.10, the magnitude of the fundamental frequency component is proportional to the modulation index. The lower harmonics which are more harmful to the motor, are either greatly reduced or canceled when high values of frequency ratio R are used.

Electromagnetic Torque

The simulation results show that the ripples of the electromagnetic torque generated is largely influenced by the value of the frequency ratio used. The magnitude of the ripples reduces as the frequency ratio increases. Fig.11 shows the waveform of a typical electromagnetic torque when a low frequency ratio is used.

EXPERIMENTAL RESULTS

The results presented in this section were obtained from a transputer-based PWM inverter system (Fig.12). Due to its fast processing speed, the transputer was used to generate a 3-phase PWM waveform in real-time, using the same expressions as described in eqn.(1a) and (1b). The programs were coded in

OCCAM, a programming language closely assocaited with the transputer. The resulting PWM signals were fed to a 3-phase metal-oxide semiconductor field-effect transistor (MOSFET) inverter via an interface circuit. A 2.2kW 3-phase cage rotor induction motor was used for the test. The instrument used to display the results in the *d-q* frame consisted of a 3-phase to 2-phase transformer employing operational amplifiers and R-C circuits. It is shown that the experimental results compare favourably with the simulation results (see Fig.13-16).

CONCLUSION

It has been shown that the solution of the discretized state variable formulated machine model of a 3-phase induction motor is greatly simplifed by the use of matrices. The software package, *PC-MATLAB*, has proved itself as an ideal tool to provide a solution to the problem of machine simulation described in this paper. It allows engineers to analyze machine models in state-variable form without involving considerable mathematical complexity, and also provides computing power at an affordable cost as in contrast to main-frame or mini-computers. It is believed that with the ever-increasing computing power offered by microprocessors, simulation of machine at PC level will become more cost effective and popular in the future.

REFERENCES

Bose,B.K.(1986) "Power Electronics and AC Drives," Prentice Hall.

Jacovides,L.J.(1973) "*Analysis of induction motor drives with noninusoidal supply voltages using Fourier analysis*," IEEE Trans. Ind. Appl., vol. IA-9, pp. 741-747.

Murphy,J.D.M.(1976) "*Analysis of inverter fed induction motors*," Int. J. Elec. Eng. Educ., vol. 13, pp. 359-369.

Jain,G.C.(1984) "*The effect of voltage waveshape on the performance of a 3-phase induction motor*," IEEE Trans. Power App. Syst., vol. Pas-83, pp. 561-566.

Lipo,T.A. and Turnbull,F.G. (1975) "*Analysis and Comparison of Two Types of Square-Wave Inverter Drives*,", IEEE Transactions on Industry Applications, Vol. IA-11, No. 2.

Bowes,S.R. and Clare,J. (1983) "*Steady-state performance of PWM inverter drives*," IEE Proceedings, Vol. 130, Pt. B, No. 4.

Jayne,M.G.,Bowes,S.R. and Bird,B.M. (1977) "*Developments in PWM Inverters*," Proceedings of 2nd IFAC Symposium on Control in Power Electronics, Düsseldorf.

Bowes,S.R. and Clements,R.R. (1982) "*Computer-aided design of PWM inverter systems*," IEE Proc., Vol. 129, Pt. B, No. 1.

APPENDIX

List Of Symbols.

p	=	differential operator
R_s	=	stator resistance
R_r	=	rotor resistance
L_s	=	stator leakage inductance
L_r	=	rotor leakage inductance
M	=	mutual inductance
ω_r	=	angular speed of rotor (electrical)

Matrices D, S, C, S_1 and S_2. The matrices D, S and C used in the transformations of frame are defined as:

$$D = \sqrt{2/3}\begin{bmatrix} 1 & 1/2 & 1/2 \\ 0 & -3/2 & 3/2 \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \qquad S = \begin{bmatrix} \cos\omega_r t & \sin\omega_r t & 0 \\ -\sin\omega_r t & \cos\omega_r t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C = \sqrt{2/3}\begin{bmatrix} \cos\omega_r t & \cos(\omega_r t+120^\circ) & \cos(\omega_r t-120^\circ) \\ -\sin\omega_r t & -\sin(\omega_r t+120^\circ) & -\sin(\omega_r t-120^\circ) \\ 1/\sqrt{2} & 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$$

The matrices S_1 and S_2 used to express the inherent symmetry of the motor waveform are defined as:

$$S_1 = \begin{bmatrix} -1/2 & \sqrt{3}/2 & 0 & 0 \\ \sqrt{3}/2 & -1/2 & 0 & 0 \\ 0 & 0 & -1/2 & -\sqrt{3}/2 \\ 0 & 0 & \sqrt{3}/2 & -1/2 \end{bmatrix} \qquad S_2 = \begin{bmatrix} 1/2 & \sqrt{3}/2 & 0 & 0 \\ -\sqrt{3}/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1/2 & \sqrt{3}/2 \\ 0 & 0 & -\sqrt{3}/2 & 1/2 \end{bmatrix}$$

State Variable Equation For Induction Motor. The equations for the general induction machine model in *d-q* frame with the usual assumptions can be expressed in matrix form as:

$$v = Ri + L\,p\,i + \omega_r G\,i \qquad \text{(i)}$$

where $v = [v_{ds} \;\; v_{qs} \;\; v_{dr} \;\; v_{qr}]^T$

$i = [i_{ds} \;\; i_{qs} \;\; i_{dr} \;\; i_{qr}]^T$

v^T = transpose of the matrix v

$$R = \begin{bmatrix} R_s & 0 & 0 & 0 \\ 0 & R_s & 0 & 0 \\ 0 & 0 & R_r & 0 \\ 0 & 0 & 0 & R_r \end{bmatrix} \quad L = \begin{bmatrix} L_s & 0 & 0 & 0 \\ 0 & L_s & 0 & 0 \\ 0 & 0 & L_r & 0 \\ 0 & 0 & 0 & L_r \end{bmatrix} \quad G = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & M & 0 & M \\ -M & 0 & -M & 0 \end{bmatrix}$$

Rearranging (i) gives,

$$v = (R + \omega_r G)i + Lpi$$

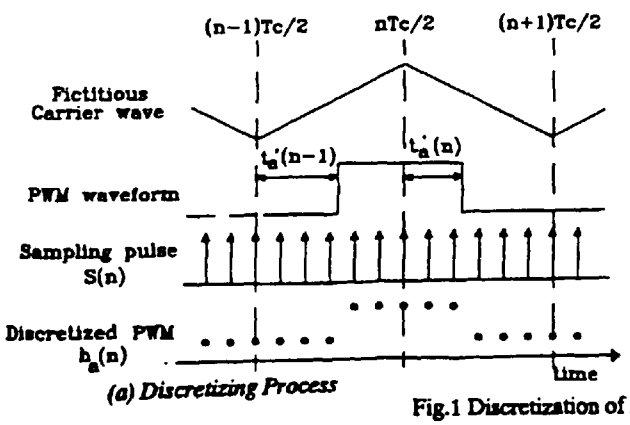
$$\text{or,} \quad pi = -(R + \omega_r G)L^{-1} + I L^{-1} v \qquad \text{(ii)}$$

Thus, by comparing (ii) with the standard state-variable equation (iii),

$$\dot{x} = Ax + Bu \qquad \text{(iii)}$$

it gives A and B, which are defined as in (iv),

$$A = -(R + \omega_r G)L^{-1}$$
$$B = I L^{-1} \qquad \text{(iv)}$$

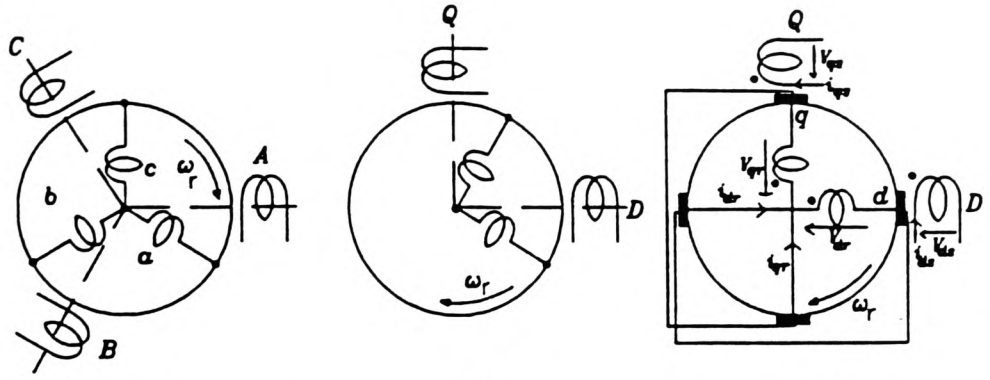


(a) Discretizing Process

```
Ts=(Tc/2)/N;
S=0:Ts:N*Ts*(2*R);
p=1;q=1;
for q=1:2*R;
   for p=N*(q-1)+1:N*q;
      if rem(q,2) >0
         if S(p)<ta'(q)+(q-1)*(Tc/2)
            ha(p)=-1;
               else
            ha(p)=1;
         end,
         else
         if S(p)<ta'(q)+(q-1)*(Tc/2)
            ha(p)=1;
               else
            ha(p)=-1;
         end,
      end,
   end,
end
```

(b) Program

Fig.1 Discretization of PWM Waveform



(a) (b) (c)

Fig. 2 Transformation of Induction Motor Windings to *d-q* frame

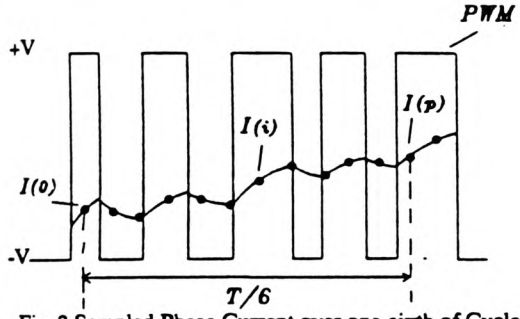


Fig. 3 Sampled Phase Current over one-sixth of Cycle

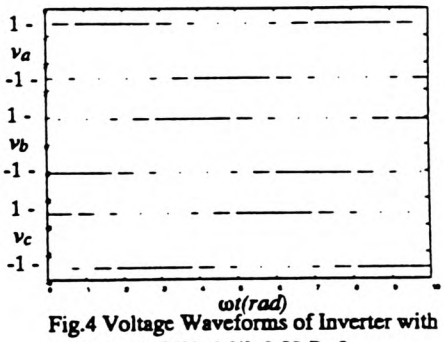


Fig.4 Voltage Waveforms of Inverter with f=50Hz,M/f=0.02,R=9

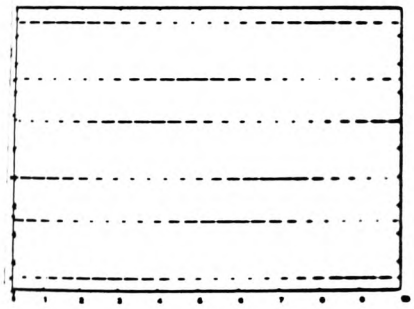
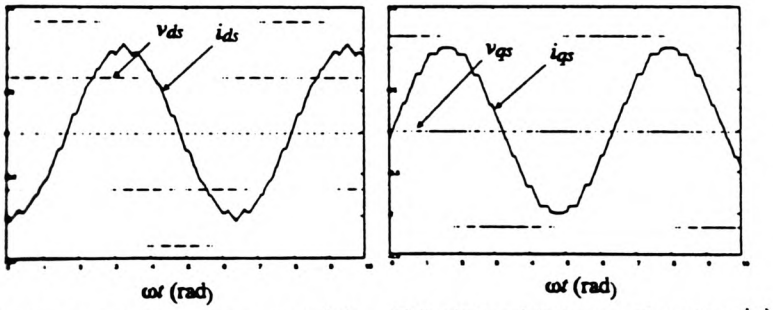


Fig.5 Voltage Waveforms of Inverter with f=35Hz, M/f=0.02 and R=15.



(a) Direct-axis Voltage and Current (v_{ds},i_{ds}) (b) Quadratur-axis Voltage and Current (v_{qs},i_{qs})

Fig.6 PWM stator voltages and currents waveforms in *d-q* frame with f=50Hz, M=0.99 and R=15 and slip=0.02.

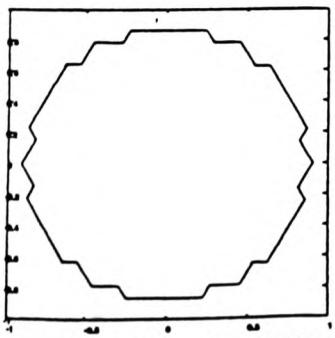


Fig.7 Locus of Stator Phasor of PWM Wavefrom (f=35Hz,M=0.99,R=21)

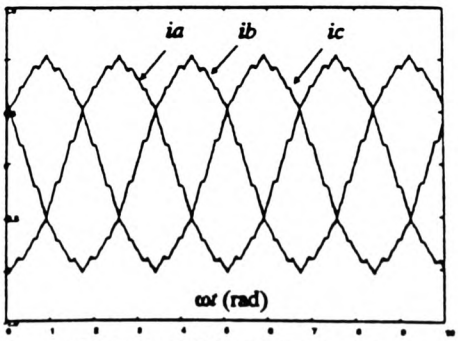


Fig.8 Motor Phase Currents with f=50Hz,M=0.99,R=15,slip=0.2

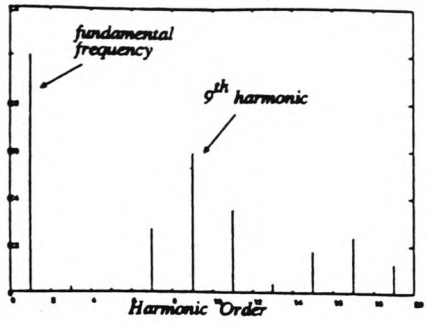


Fig. 9 Harmonic Spectrum of PWM Wave corresponding to Fig.4

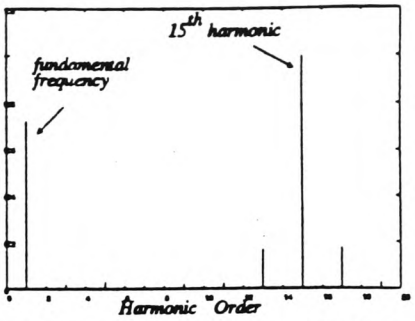


Fig. 10 Harmonic Spectrum of PWM Wave corresponding to Fig.5

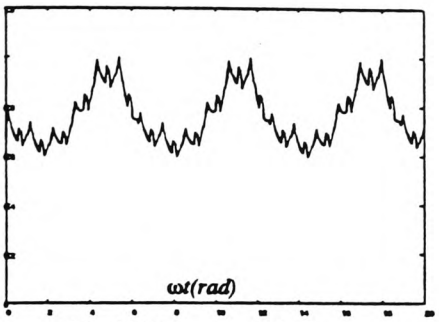


Fig. 11 Electromagnetic Torque (f=50Hz,M=0.99,R=6,slip=0.02)

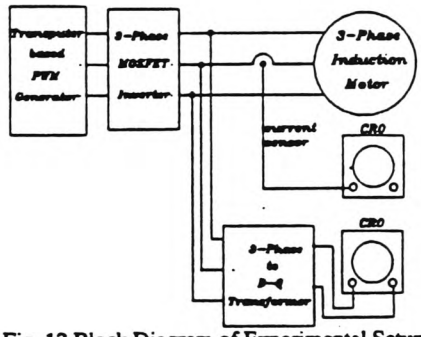


Fig. 12 Block Diagram of Experimental Setup

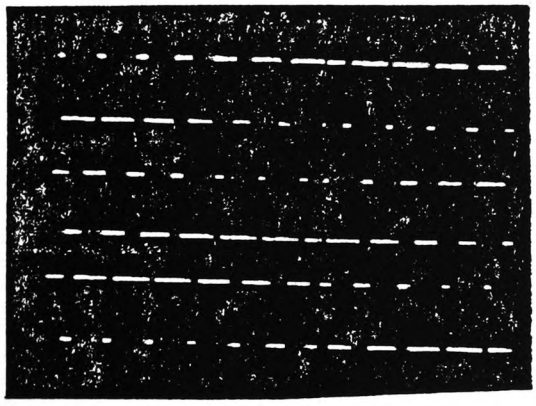


Fig. 13 Experimental Result corresponding to Fig. 5

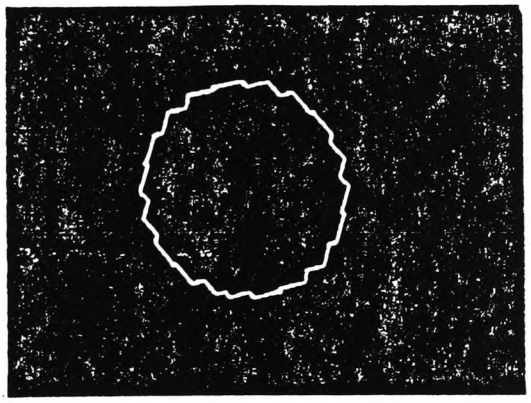


Fig. 14 Experimental Result corresponding to Fig.7

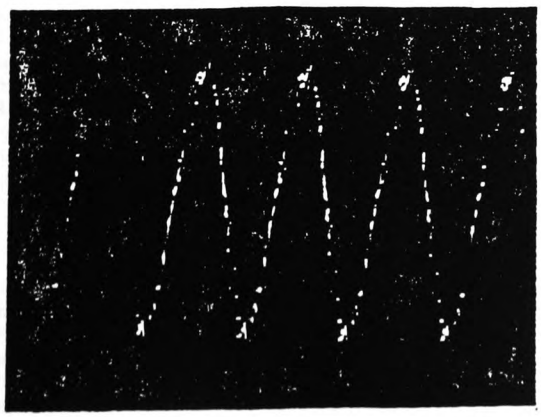


Fig. 15 Experimental Result corresponding to Fig.8

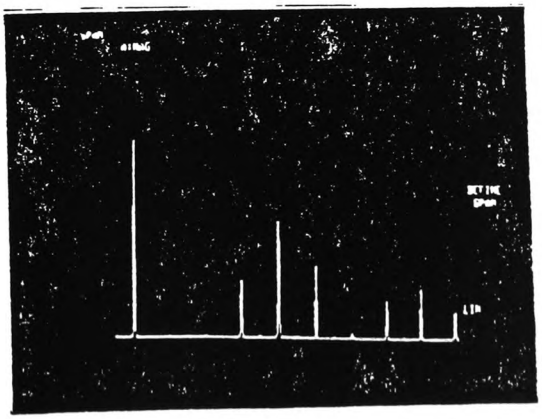


Fig. 16 Experimental Result corresponding to Fig. 9

FIRENZE 1991

THE TRANSPUTER CONTROL OF VARIABLE SPEED INDUCTION MOTOR DRIVES

C K P Luk, M G Jayne, D Rees
Department of Electronics and Information Technology, Polytechnic of Wales, Pontypridd, Mid Glamorgan CF37 1DL, United Kingdom

Abstract. The results of an initial investigation into the application of the transputer to the control of a 2.2 kW 4-pole 3-phase induction motor are presented. A novel method has been developed for the generation of a 3-phase pulse-width modulated (PWM) waveform in real-time by virtue of the fast computing speed of the transputer. Experimental results on the generation of the PWM waveforms and a 'soft-start' process for the induction motor are also included. The results of a simulation study of the motor shows the necessity of the 'soft-start' process. The limitations of the transputer drive system and further developments of the present work are discussed.

Keywords. Variable Speed Drive, Transputer, OCCAM, Pulse-width Modulation (PWM), Parallel Processing, Induction Motor, Vector Control.

INTRODUCTION

When pulse-width modulation (PWM) strategies were first applied to the generation of variable-voltage, variable-frequency (VVVF) power sources for the control of variable speed induction motors in the early 70's, they were mainly implemented by analogue means (Jayne, 1983). By the mid 70's, look-up table methods in which predetermined switching angles were stored in memory, were then gradually developed and used in digital systems. However, the processing speed of the microprocessors available at that time was not fast enough for the generation of PWM waveforms in real-time. As a result, most of the digital PWM systems used the microprocessor to generate PWM waveforms by clocking out the corresponding values in the look-up table for a specified operational condition of the inverter (Dwyer, 1979).

The recent technical advancements and falling prices of power semiconductors and microprocessors, together with an increasing demand for high performance AC drive systems, have resulted in some authors (Kubo, 1985; Harashima, 1985) designing multi-processor based induction motor drive systems. Such systems have the advantages of increased computing power and hence the capability and flexibility to implement complex control schemes. Advanced 'vector control' schemes were launched in these systems and good experimental results were reported. On the other hand, since the processors in these systems shared a common memory, extra design effort was required to avoid the problem of 'bus contention'. Moreover, programming the processors 'in parallel' is relatively more difficult than conventional sequential programming and a high overall efficiency of the multi-processor system may be difficult to achieve (Howe and Moxon, 1987). This partly explains why further developments on the application of multi-processors to drive systems have not been widely reported.

The recently introduced transputer by INMOS has brought concern in various areas of real-time control due to its fast speed and facilities to support parallel processing (INMOS, 1988). Although the work associated with AC motor control to date is elementary, it is believed the concept of parallel processing can be exploited in the inherent parallelism existing within high performance AC motor drive systems when advanced control strategies such as 'vector-control' are to be incorporated (Asher and Summer,1990). The application of transputers is strongly expected to continue in this area. This paper presents the result of the initial work on the application of the transputer in the control of variable speed induction motor drives. The widely used 'Regular Sampled Asymmetric' PWM scheme was employed to generate a three-phase PWM waveform in real-time by the transputer. Experimental results on the 'soft-starting' of a three-phase 2.2 kW induction motor are included.

THE TRANSPUTER AND OCCAM

The emergence of the transputer can be seen as a combined result of the rapid development of the Very Large Scale Integrated (VLSI) devices and the increasing demand for processors which support parallel processing. The name 'Transputer', derived from the two words '**TRAN**Sistor' and 'com**PUTER**', underlines its original design intention. As transistors have been the building blocks of computers, so will transputers be the building blocks of multi-processor systems.

OCCAM, which was developed alongside with the transputer, is a simple concurrent programming language based explicitly on a model of concurrency and communication which is well matched to the architecture of the transputer. Fig.1 illustrates that the close relationship of OCCAM with the transputer in the design of a multi-processor system is similar to that of Boolean algebra with logic gates in the design of discrete digital circuits.

SYSTEM HARDWARE

The simplified block diagram of the complete PWM inverter drive used in this investigation is shown in Fig.2. It consists of the transputer system, the interface, the power conditioning unit and the motor. The transputer system consists of a 'host' transputer mounted inside a personal computer, and two external transputers mounted on a separated board. The interface includes mainly the pulse-width modulation waveform generator and the associated circuitry. The power conditioning unit consists of a three-phase inverter bridge which uses metal-oxide semiconductor field-effect transistors (MOSFET) as the switching devices.

Transputer System

The transputer system consists of a standard B008 board hosted by a IBM compatible personal computer and a B003 board which consists of four T414 transputers as shown in Fig.3. A program called 'server' run on the host computer provides access to the screen, keyboard, and filing system for the TDS. Only two of these four transputers were used to produce the results in this paper. The transputer development system (TDS) runs on the B008 board.

PWM Waveform Generating Circuit

The schematic block diagram for the PWM generator is shown in Fig.4. It consists mainly of two standard INMOS link adaptors (IMSC011), an Intel 8254 programmable interval timer, and buffers/logic gates. The link adapters are both configured in mode 1 as a serial-to-parallel peripheral interface. The upper link adapter (LA1) is used for the output of data to the 8254 timer data bus, whereas the lower one (LA2) is used for the output of control and address signals. The **QValid** (data valid) lines of both link adapters are connected to their respective **QAck** (data acknowledged) lines via two inverter gates connected in series. The propagation delays (typically 20 ns per gate) introduced by the two gates provide appropriate timing for automatic handshaking.

GENERATION OF PWM WAVEFORM

The novel method of generating the three-phase PWM waveforms by means of the transputer is illustrated in Fig.5. The method has proved to be superior to conventional methods when implemented by microprocessors (Midoun, 1985). The pulse-width, T_n, for phase *A* at the n^{th} apex of the carrier wave for a regular sampled asymmetric PWM wave is determined by :

$$T_n = T_c/4*[1+(-1)^n*M*(\sin A_n)] \qquad (1)$$

where T_c is the carrier wave period, $A_n = \pi*n/R$, M is the modulation index and R is the frequency ratio of the carrier wave to modulating wave.

In Fig.5, the triangular waveform (i) represents the fictitious carrier wave with a period of T_c. The waveform (ii) is the regular sampled asymmetric PWM waveform required. The value of the pulse-width, T_n, at the n^{th} apex of the carrier wave is computed and scaled by the transputer on-line by means of eqn.(1) , and the scaled value is then

ted to the 8254 timer at the instant t_n, by means of the timer nding signal (iii) generated by the internal hardware timer of the nsputer. The output of the timer is shown in waveform (iv). To vert waveform (iv) into the PWM waveform required, an inverting nal (v) in synchronous with apexes of the carrier waveform is also erated by the transputer. The inverting waveform and the output m the timer are inputed to the 2-input Exclusive-OR gate (see g.4), which then outputs the final PWM waveform (vi).

OCCAM PROGRAM DEVELOPMENT

e objectives of the program to be developed were, firstly, to oduce a three-phase regular sampled asymmetric PWM waveform a given set of PWM parameters in real-time; and secondly, to hieve a 'soft-start' process by keeping the voltage-to-frequency tion constant during the start-up of the motor. The 'logical model' of e program was first developed in the TDS. The program was then plitted' and the parts of the program were then allocated to different ocessors by a process called 'configuration'. A program flow agram of the three OCCAM processes running concurrently on three parate processors is shown in Fig.6. The process '**monitor**' which is 'EXE' runs on the host transputer, It provides the user inputs from e keyboard and the monitor for the output of results. The values of e PWM parameters, which include the modulating frequency (f), the odulation index (m) and the frequency ratio (R), are inputed to the ocess '**monitor**'. In addition, the number of increment from start-up full speed (N), and the modulation index at the start-up instant (m_0) overcome the voltage drop across the stator windings (or the oltage boost technique') at start-up, are also inputed to '**monitor**'. he process '**monitor**' is then ready to output the data. The processes ontrol' and '**pwm**' which are '**SCs**' run on the processors T0 and f network respectively. The process '**control**' manages the flow data, whereas the process '**pwm**' generates the PWM in real-time. fter the initialisation procedures, the process '**monitor**' is ready to utput the start-data. The process '**control**' is also ready to input the art-up data from the process '**monitor**' whereas the process '**pwm**' is eady to input data from the process '**control**'. As shown in Fig.6, ey communicate with each other via the channels at the instants of nput (symbol '?') and output (symbol '!'). Communications take lace when the processes at both ends of the channel are ready. When he soft-start process finishes, the process '**pwm**' sends a 'full speed' ag back to '**monitor**', which then displays the information on the creen. Each process then runs a looping procedure to detect any input f command, such as a change in the value of the modulating requency (f) or the frequency ratio (R), from the keyboard by the ser. The process '**pwm**' then computes the switching instants on-line nd sends the values to the timer. If the user command is to 'quit' the ystem, the process '**monitor**' will send a 'finish' tag to the network. he network will then respond by sending back an 'acknowledge' tag. he processes will then be terminated properly. In parallel processing, it is important to ensure that all the processes terminate properly.

The OCCAM Program

The transputer development system (TDS) which runs on the B008, provides an environment for the programs to be edited, compiled and executed. The TDS uses a full screen structured editor which uses the concept of 'folding up' parts of the program. This provides a very efficient way of navigating around and viewing parts of a large program. The main part of the OCCAM program is shown below, where the symbols '...' represents program **fold** containing more program lines within. The '**PAR**' command is used to indicate the processes following are run in parallel. The channels of a process are specified by their name in bracket after the name of the process.

```
... channel declarations
PAR
  monitor(host.to.control,control.to.host)
  control(control.to.host,host.to.control,
          control.to.pwm,pwm.to.control)
  pwm(pwm.to.control,control.to.pwm)
```

Configuration of the Transputer Network. The OCCAM program described above is a logical model of the final program if the three different processes under the command '**PAR**' are to be run on three different processors. The description of information, in addition to the main program, about the link topology and the allocation of code to individual processors is called 'configuration'. The OCCAM channels of the above program were mapped with the transputer links as shown in Table 1. The configuration of the network is shown in Fig.7. In the configuration statements, the command **PLACED PAR** was used in place of **PAR**. In addition, a **PROCESSOR** statement and a **PLACE .. AT** statement were used to indicate the type of processor and the address of the physical link used for each process. The main part of the configuration statements is shown below.

```
... channel declarations
PLACED PAR
  PROCESSOR 0 T4
    PLACE control.to.host AT link0out:
    PLACE host.to.control AT link0in:
    PLACE control.to.pwm AT link2out:
    PLACE pwm.to.control AT link2in:
    control(control.to.host,host.to.control,
            control.to.pwm,pwm.to.control)
  PROCESSOR 1 T4
    PLACE to.timer.data AT link0out:
    PLACE to.timer.control AT link1out:
    PLACE pwm.to.control AT link3out:
    PLACE control.to.pwm AT link3in:
    pwm(pwm.to.control,control.to.pwm)
```

It is also instructive to show the top level of the OCCAM program for the transputer network. On this program level, the type of program for each process is shown. Two types of OCCAM programs are recognized by the TDS — the '**EXE**' which runs on the host transputer, and the '**PROGRAM**' which runs on the transputer network (see Fig.7). In the program shown below, the process 'monitor' is an '**EXE**'. The '**PROGRAM**' network contains two '**SC**' programs (separately compiled programs) and the configuration statements of the network. A '**SC**' contains the program codes to be run on a single processor. The symbol 'F' shows that network is a 'filed' **fold** and the symbols '{{{' and '}}}' represent the start and finish of the boundary of an unfolded program **fold**.

```
... EXE monitor
{{{ PROGRAM network
{{{ F  network
...    channel protocol declarations
... SC control
... SC PWM
...    configuration statments
}}}
}}}
```

EXPERIMENTAL RESULTS

PWM waveform generation in real-time

The results of the generation of a three-phase regular sampled asymmetric PWM by the transputer in real-time is showed in Fig.8. Typical values of modulating frequency, modulation index and frequency ratio were used.

Soft-start

The results of a direct-on-line simulation test on the 2.2 kW induction motor are illustrated in Fig.9(a) and Fig.9(b) respectively. Fig.9(a) shows the electromagnetic torque oscillation during the start-up process, whereas Fig.9(b) shows the corresponding inrush current. Such undesirable effects can be reduced by means of a 'soft start' technique. Such a technique was implemented by means of the transputer-based drive system. The transputer computed the switching angles between each frequency increment interval (0.25 seconds) by keeping the voltage-to-frequency ratio constant during starting. A soft-start test on the motor with the voltage/frequency ratio being kept constant is shown in Fig.10. As shown in Fig.10(a), at the start-up stage, both the fundamental frequency of the PWM wave and the modulation index are low. In the middle stage of the start-up which is shown in Fig.10(b), the fundamental frequency and the modulation index have both increased proportionally. In Fig.10(c), the PWM waveform of fundamental frequency of 33 Hz and modulation index of 0.95 is shown. The motor line current at different stages during the 'soft-start' process is shown in Fig.11. It is evident from Fig.11(a)-(c) that the motor line current is roughly constant during the full range of the start-up process. The harmful effect of inrush current is largely eliminated.

CONCLUSION AND FUTURE WORK

An initial investigation into the application of the transputer to the control of a PWM inverter drive has been presented. The fast processing speed of the transputer enabled the PWM waveforms to be generated in real-time. However, the relatively slow communication speed (3μs / byte) of the links was found to be an inferior feature when compared with conventional microprocessors by imposing further limits on the 'minimum pulse-width' of the PWM waveform. It is believed that the new generation transputer, H1, which has a communication speed ten times faster may provide a realistic solution to this problem. A further investigation into the application of 'vector

control' to the present PWM drive system to achieve good dynamic response is already under way. The two 'unused' transputers of the B003 board have been used in the new system to provide extra computing power and additional links for the speed counter circuit and the analog-to-digital converter. The links are mapped with the OCCAM channels as shown in Table-2. The configuration of the extended transputer system is shown in Fig.12. The process **'speed'** which runs on T2, inputs and calculates the speed of the motor, whereas, the process **'current'** which runs on T3, inputs and calculates the phase currents of the motor. The result of this further investigation is expected to be published at a later date.

REFERENCES

Dwyer, E. (1979)."A Lookup Table Based Microprocessor Controller For A Three Phase PWM Inverter", *IEEE / IECI Proc. 5th Ann. Conference Industrial and Control Applications of Microprocessors.* March 19-21,1979

Kubo, K. et al. (1985). "A fully digitized speed regulator using multimicroprocessor system for induction motor drives", *IEEE Trans. On Industry Applications.* Vol. IA-21, No.4, July/Aug 85.

Harashima, F. (1985). " Multimicroprocessor-based control system for quick response Induction drive", *IEEE Trans. On Industry Applications.* Vol.IA-21, No.4, May/June 85.

Jayne, M.G. (1983). "A Microprocessor-based Control Strategy for PWM Inverters", *Proc. 3rd IFAC Symposium on Control in Power Electronics and Electrical Drives,* Lausanne, Switzerland, September.

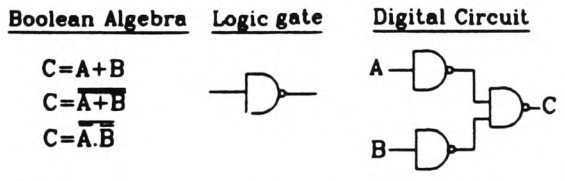
Howe,C.D., Moxon, B.(1987). "How to program parallel processors", *IEEE Spectrum,* September, 1987.

Asher and Summer, (1990). "Parallelism And The Transputer For Real Time High Performance Control Of AC Induction Motors.", *IEE Proc. Vol.137, Pt.D, No.4,* July,1990.

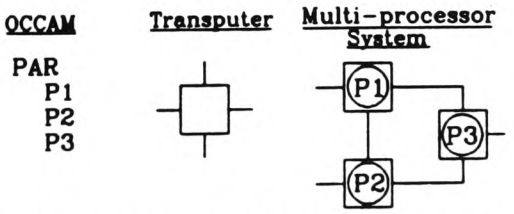
INMOS Limited, (1988)."*Transputer Development System*", Prentice Hall.

Midoun, A. (1985). "PWM Strategies for microprocessor control of variable speed drives," PhD Thesis, University of Bristol, 1985.

Intel Corporation, (1989). *Microprocessor and Peripheral handbook Vol.2.- Peripheral,* 1989.



(a)The use of Boolean algebra in the design of digital circuit consisting of logic gates



(b)The use of OCCAM in the design of multi-processor system consisting transputers

Fig.1 Comparsion of Boolean algebra/Logic gate with OCCAM/Transputer

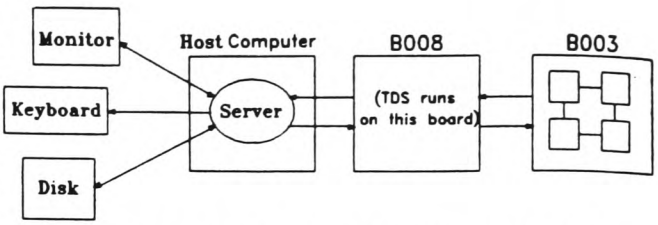


Fig.2 Simplified block diagram of complete drive system

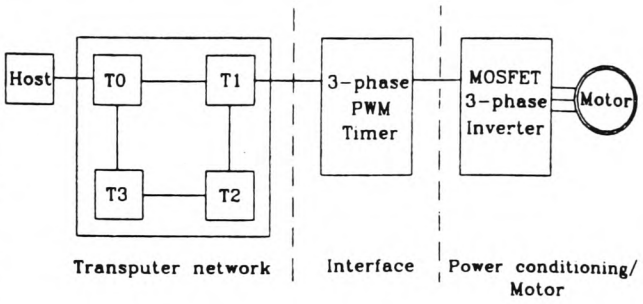


Fig. 2 The transputer system

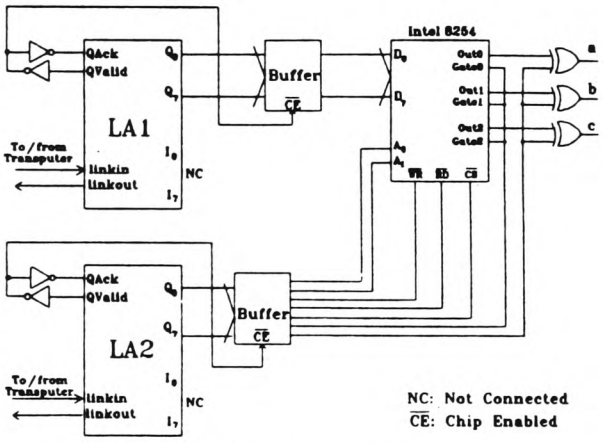


Fig.4 Schematic block diagram of PWM generator

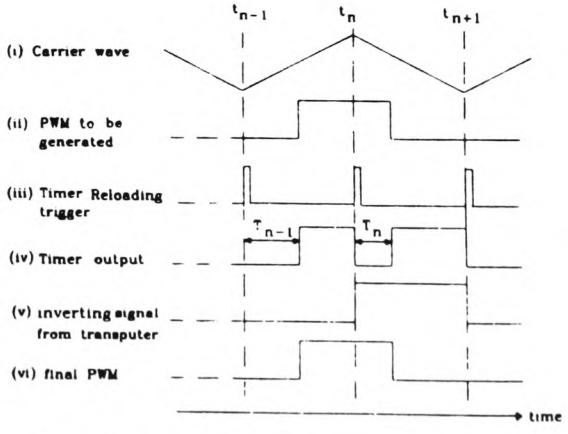


Fig.5 Novel method of generating the PWM waveform

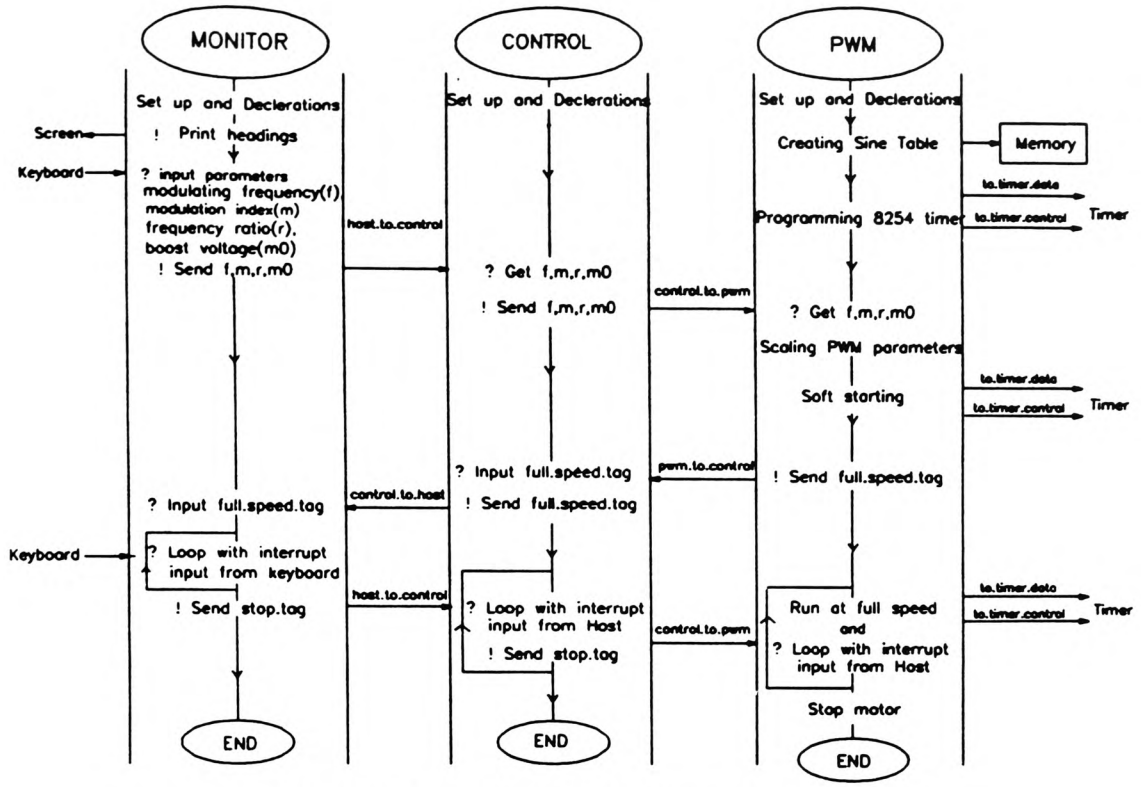


Fig.6 Program flow diagram of the three concurrent OCCAM processes which run on separate processors

Processor	Channel Name	Link Number	Direction
0	control.to.host	0	out
	host.to.control	0	in
	control.to.pwm	2	out
	pwm.to.control	2	in
1	pwm.to.control	3	out
	control.to.pwm	3	in
	to.timer.data	0	out
	to.timer.control	1	out
2 and 3	not configurated		

Table 1 Configuration table of B003 board

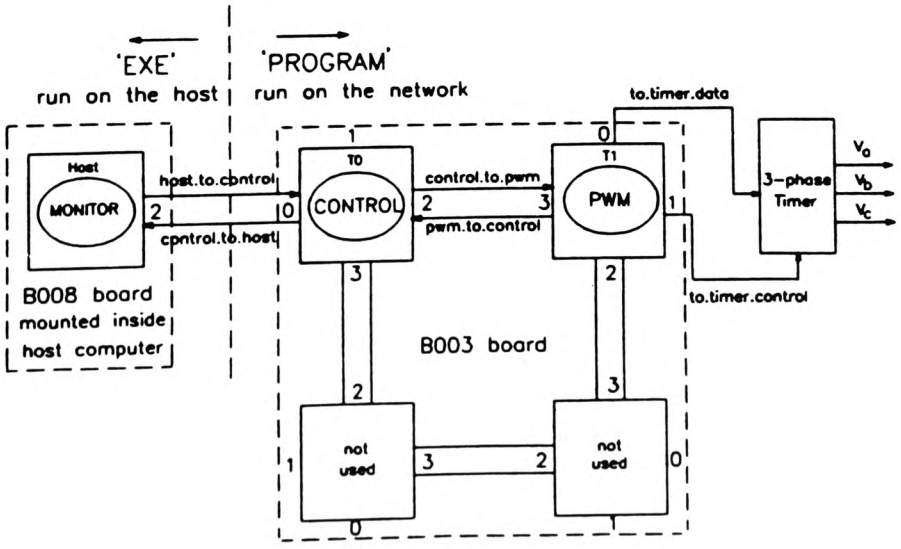
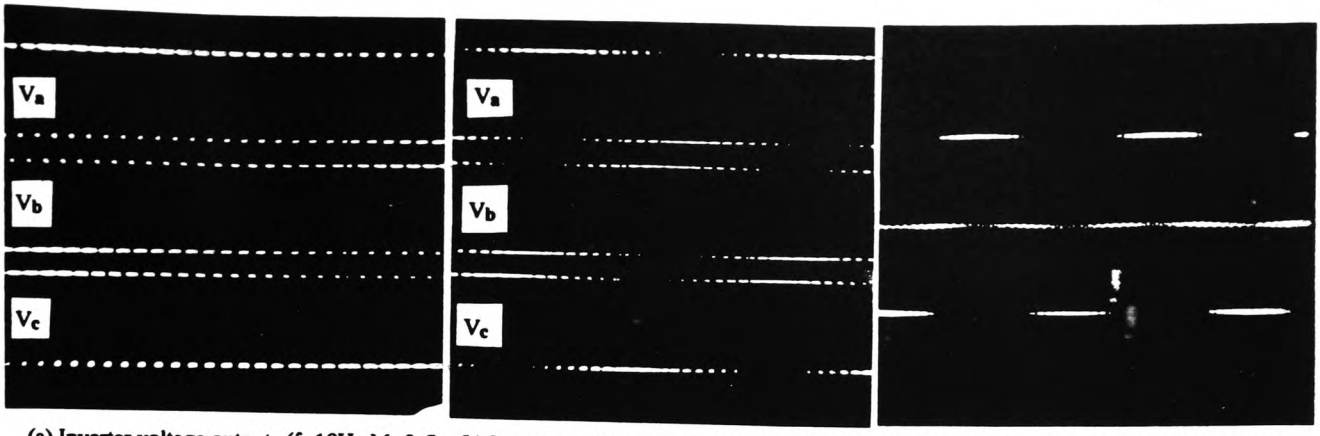
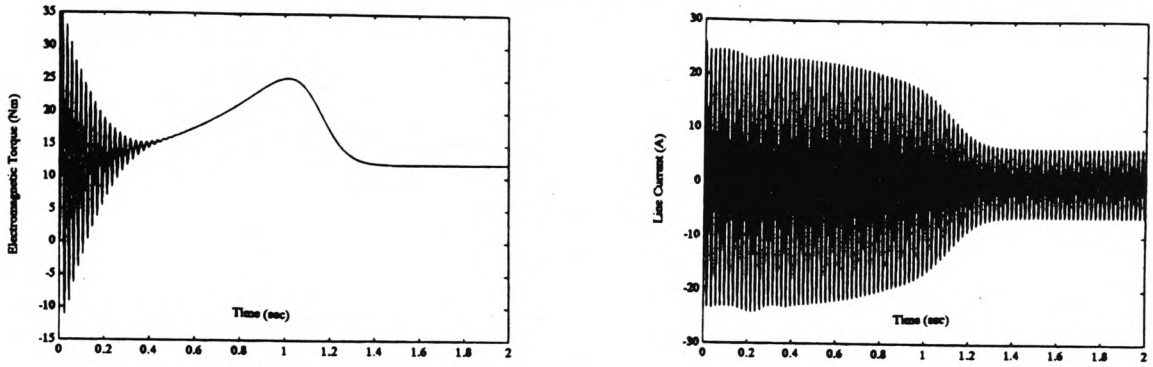


Fig.7 Configuration of the transputer network system



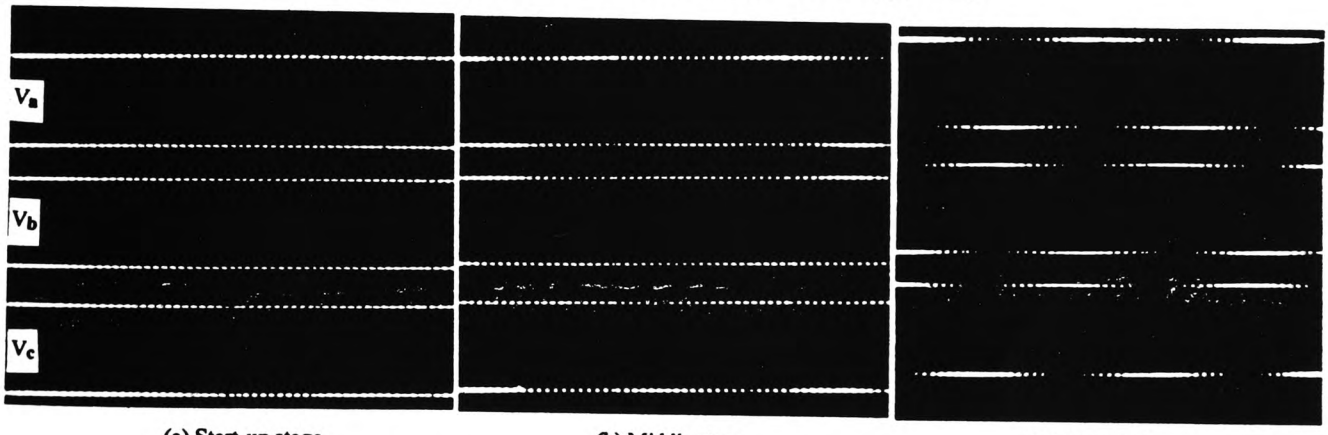
(a) Inverter voltage outputs (f=10Hz,M=0.6) (b) Inverter voltage outputs (f=33Hz, M=0.9) (c) Line voltage of motor (f=33Hz,M=0.95)

Fig.8 Generation of regular sampled asymmetric PWM by the transputer in real-time



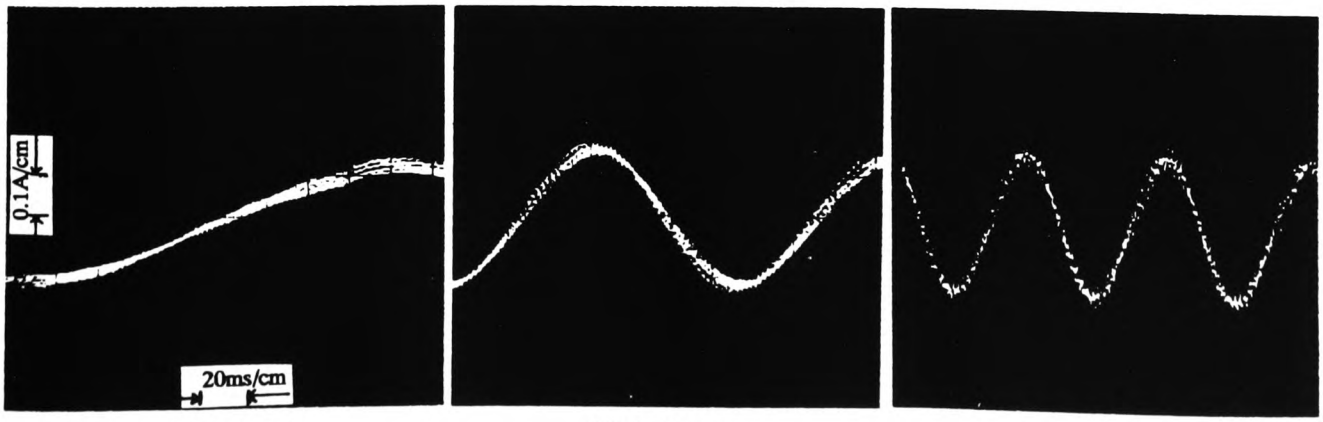
(a) Electromagnetic torque (b) Current inrush

Fig.9 Simulation test on direct-on-line start of the motor at rated supply voltage



(a) Start-up stage (b) Middle stage (c) Full speed

Fig.10 Inverter voltage outputs in the soft-start process
(carrier freq.=600Hz, mod. index at full speed=0.95)



(a) Start-up stage (b) Middle stage (c) Full speed

Fig.11 Motor line current in the soft-start process (carrier freq.=600Hz, mod. index at full speed=0.95)

Processor	Channel Name	Link Number	Direction
0	control.to.host	0	out
	host.to.control	0	in
	control.to.pwm	2	out
	pwm.to.control	2	in
	control.to.current	3	out
1	pwm.to.control	3	out
	control.to.pwm	3	in
	pwm.to.speed	2	out
	speed.to.pwm	2	in
	to.timer.data	0	out
	to.timer.control	1	out
2	speed.to.pwm	3	out
	pwm.to.speed	3	in
	speed.to.current	2	out
	current.to.speed	2	in
	to.counter.data	0	out
	to.counter.control	1	out
3	current.to.speed	3	out
	speed.to.current	3	in
	current.to.control	2	out
	control.to.current	2	in
	to.adc	0	out
	from.adc	0	in
	to.dac	1	out

Table 2 Configuration table of B003 board in the new system

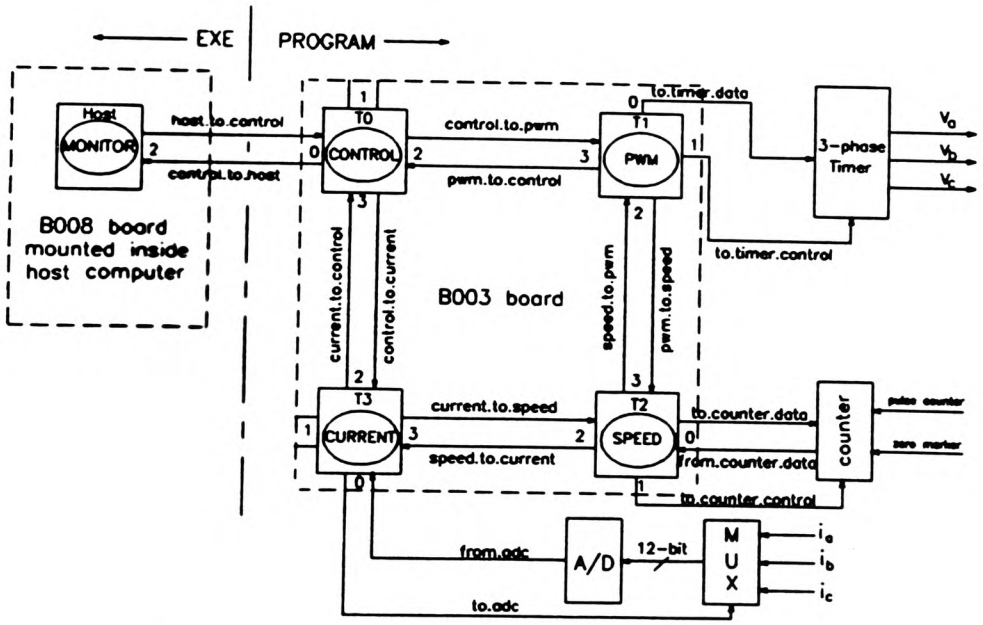


Fig. 12 Configuration of the transputer system for further investigation