SPEED CONTROL OF A MICROPROCESSOR BASED DC DRIVE

Part II: Hardware and Software Implementation

Salah Gh. Ramadan, R. Mostafa, and Gamal Sarhan

I. INTRODUCTION

In this part of study, a complete Hardware and Software realization is given for the efficiency improved speed control dc drive system discussed in [4] and in Part I of this article. The drive is designed to operate in the four quadrant to be suitable with applications requiring frequent accelerating and brake in both direction of rotation. In such cases a considerable part of energy stored in the mechanical load can be recovered to the source of power during brake.

The operating algorithm is arranged such that;

- only two high speed switching elements (MOSFETs) perform all switching duties in the four quadrants to enable using of lower cost devices for the remaining two elements of the four quadrant DC chopper bridge.
- Operating the system at the optimal ratio of (I_a/I_f) at all speeds. This is satisfied in the control system design as one of the control objectives.

The machine language programs of the individual controllers and the complete program of the whole system are experimentally verified. The results obtained are satisfactory.

II. HARDWARE IMPLEMENTATION

The microprocessor system in use is based on 6800 MPU family. It consists of the microprocessor trainer ET3400A of Zenith data systems and a number of interfacing chips and peripherals. The complete design and realization of the data acquisition system is given in Fig.1. In the following sections there is a brief description of each block and how it is function.

A. The Peripheral Interface Adapter (PIA) 6821

This 40 pin chip is used to simplify the interfacing of the micropocessor to the external peripheral devices. It provides the four basic interfacing tasks of address decoding, buffering, latching and timing.

The MPU side of PIA is connected to the system such that the internal registers and the output ports of PIA has the addresses given in table I. The interrupt request lines IRQA and IRQB are connected to the system NMI line. This means that interrupts from PIA will have the highest priority.

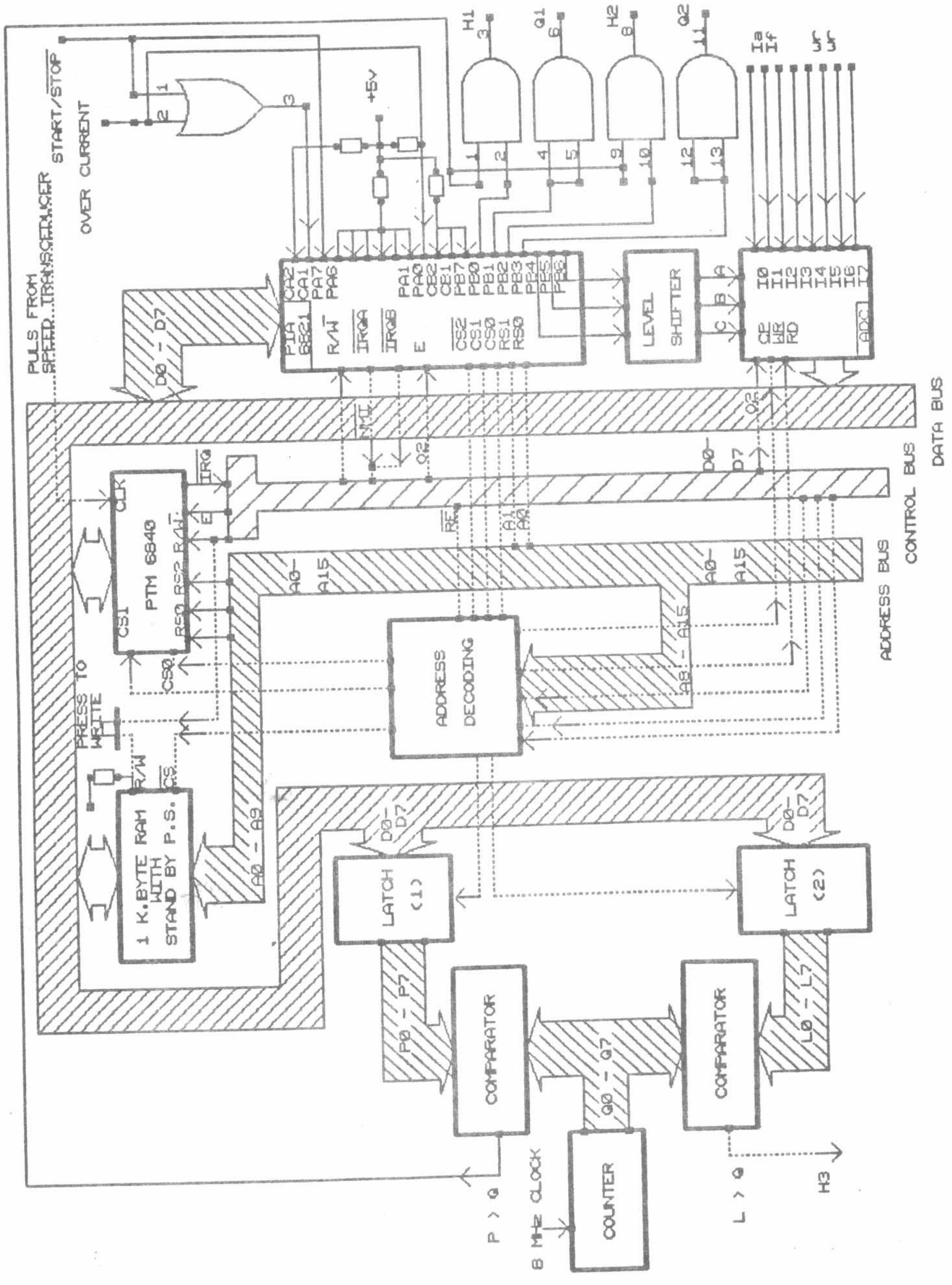


Fig.1 The System Hardware

ADDRESS	STATE OF BIT 2 OF CONTROL REGISTER		REGISTER SELECTED
	CA2	CB2	
8000	0	X	DDRA side A Data Direction Register.
8000	1	X	ORA side A Output Register.
8001	X	X	CRA side A Control Register.
8002	X	0	DDRB side B Data Direction Register.
8002	X	1	ORB side B Output register.
8003	X	X	CRB side B Control Register.

Table I Addresses of PIA registers

The peripheral side of PIA is connected so that:

1. The lines PB0 to PB3 are used as output lines to drive the four power transistors and thus the MPU has full control on power circuit and can change mode of operation of motor by changing the least significant hex digit written to ORB.
2. The lines PB4 to PB6 are used as output lines and this resemble select the lines A, B and C of the analog data selector. This gives the MPU the

   ability to select any of the analog data (I_a, I_f, W_r) to be converted by Analog to Digital Converter ADC.
3. The lines PB7, CB1 and CB2 are used as input control lines, and the control register is configured such that the PIA generates IRQ on +ve transition of CB1 and -ve transition of CB2. These lines are connected to LEFT/RIGHT switch. Thus changing state of this switch generates NMI to tell the MPU about the new direction of rotation.
4. The DDRA is configured such that all lines of ORA are input lines. PA0 monitors OVERCURRENT line and PA7 monitors START/STOP line.
5. The Control register CRA is configured such that the PIA generates interrupts on +ve transition of CA1 and -ve transition of CA2. This tells the MPU whether armature current exceeds its limit or the state of operation is to be changed.
6. The remaining lines of input port (A) are not used and are pulled up to the +5 V.

B. The programmable Timer Module PTM (M6840)

It is a programmable peripheral control device that can perform external timing operations. It can be programmed to generate time delays, control signals one shot and continuous pulses of variable duty cycles,... etc. The main purpose of the PTM (Fig.1) is to allow the MPU to be free of the timing tasks. Once the timer is initialized by the MPU , additional service is not required by the MPU until the timer generates an interrupt. The MPU side of PTM is connected to system such that the internal registers has addresses shown in Table II.

The PTM 6840 contains three timer sections those can operate independently or be cascaded together to provide longer time intervals. In present application the PTM performs two important control functions , these are:

1. The Timer #3 is used to generate continuous interrupts to the MPU every 1.178 ms interval. These are necessary to control the operation since each interval represents one armature circuit sampling period.
2. The Timer #1 is used to count the pulses from the speed transducer over a period of 21 interrupts such that the complement of the number in the counter #1 expresses the motor speed in rad/sec.

ADDRESS	REGISTER SELECTED	
	READ CYCLE	WRITE CYCLE
8200	NONE	CONTROL REGISTER #3 OR #1
8201	STATUS REGISTER	CONTROL REGISTER #2
8202	COUNTER #1	LATCH #1
8203		
8204	COUNTER #2	LATCH #2
8205		
8206	COUNTER #3	LATCH #3
8207		

Table II address of PTM registers

C. The Analog to Digital converter.

The ADC is used in the system to read the feed back signals expressing the field current and the armature current. It is used also to read the input reference speed as a 9-bit word. The circuit diagram of the ADC is shown in Fig.2, where the 8-bit ZN 448 ADC is used together with an 8-channel analog multiplexer 4052. The select lines of 4052 are connected to the lines PB4, PB5 and PB6 of PIA through a level shifter (Fig.1). This enable MPU to select either of the analog input signals.

The ZN448 ADC is a successive approximation type of fast 9 μs conversion time. It is also an MPU compatible and so it is connected directly to data bus. The RD line and WR line of the ADC are connected to address the decoder such that any MPU operation causing 8300 address to be written on the address bus will start the conversion and a next READ operation from the address 8301 will cause the digital value to appear on data bus. As previously stated, the MPU must first select the required analog input signal to be converted.

D. Program Storage

The controller program is stored in a 1k-byte RAM supplied from a standby power supply. It is connected such that it is normally in the read state. To put this RAM in a WRITE state for program writing or subsequent program changes the depress switch connecting the RAM to the R/W line of the system must be pushed down first. This RAM occupies the addresses from 8C00 to 8FFF where the controller programs and associated subroutines are stored.

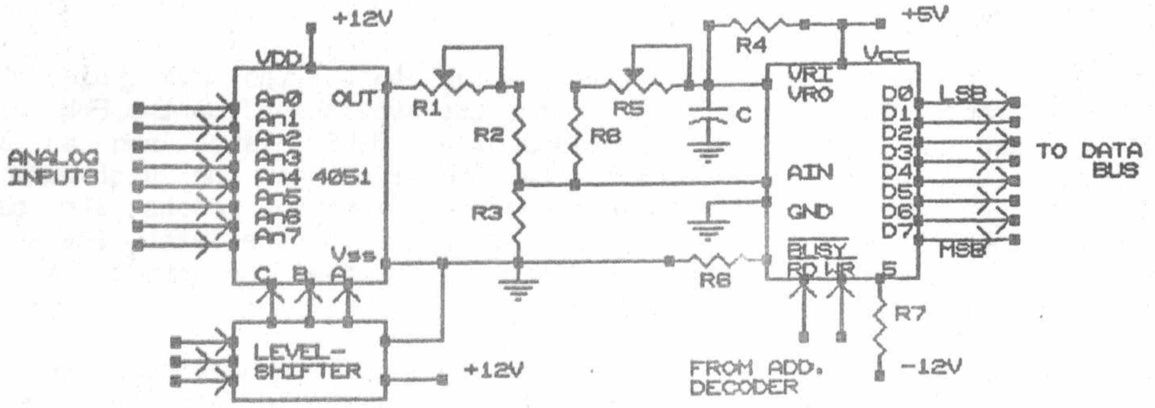
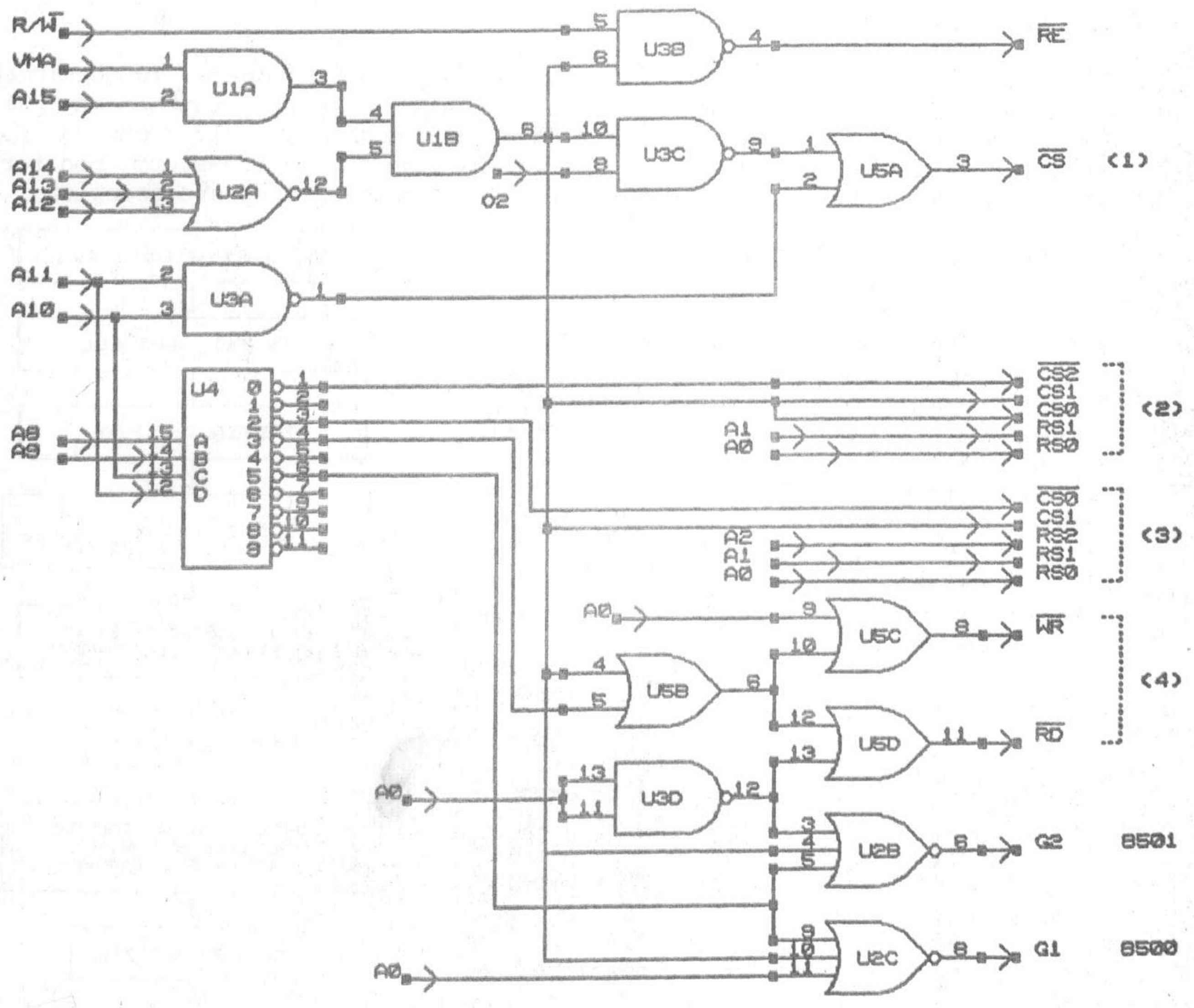


Fig.2 Connection of the Analog/Digital Converter



(1): [illegible]M [8C00-8FFF] (2): PIA [8000-8003] (3): PTM [8200-8206] (4): ADC [8300-8301]

Fig.3 The Address Decoding Logic

E. PWM Output Ports

The output of the microprocessor system which control the duty cycles of the armature chopper and of the field chopper are available from two PWM output ports. Each of them (Fig.1) consists of an 8-bit latch and an 8-bit magnitude comparator which compares the word written in the latch with the output of an 8-bit free running counter. The comparator resets its output line P>Q when the counter output word exceeds that of the latch. The counter is clocked from a 8 MHz crystal oscillator. This gives the output from each comparator at a frequency of 31.125 kHz.

The PWM output ports are connected to address decoder such that the armature port has address 8500 and the field port has address 8501. The MPU can control the duty cycle of the armature or of the field circuit by writing a suitable word in the appropriate port.

F. Address Decoder Logic

As shown in Fig.3, the address decoder of the added hardware is designed such that all these hardware occupy a small space in system address map with a fairly small number of ICs chips. This is necessary to give chance for using usual accessories of such a MPU system such as I/O accessory, monitor, etc. Only 16 k-byte space from address 8000 to address 8FFF are occupied.

III. THE SOFTWARE ALGORITHMS

The MPU control system used is responsible for the overall drive operation including logic operation and signals correction. The logic operations include: the START/STOP, LEFT/RIGHT, and the OVER-CURRENT protections. The signal processing operations are those such as the measuring reference signals, measuring actual signals, implementing correction algorithms etc. In the following sections the general description of drive setup after system hard RESET is given.

A. The System Operation

After the order RESET or when the power is switched ON, the MPU is directed to execute the starting program at the address 8C00. The flow chart shown in Fig.4 illustrates various activities performed. The program initializes all variables, set the NMI vector to 8F90, initialize the PIA and the PTM. Then the field current is maximized by calling the subroutine MAX.I_f. The PTM is reconfigured to generate interrupts every T_a=1.125 ms (sampling period of armature current controller) then program halts waiting start instruction.

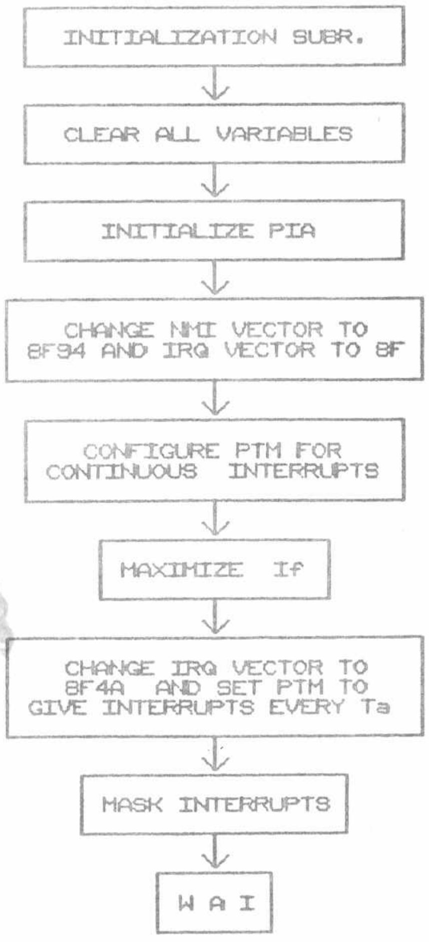

INITIALIZATION SUBR.
CLEAR ALL VARIABLES
INITIALIZE PIA
CHANGE NMI VECTOR TO 8F94 AND IRQ VECTOR TO 8F
CONFIGURE PTM FOR CONTINUOUS INTERRUPTS
MAXIMIZE If
CHANGE IRQ VECTOR TO 8F4A AND SET PTM TO GIVE INTERRUPTS EVERY Ta
MASK INTERRUPTS
W A I

Fig.4 Initialization

Moving the START/STOP switch to the START position generates a NMI to the MPU. The NMI routine directs the MPU to execute the speed controller routine. During execution interrupts from the PTM are enabled. Servicing of these interrupts performs and coordinate the normal control system activities, so an explanation of the IRQ routine is necessary at this point.

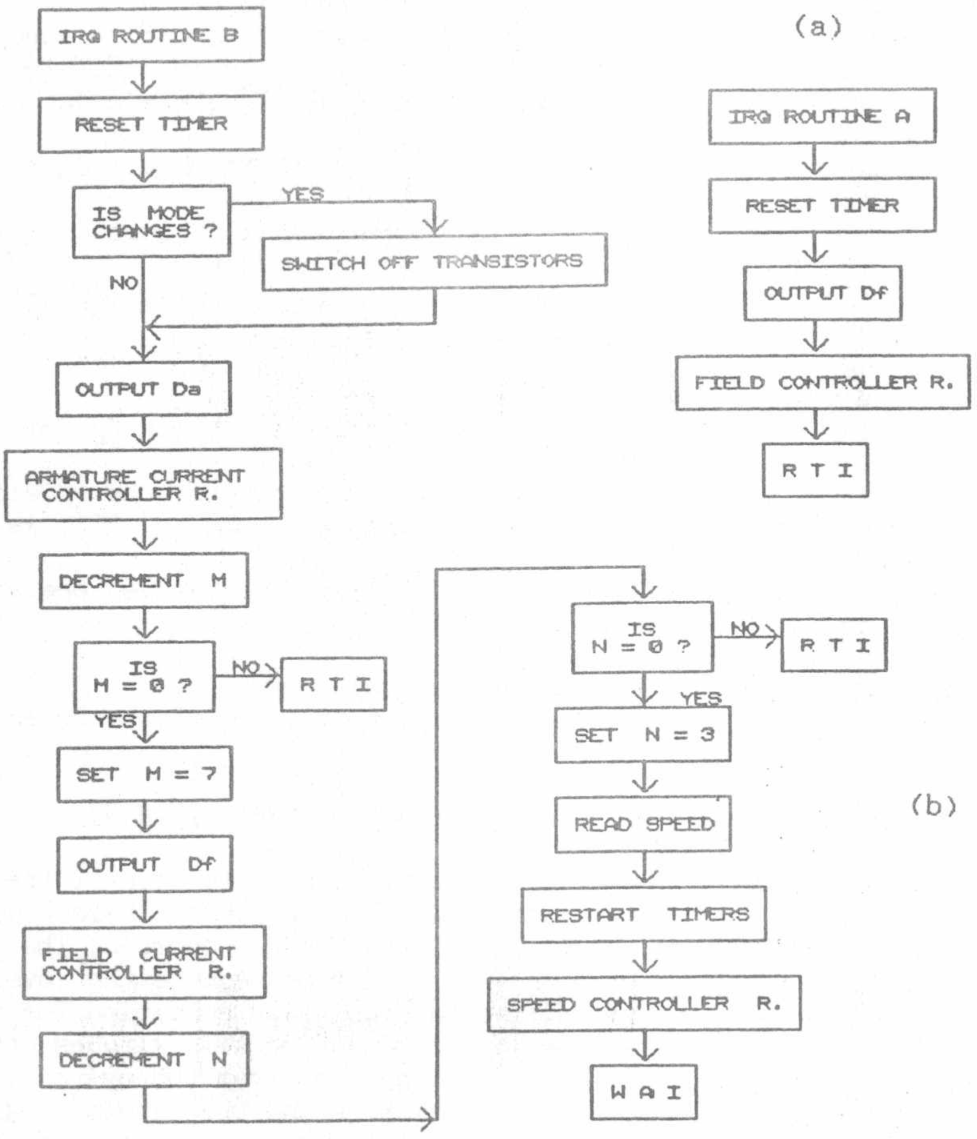


Fig.5 IRQ Routine

B. The Interrupt Request Routine

Two interrupt request routines are handled according to the drive state. The IRQ1 routine is handled in conjunction with the field maximizing subroutine. Its flow chart is shown in Fig.5a. The IRQ2 routine is handled in the normal drive operation. The flow chart of Fig.5b illustrates various activities of this routine. It coordinates the main controller loops as the speed control loop, the field current control loop and the armature current control loop. The only source of interrupts is the timer #1 of the PTM while interrupts from the other timers are disabled. The following activities are performed at different rates;

1. Every time the MPU receives an IRQ the following operations are done;
   a. Resetting timer by reading counter #1 register.
   b. If the mode of operation is not changed the armature chopper duty cycle is updated, otherwise the power is switched off to give the transistors a chance to be completely driven off before mode changing in next IRQ.
   c. The armature current controller is serviced to obtain D_a for the next sampling period.
   d. The interrupts counter M is decreased, if it is zero a return from interrupt is executed otherwise the routine continues.
2. Every 7 consecutive interrupts ($T_f=7T_a$) all the operations are executed in addition to;
   a. Resetting interrupt counter M to 7.
   b. Updating field current duty cycle D_f.
   c. Executing the field current controller to obtain D_f for the next field sampling period.
   d. Decreasing again the counter N which counts the field current sampling periods. If this counter becomes zero a RTI is executed otherwise routine continues.
3. Every three field samples ($T_n=3T_f=21T_a$) all the above operations are performed in addition to the following;
   a. Resetting the counter (N) to 3.
   b. Reading the counter #3 to obtain actual speed.
   c. Executing the speed routine and wait till end of the speed sampling period.

C. The Non Maskable Interrupt requisite routine (NMI) [Fig.6]

All the external instructions that require immediate care of the MPU are handled such that they generate a NMI to the MPU. These instructions include; OVER CURRENT, START/STOP, and RIGHT/LEFT controls. The first of them produces the highest priority NMI. This is because the over current condition is the most severe and harmful operating condition for the power circuit. When the MPU receives a NMI, it first of all switches off all the power transistors since this interrupt may come from the overcurrent line. Then the MPU reads the PIA control registers and the output registers to discover the source of the interrupt and jump to execute the suitable NMI routine. As shown in Fig.8 there are five NMI routines as following;

1. **Over Current Routine**, this routine keeps the transistors off and display a clarifying message. Then it sets the interrupt flag and halts. This blocks any IRQ that may switch the transistors on. To restart the system, it must be RESET first and initialized again at address 8C00
2. **Stop Routine** , this routine displays a clarifying message and then jumps to BRAKE SUBROUTINE to brake the motor speed to zero. Then the program blocks the interrupts and halts. To overcome this halt state the START/STOP switch must be returned to START position.

3. **Start Routine**, it displays also a clarifying message (Start.), and define the starting of the speed controller routine.
4. **Right to Left Routine**, a clarifying message is displayed and BRAKE SUBROUTINE is called to reduce speed to zero. Then MPU jumps to the start routine to accelerate the motor in the reverse direction.
5. **Stray Interrupt**, stray or noise interrupt may be caused by coupling of any external noise to the system NMI line. The MPU discovers this interrupt by examining the interrupt flags of the PIA. When neither of these flags are set, the MPU decides that the received NMI is a stray one. It ignores this interrupt and returns to the routine which was being executed before receiving this interrupt.

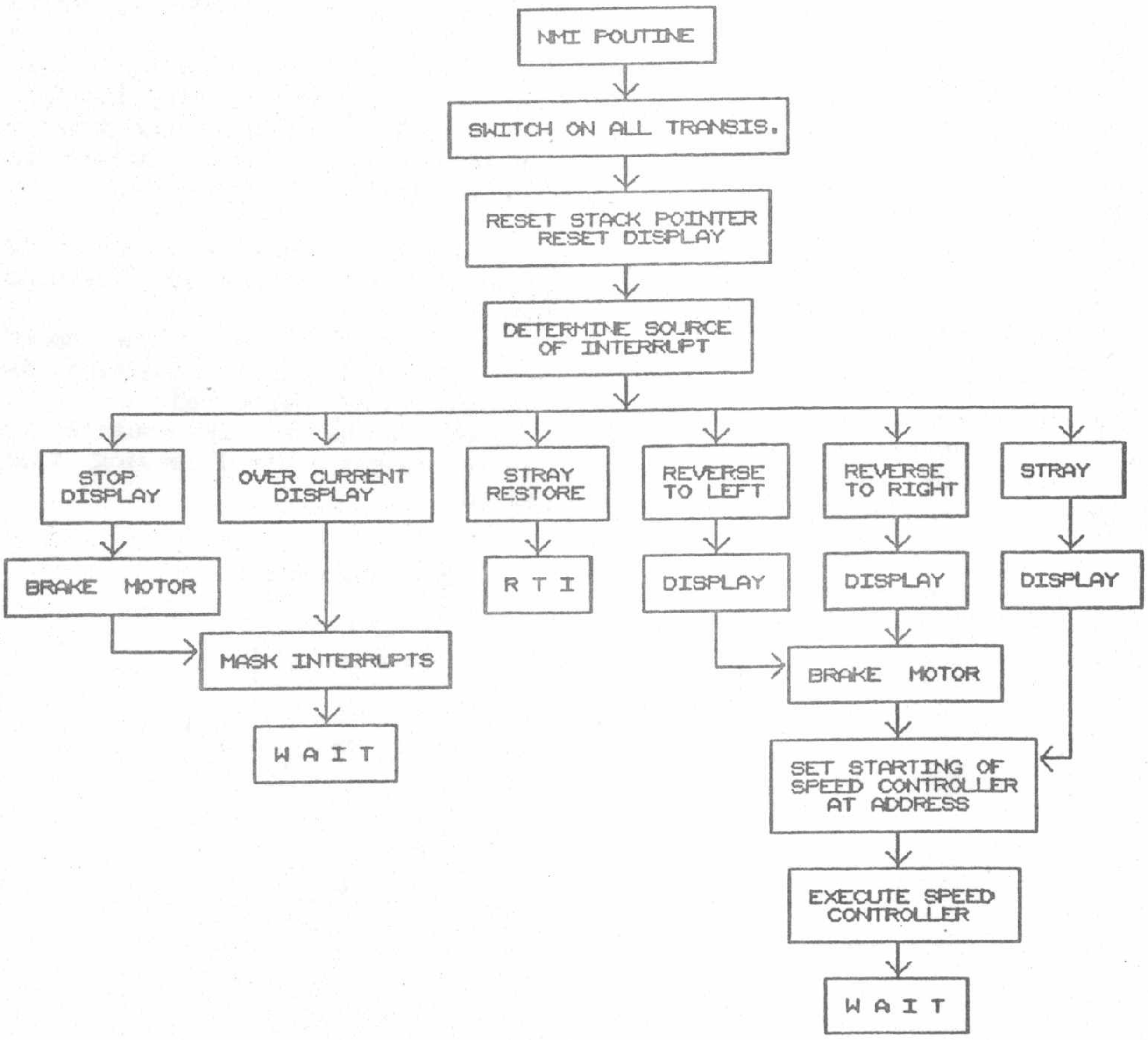


Fig.6 The Nonmaskable Interrupt Routine

D. Non Periodic Routines

These routines are not periodic but are called in transient conditions.

1. **Initialize Subroutine** starts at address 8C00 and performs the activities illustrated in Fig.4 and explained in the following:
   a. It clears all memory locations that are used as variables in different controllers.
   b. It sets the starting address of K(n) look up table.
   c. Configure the PIA for normal operation.
   d. Sets address of NMI routine.
   e. Maximizes the field current.
   f. It reconfigures the PTM to generate normal interrupts every 1.124 ms required to coordinate the normal controller activities, then it waits for the start instruction.
2. **Field Current Maximizing** Routine is shown in Fig.7. It selects IRQ1 routine, sets the field current reference to its maximum by clearing X_{ef}. The PTM is configured to give interrupts every T_f, thereafter the program waits for new interrupt. On receiving interrupt the IRQ1 routine is executed to adjust the field current according the new reference.

   After each interrupt the value of e_f is tested to see if the field current reaches the steady state. If this is the case, the MPU returns from the subroutine otherwise it continues.
3. **Brake Subroutine** is called in situations requiring reducing of the speed to zero. These are stopping condition and reverse direction condition. As it can be seen in Fig.8 the following activities are performed:
   a. Directing the last jump at address (8F8F) of the main IRQ routine to address (8C8F) in brake routine. This is to prevent control leaving this routine at end of the speed sampling period.

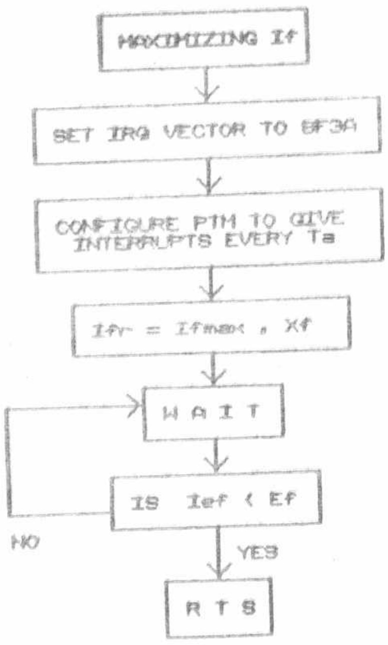


Fig.7 Field Maximization

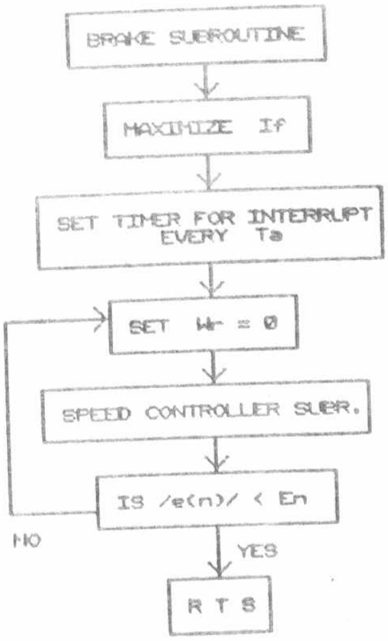


Fig.8 Brake Subroutine

b. Maximizing field current.

c. Enabling normal speed control operation with the speed reference $W_r=0$. The speed controller is started at point C (address 8CBE) to prevent reading of the external reference speed. The routine checks the logic variable (E) of the speed controller to see if the speed error is small enough to consider that the motor has been stopped. If this is the case subroutine switches off all transistors and returns to the calling routine.

E. The Periodic Routines

These routines represent subroutines of the main IRQ2 routine which are called at different points with different sampling rates. This routines constitutes digital controllers in the three control loops, that are speed controller, field current controller and armature current controller.

1. The Speed Controller Routine

The flowchart of the speed routine is shown in Fig.9. This routine has three calling points. The first point (A) includes some activities that are of non periodic nature. The speed routine is started at this point when the controller is to be executed for the first time in the starting condition. The calling point (B) is the normal start point where all activities are of periodic nature. Usually the control is directed to this point from the IRQ routine at start of each speed sampling period. Calling point (C) is also of periodic nature and it represent the part of activities in (B) after measuring the reference speed. The MPU is directed to this point from BRAKE

SUBROUTINE. In general the periodic part of this controller is repeated at a sampling rate of T_n=24.243 ms.

2. The Field Current Controller

The flow chart of the field current controller is shown in Fig.10. This controller occupies statements at address 8DFF to 8E9C, the maximum execution time of these statements is 721 µs. This subroutine starts reading value of I_f by the ADC and then calculate the field current error as;

$$e_f = I_{fr} - I_f + X_f$$

where X_f is an output of efficiency optimization routine. Then it performs PI correction on this error to obtain the field chopper duty cycle. This routine limits I_f by setting D_f to certain minimum value if I_f exceeds a certain maximum limit. The field current reference I_{fr} is externally written before drive setting up.

3. Armature Current Controller

The flowchart of armature current controller is shown in Fig.11. It is very similar to field current controller with the exception that the armature current error is given by;

$$e_a = I_{ar} - I_a - X_a$$

I_{ar} is the output of speed controller while X_a is the output of efficiency optimization controller. The maximum execution time is also 721 us. The output of this controller is the armature duty cycle D_a.

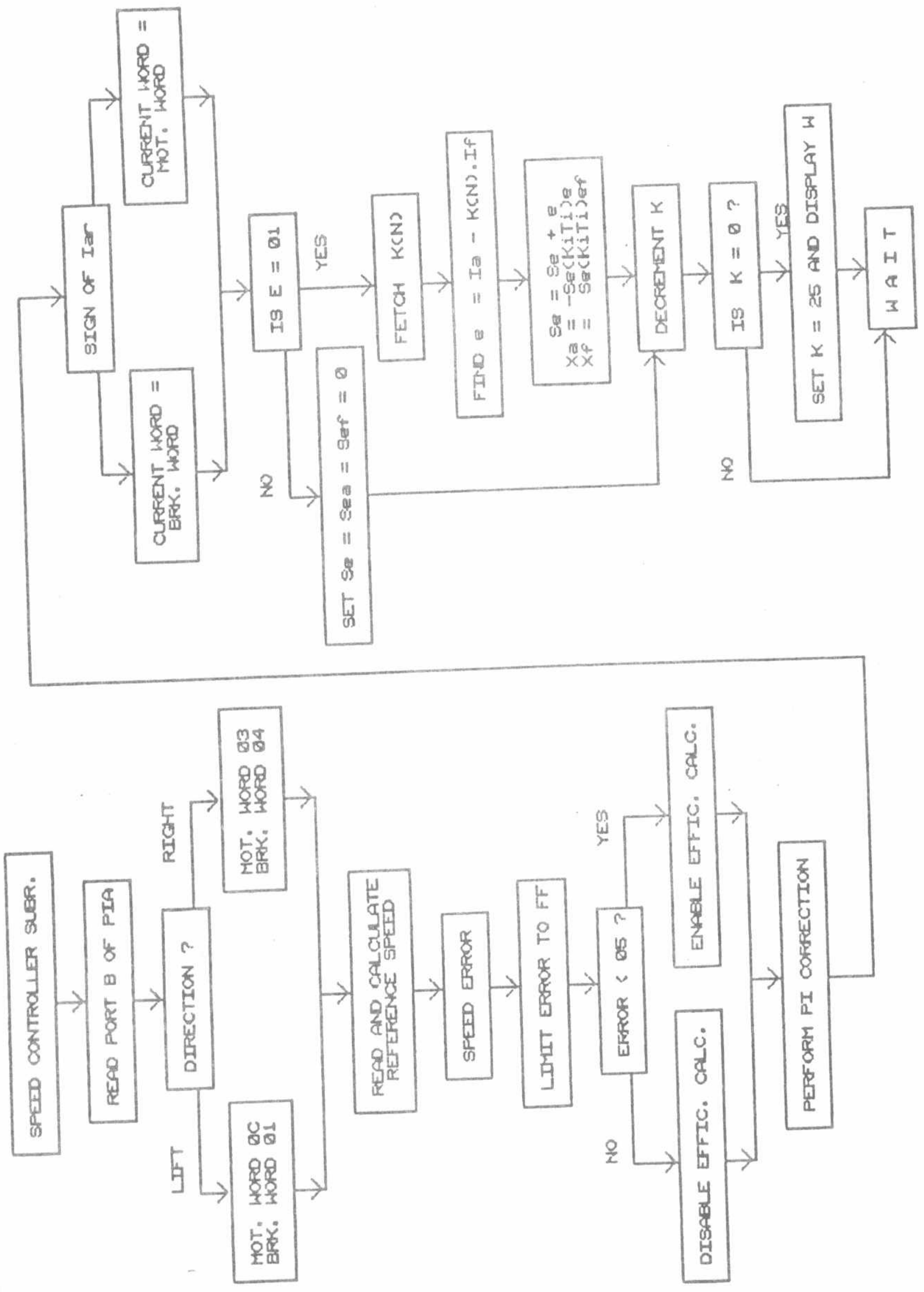


Fig.9 Speed Controller Routine

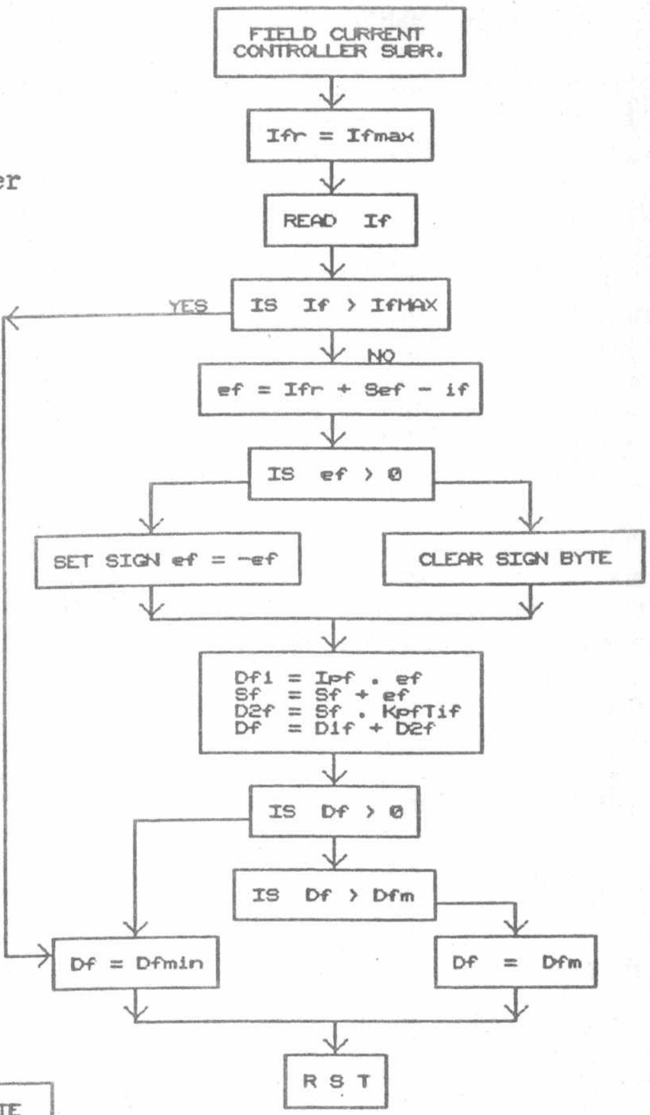


Fig.10 Field Current Controller

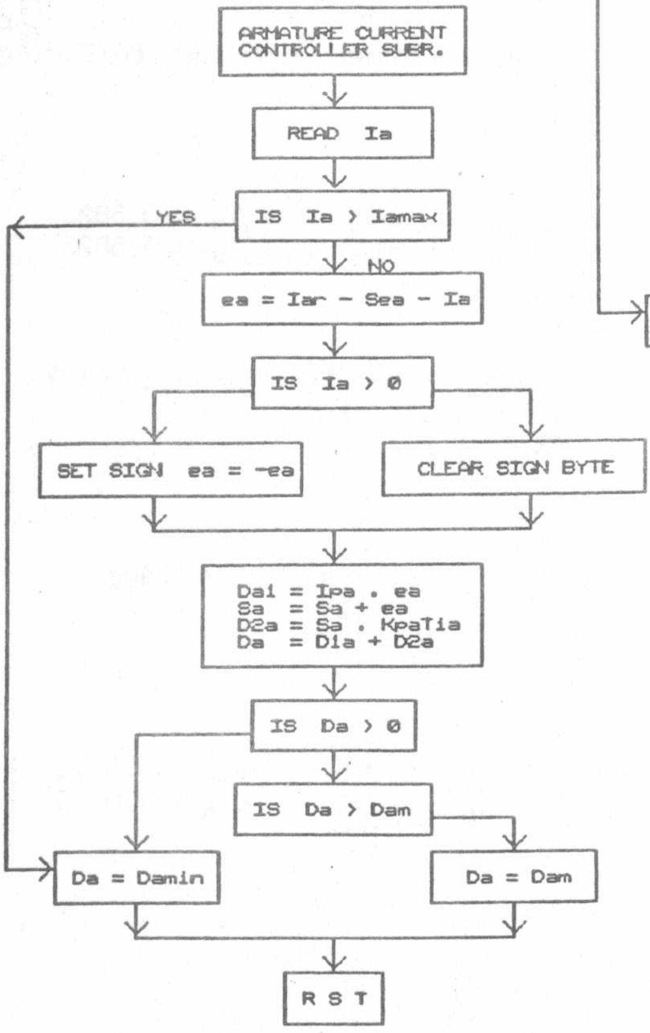


Fig.11 Armature Current Controller

IV. EXPERIMENTAL RESULTS

The aim of this section is to test the whole system. The steady-state and the transient responses will be evaluated under different modes of operations. The machine used is a 20 A, 18 V dc separately excited dc motor.

The speed responses of the control system are evaluated experimentally by a step speed input $W_r = F0_{16}$ rad/s = 2292 rpm. These responses are shown in Figs.12 to 16. The actual motor speed and armature current are displayed using dual channel storage oscilloscope at various values of K_{pn}, $K_{in}T_n$, K_{pa} and $K_{ia}T_a$. In general the steady state speed error is very small (its numerical value can be read at memory location 000A), and the speed is stable although the overshoot increases at high values of $K_{in}T_n$ and K_{pn}. The inner current loop seems to be unstable for values of $K_{in}T_n > .8_{16}$ and $K_{pn} > 2$ although speed itself is stable.

The oscillograms in Fig.12.b and Fig.13.b illustrates the steady-state speed and the corresponding armature current. The armature current fluctuates with the sampling frequency of the speed loop. This is expected since reference armature current is updated at this frequency.

The responses recorded in the oscillograms are performed with the following constants for the different controllers.

Response of the speed and armature current:

Fig.12.a,b	$K_{pn} = 2.0$	$K_{in}T_n = 0.3125$	$K_{pa} = 0.3125$	$K_{ia}T_a = 0.5625$
Fig.13.a,b	$K_{pn} = 3.0$	$K_{in}T_n = 0.5000$	$K_{pa} = 0.3125$	$K_{ia}T_a = 0.5625$

Response of the shunt field winding:

Fig.14.a	$K_{pf} = 0.65519$	$K_{if}T_f = 0.40625$	rated field voltage (12V)
Fig.14.b	$K_{pf} = 0.65519$	$K_{if}T_f = 0.40625$	$E = 24$ V
Fig.14.c	$K_{pf} = 0.65519$	$K_{if}T_f = 0.4375$	$E = 24$ V

Response of the series field winding:

Fig.15.a	$K_{pf} = 0.625$	$K_{if}T_f = 0.40625$	rated series voltage.
Fig.15.b	$K_{pf} = 0.625$	$K_{if}T_f = 0.40625$	$E = 12$V
Fig.15.a	$K_{pf} = 0.625$	$K_{if}T_f = 0.5$	$E = 12$V

Response of the speed:

Fig.16.a	$K_{pn} = 3.0$	$K_{in}T_n = 0.5$	$K_{pa} = 0.375$	$K_{ia}T_a = 0.75$
Fig.16.b	$K_{pn} = 1.0$	$K_{in}T_n = 2.0$	$K_{pa} = 0.375$	$K_{ia}T_a = 0.75$

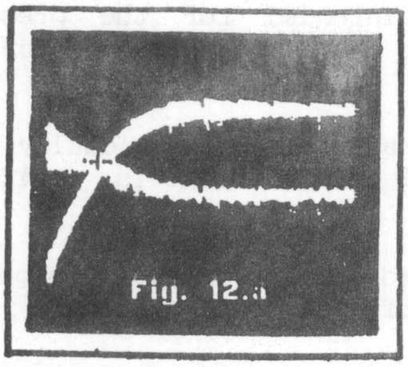


Fig. 12.a

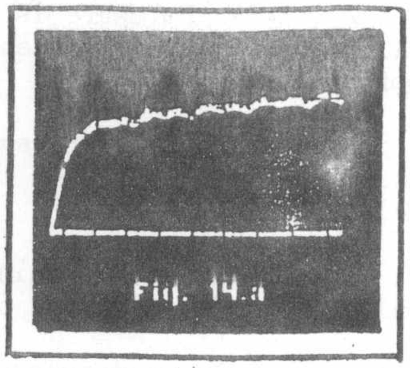


Fig. 14.a

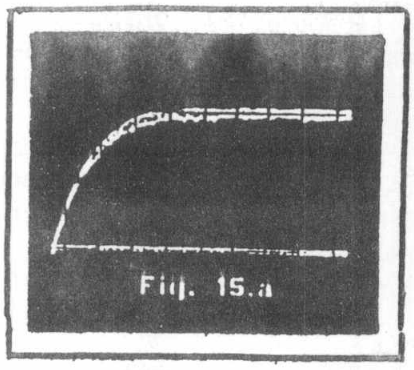


Fig. 15.a

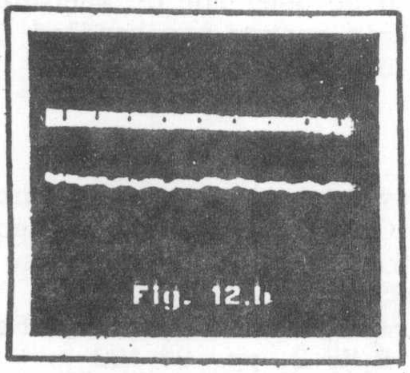


Fig. 12.b

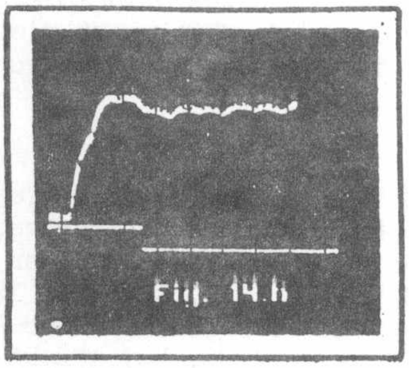


Fig. 14.b

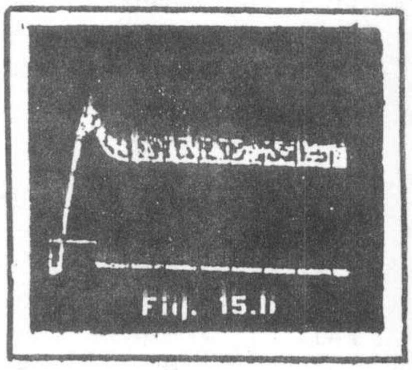


Fig. 15.b

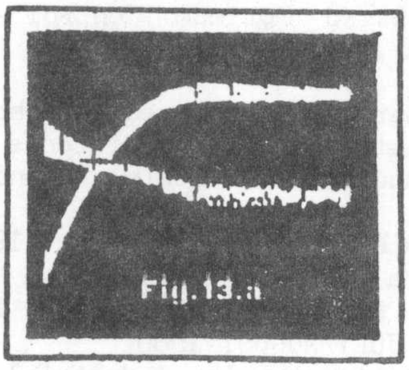


Fig.13.a

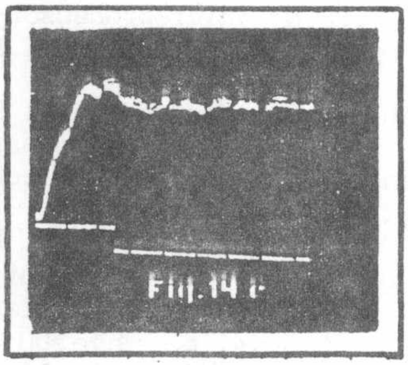


Fig.14 c

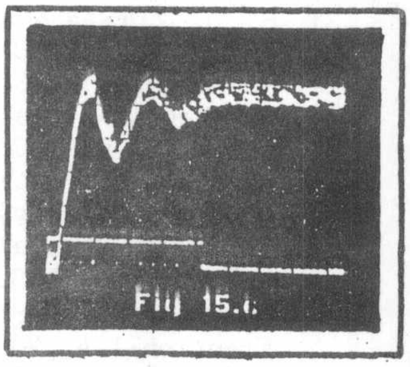


Fig 15.c

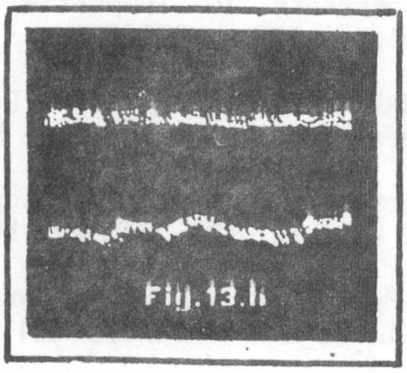


Fig.13.b

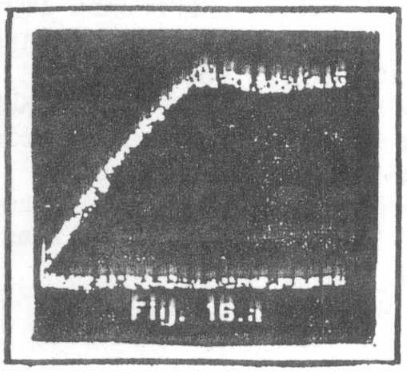


Fig. 16.a

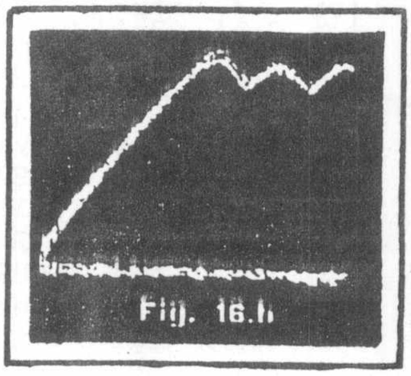


Fig. 16.b

CONCLUSION

The control system is completely realized using the 6800 MPU and some of its peripheral chips. Although this hardware is intended for the present application a good deal of generality is present making it suitable for a number of similar microprocessor controllers .

The control algorithms, programs and subroutines are tested individually and as a whole. Correct and satisfactory operation of this software is verified experimentally by overall system operation.

Satisfactory transient responses are obtained and further response improvements is practically possible simply by software modifications.

It is important to remember that the principle can be used for dc machine of high power rating where large quantities of electric energy are handled and the control system cost is only a small fraction of total cost.

REFERENCES

[1] T. EGAMI, J. WANG & T. TSUCHIYA: "Efficiency Optimized Speed Control System, Synthesis Method Based On Improved Optimal Regulator Theory-Application to Separately Excited DC Motor System" , IEEE Trans. on I.E , Vol. IE-32, No.4 p.p 372-380, November 1985

[2] R. J. HILL: "Chopper Control of DC Disc-Armature Motor Using Power MOSFETs" ,IEE PROC. March 1985, Vol.132,Pt. B,No.2, pp. 93-100 .

[3] B. S. DEWAN & A. MIRBOD: "Microprocessor-Based Optimum Control For Four-Quadrant Chopper", IEEE Trans. on I.A., Vol.IA-17, No. 1 pp. 34-40, Jan/Feb. 1981.

[4] R. Mostafa, Salah Gh. Ramadan and Gamal Sarhan: "Choppered DC Motor Drive using MOS and Bipolar Power Transistors"; 2nd Conference on Aeronautical Sciences and Aviation Technology, Military Technical College Cairo-Egypt 1987.

[5] B.C.KUO: "Digital Control Systems", HOLT, RINEHART & WINSTON Inc. 1980

[6] W.G.DUNFORD & S. B. DEWAN: "The Design of a Control Circuit of a Two Quadrant Chopper Based on the Motorola 6800 Microprocessor" IEEE Trans.on I.A.,VOL.IA-16,NO.4, pp. 495-500 ,JULY/AUGUST 1980

[7] B.J.CARDWELL, C.J.GOODMAN: "Response Improvements in Industrial DC Drives Driven From Optimal Analysis" IEE PROCEED, May 1984,Vol.131,Pt.B,N , p.p 91-99.

[8] R.R.SUL, B.J.VASANTH, T.KRISHNAN & M.KUMAR: "Microprocessor- Based Speed Control System for High-Accuracy Drives" IEEE Trnas. on I.E, VOL. IE-32 NO.3, pp.209-214, AUGUST 1985.

[9] J.B.PLANT, S.J.JORNA & Y.T.CHAN: "Microprocessor Control of Position or Speed of an SCR DC Motor Drive"IEEE Trnas. on I.E, VOL. IECI-27 NO.3, pp.228-234, AUGUST 1980

[10] M.R.STOJIC: "Design of the Microprocessor-Based Digital System for DC Motor Speed Control"IEEE Trnas. on I.E, VOL. IE-31 NO.3, pp.244-274, AUGUST 1985.

[11] A.K.LIN & W.W.KOESEL: "A Microprocessor Speed Control System" IEEE Trnas. on IECI, VOL. IECI-24 NO.3, pp.241-249, AUGUST 1977.

[12] Heath Company: "Individual Learning Program; MICROPROCESSORS" EE-3401 and ""Individual Learning Program: MICROPROCESSOR APPLICATIONS" ee-3405 Copyright (C) 1977, Heath Company U.S.A.