01 Jan 1987

Computer-Aided Testing Of Electrical Machines: Software Development

Max Darwin Anderson
Missouri University of Science and Technology, mda@mst.edu

Steven A. Hauk

Robert Laramore

Hardy J. Pottinger
Missouri University of Science and Technology, hjp@mst.edu

Recommended Citation

M. D. Anderson et al., "Computer-Aided Testing Of Electrical Machines: Software Development," *IEEE Transactions on Power Systems*, vol. 2, no. 3, pp. 824 - 830, Institute of Electrical and Electronics Engineers, Jan 1987.
The definitive version is available at https://doi.org/10.1109/TPWRS.1987.4335215

Computer-Aided Testing of Electrical Machines: Software Development

Max D. Anderson, Senior Member
Steven A. Hauk
Robert Laramore, Member
Hardy J. Pottinger, Associate Member
University of Missouri-Rolla

Abstract

Computer-aided test stations have been developed for research, experimentation, and testing of electric motors and generators. Data acquisition and control of the machines and drives are provided through a computer and its interfaces. Machines are connected to the computer through transducers for data acquisition, and power electronic drives for machine control.

General software was written to enable the monitoring of all transducers continuously on the CRT screen while giving the user manual control of the machines and drives. Specialized software was written for automatic control of the induction, dc, and synchronous machines for specific experiments on each type of machine.

1. Introduction and Overview

The Emerson Electric Company Machines and Drives Laboratory is designed for computer controlled research and testing of machines and drives. It is used by the faculty and graduate electrical engineering students for research and by the undergraduate students for power engineering education. Computer-aided testing is accomplished by near real-time computer control of the machines and simultaneous data acquisition. Computer control is versatile, precise, and rapid, thus offering many capabilities. Data acquisition can be done at a swift sampling rate to allow for many test data points to be gathered on the state of the motors. The data is easily processed and displayed for quick feedback on a test run.

The laboratory consists of 6 identical independent stations. Each has its own computer, machines, machine drives, and sensing devices. The computer controls the machines through their respective drives and collects data through measuring devices. A dynamometer consisting of a dc motor with a regenerative drive is used as the load or source of mechanical torque on the test machine. Computer controls are implemented for both the test motor and the dynamometer.

2. Hardware System

The hardware system consists of Emerson ac and dc electronic drives; a Motor Control Center (MCC); induction, dc, and synchronous machines; a dc dynamometer; DEC LSI 11/2 microcomputers or IBM PC's; sensing devices; and a few custom interface circuits.

The physical system's block diagram is shown in Figure 1. This figure shows the computers connected to the drives, custom circuits, and the sensing devices (transducers and signal conditioners). The drives, circuits, and sensing devices are connected to the dynamometer and the machines. Information flow is bidirectional between the machines and the computer. For control, the computer sends signals to the drives and the MCC. For data acquisition, information is sent to the computer from sensing devices and the machines.

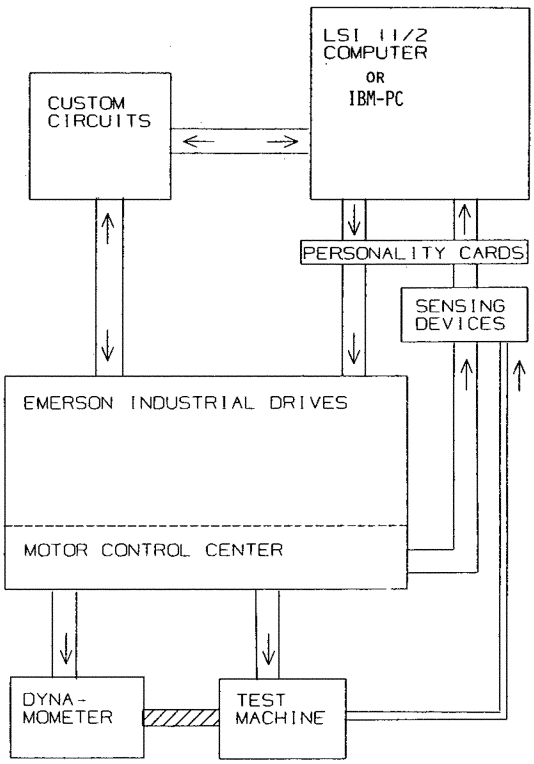

Figure 1. Hardware System Diagram

2.1. Component Descriptions

The test motors presently include a dc machine, an induction machine, and a synchronous machine. A 20 horsepower rated dc motor with regenerative drive is used as a dynamometer for precision load. The dc, induction, and synchronous machines are each 5 horsepower motors used as test machines. The system also allows for testing of other machines, provided that the physical mounting requirements are met. The shaft of the test motor must be connected to the shaft of the dynamometer. [3]

Emerson Industrial Drives [5] are used to control the machines by generating a field, controlling speed, limiting current, detecting faults, and turning switches on and off. There are three drives; a dynamometer drive, an induction motor/synchronous motor drive, and a dc motor drive. The dynamometer drive is a variable speed regenerative dc drive dedicated to the stationary dynamometer. The other two drives are used for their appropriate motors. They all operate in one of two modes, manual or computer. In the manual mode the controls are activated by switches and knobs on the drive's front panels. In the computer mode the controls are activated by computer generated signals.

87 WM 006-0 A paper recommended and approved by the IEEE Power Engineering Education Committee of the IEEE Power Engineering Society for presentation at the IEEE/PES 1987 Winter Meeting, New Orleans, Louisiana, February 1 - 6, 1987. Manuscript submitted August 26, 1986; made available for printing December 10, 1986.

0885-8950/87/0800-0824$01.00©1987 IEEE

Closely linked with the drives is the Motor Control Center. The MCC is a panel that gives access to utility power sources, a dc power source, and the signals generated by the drives. Users connect the test motor to the MCC.

The computers are DEC LSI 11/2's or IBM PC's enhanced with serial, parallel, analog input, and analog output interfaces. Analog input lines, analog output lines, digital input lines, and digital output lines are used to control and receive data from the system. These lines are accessed using machine language subroutines that address the respective interfaces. The DEC computer uses the operating system RT-11 [2] and the IBM PC use DOS 2.1.

The custom interface circuits are a tachometer counter, a multiplexer board, a relay board, and a frequency counter. The tachometer counter inputs a signal from the dynamometer's 1500 ppr generator and outputs a 12 bit binary number that gives the speed of the shaft. The multiplexer board multiplexes 64 (4 channels of 16) digital input lines into 16 lines for the digital input interface to read. For example, the output of the tachometer is connected to one of the input channels of the multiplexer boards. The relay board is used for on/off switching control. It receives a 16 bit word on the digital output lines and uses each bit to control one relay. These relays are then used as switches in the control logic of the drives.

Sensing devices include electrical transducers, torque tables, inline torque sensors, thermocouples, and signal conditioners. Sensing devices measure such values as current, voltage, power, torque, and temperature. The devices convert the measurement on a given ratio to a 10 volt dc signal which is routed to the computer.

2.2. Component Interconnections

The computer is connected to the system by 16 analog input channels, 6 analog output channels, 16 digital input lines, 16 digital output lines, and 2 control output lines. All of the analog lines connect to personality sockets where they are connected to transducers and drives. The 16 analog input channels are input to a single A/D interface at the computer end and connect to personality socket B at the other end. The 6 analog output channels come from 2 interfaces. Two of these channels come from the same interface as the 16 analog inputs and the other four come from a separate D/A interface. The 6 channels connect at one end of personality socket A where they are routed to the drives.

The remaining computer connections come from a single parallel I/O interface (digital I/O) board. The 16 digital input lines are connected to the output of the multiplexer board and the 16 digital output lines are connected to the relay board. The two control lines also connect to the multiplexer board where they are used to select the channel that is connected to the 16 digital input lines. The hardware is described in greater detail in references [1] and [3].

3. Software Development

The overall requirements for the software were to monitor and control the system. An interactive monitor mode for manual operations and an automatic mode were developed. The monitor mode is implemented by a program that constantly reads all sensors in the system and displays the values on the screen, updating the values at least every second. This mode also allows, by keyboard entry, control of the relays and computer analog outputs. The values of the outputs are also displayed on the screen.

In the automatic mode, computer programs were written for specific machines to perform specific tests automatically. The programs provide for entry of test parameters at the beginning of the program to allow for variations in the test. After the test parameters have been entered, the test is run under computer control with no operator intervention. Test results (data) are stored in a file for subsequent printout as a tabular listing and/or plotting by a graphics facility.

The software is built around two programs. The first is the Interactive Monitor Program. It is comprised of one main block of code and a few dedicated subroutines. The second is the RAMP subroutine. It is a versatile subroutine used by several programs as the execution portion of the program. It

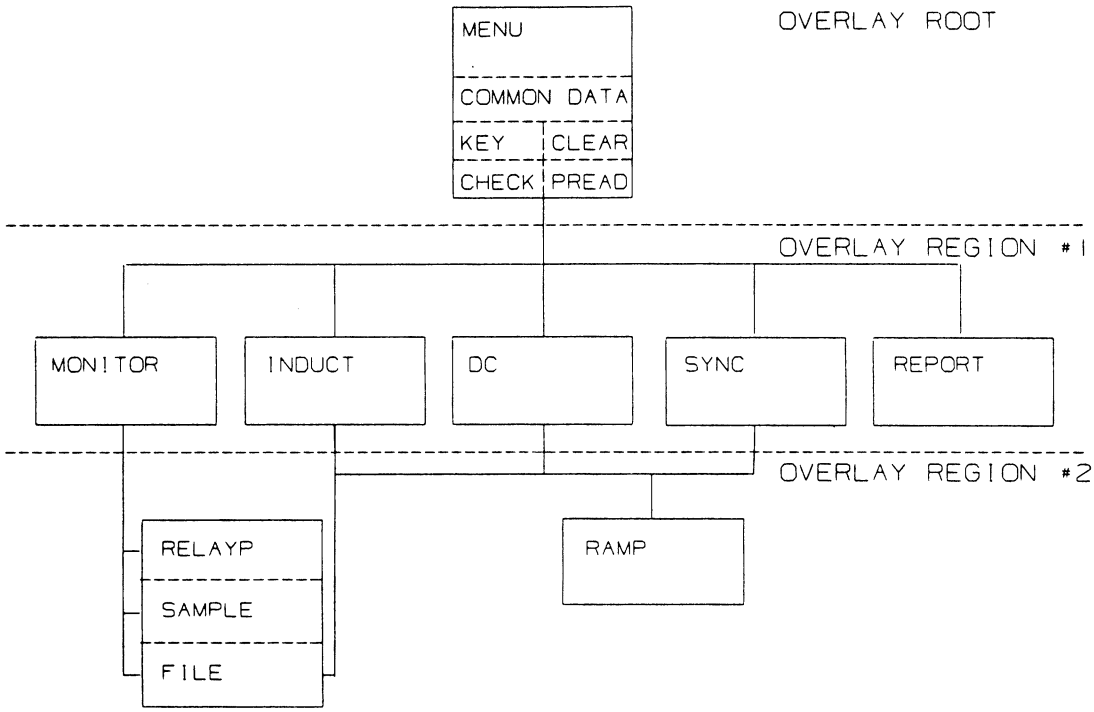

Figure 2. Software Overlay Structure for DEC Computer.

will ramp any one of the output channels from a given initial value to a given final value in a given time.

The body of software is subdivided into programs and subprograms that overlay each other on three levels. The structure of the software system is portrayed in Figure 2. The reason for the overlay instead of individual independent programs is to allow for switching between programs without changing the state of the system and to allow programs to keep track of the state. For example, relays and output channels might be set in the Monitor program and this state should stay undisturbed while switching to another program. Also the overlay structure is used because of memory constraints and because data is shared by the programs.

The highest level in the system is the root program, which is memory resident and has no overlay. The root performs several functions. First, it contains the menu. Second, it performs initialization functions such as setting the system to a known state with all relays set to default values and all output channels set to zero. Then, it obtains the laboratory station number and performs certain system configurations for differences in the stations. Lastly, it determines which personality cards are in the personality sockets and obtains data from disk files that define the cards. This data is used in the software system for labeling, scaling, etc.

Also located in the nonoverlay area along with the root are the COMMON areas and several widely used subroutines. Five COMMON areas are located here. There is one for screen editing functions, one for data storage, one for system status, and one for data for each personality card. The subroutines are for key entry management and for personality card information handling.

The second highest level in the system contains the main programs. These programs are all overlayed since no two are required in memory at the same time and there is no time critical switching between them. These main programs are Interactive Monitor, Induction Machine, Dc Machine, Synchronous Machine, and Data Listing. Induction Machine, Dc Machine, and Synchronous Machine are largely noninteractive programs specially designed to handle each respective motor. Data Listing is a program for creating data reports on the screen or the printer.

The third highest (lowest) level is an overlay region for subroutines. Most of these are used to support the Monitor program. One subroutine, RAMP, is the foundation of the Induction, Dc, and Synchronous machine programs.

Hardware is accessed by transferring data to and from registers on the hardware interfaces. Three hardware interfaces are in use; a digital I/O board, an analog input board, and an analog output board. The registers on each of these interface boards have unique addresses. Using DECUS assembly language subroutines on the DEC LSI 11/2 and LAB PAC on the IBM PC, communication between the software and the interfaces is made simple. Upon calling one of these subroutines the data is sent to or retrieved from a register on one of the interfaces. The Software/-Hardware interfaces are described in detail in reference [1] pg. 18. Programs for the DEC are written in Fortran. Programs for the IBM PC are written in Basic.

3.1. Menu Program

The Menu program is the root program that calls all other programs. It displays the menu on the screen and performs variable initialization. Information displayed on the Menu to be selected by the user are shown in Table I. All analog outputs and the registers that hold the values are initialized to zero. The relays and the variable that keeps track of the relay values are all set to default values. The last function of the Menu program is to display the menu and call any program the user selects.

TABLE I

1. Monitor (Interactive Mode)
   a. Manual (Emerson Industrial Drives Panels)
   b. Computer (Keyboard Control)
2. DC Machine
3. Induction Machine (Automatic Mode)
4. Synchronous Machine
5. Data Listing. (Recorded during experiment.)

3.2. Interactive Monitor Program

The Monitor program is a major component of the software system. It is the interactive program that monitors all analog and digital inputs, and allows keyboard control of all digital and analog outputs. It displays on the screen everything it is monitoring and controlling, and prompts the user for keyboard inputs.

This program monitors all transducers, the tachometer, the frequency counter, the thermocouple, the torque table, and the inline torque sensor. It also controls all of the relays and all of the analog output channels. It also has the capability to store a snapshot of everything that is being monitored, at the press of a key. This can be done up to 100 times with present memory limitations. Other capabilities include being able to zero any channel at any time and displaying any one channel at double size for viewing at a distance.

There are two states in which this program will run, the manual mode and the computer mode. Figure 3 shows the screen of the Monitor program in both its modes. These two modes correspond to the two modes of the drives. The drives, when in the manual mode, are controlled by the knobs and buttons on their front panels only. In the computer mode, control is achieved through computer analog and digital outputs, and the front panel controls are made inoperative. Each drive has a switch on the front panel that selects the operating state. In the manual mode, the program only monitors the analog and digital inputs and makes no attempt to control the relays or the analog outputs. In the computer mode, the program performs the monitoring function and allows for the setting of relays and outputs from the computer.

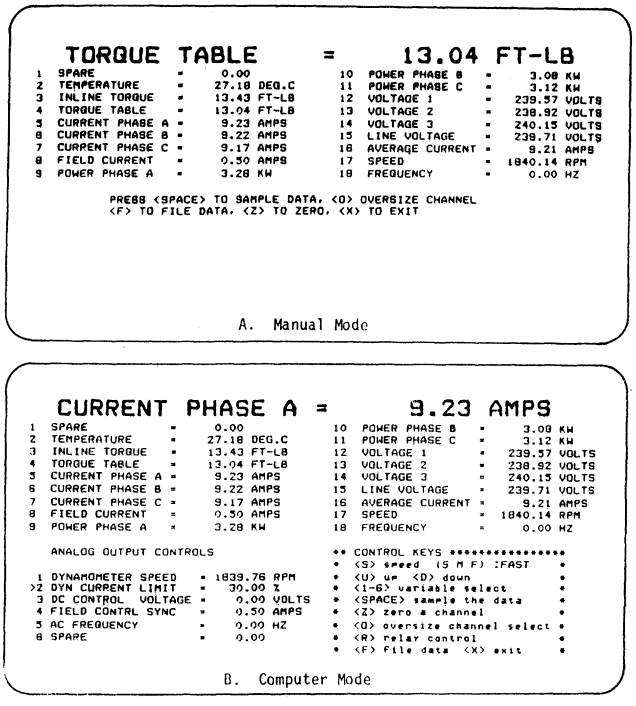

Figure 3. Monitor Screens

The program works by repeatedly sampling and updating the analog and digital inputs until a key is pressed that causes alternative action. When one of the number of keys are pressed the program stops metering for a moment and executes a subroutine or statement that performs the desired function. The keys and their functions are shown in Table II.

TABLE II
MONITOR INTERACTIVE FUNCTIONS FOR THE COMPUTER MODE

KEY	FUNCTION ENVOKED
1-6	- Choose selected output channel
U	- Increase value of selected output channel
D	- Decrease value of selected output channel
S	- Select increment/decrement value for U and D
Z	- Zero a desired channel
O	- Select channel for large size display
R	- Change relay settings
F	- File data
Space	- Store one sample of data
X	- Exit the program

Note that the manual mode uses a subset of these same keys. Most of the keys are self explanatory, but those dealing with the selected output channel require additional description. There are 6 physical analog outputs, but only one may be selected at any given time for modification by pressing its corresponding number. As shown in Figure 3B, a small arrow on the screen points to that channel to verify that it is the selected output channel. Its value can be changed at a slow, medium, or fast rate. This rate is chosen by the S key and the selected increment is displayed on the screen. Pressing this key permutes the rate between slow, medium, and fast. These three rates are a factor of 10 apart. For example, the dynamometer speed can be changed by increments of 1, 10, or 100 r/min. pressing the U key the value of the selected output channel will increment by either the slow, medium, or fast increment value. The D key will cause a decrement in the same manner. Once the U or D key is pressed the value will change on the output channel and on the screen.

3.3. RAMP Subroutine

One of the most common processes associated with testing a motor is the facility to ramp a variable continuously from an initial value to a final value. RAMP is a program to approximate this by a linear series of discrete values. This program will ramp any of the analog output channels. Because this is a common process, it was made into a subroutine for the use of several programs. The Induction, the Dc, and the Synchronous machines programs all use the RAMP subroutine for ramping analog output channel values. the flow chart for RAMP is shown in Figure 4.

The size of the discrete steps in terms of the variable being ramped is determined by the time given for the ramp, the initial value, and the final value. These three values determine the slope, and the slope multiplied by Δt gives the size of each step, Δy. Δy is added to the output value every Δt, creating the discrete ramp. The time increment Δt and the variable increment Δy are made small in comparison to the test run time and the difference between initial and final y to minimize the error caused by the discrete ramp.

The first four variables that must be passed to the subroutine are the analog channel number, the initial value, the final value, and the time for the ramp. These values must be passed in a certain form. The analog channel number must be passed as an integer from 1-6. The ramp time must be passed in seconds. The initial and final values are passed in the encoded form, but are not quantitized yet. They are encoded on a continuous scale from 0 to 4095 and are not rounded. The rounding is left to the RAMP subroutine.

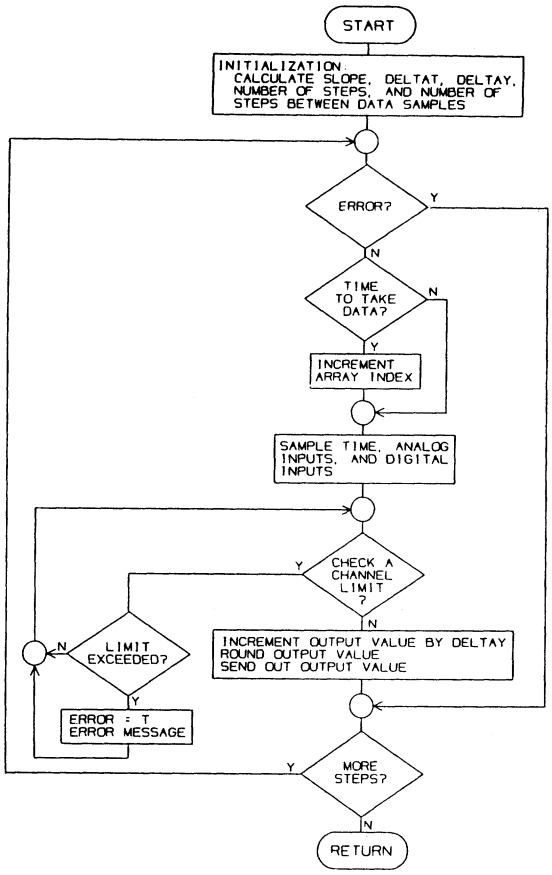

Figure 4. Flowchart for Ramp Subroutine

Another variable that must be passed is the frequency at which data is to be sampled. This is passed in data sets per second. A data set is one sample for each input channel. The maximum sampling rate is $1/\Delta t$; i.e., one sample every Δt seconds. Other allowable sampling intervals are integer multiples of Δt, because the data acquisition and the analog output channel control are in the same program loop. The value (in data sets per second) does not have to be passed in frequencies based on multiples of Δt. Any number can be entered, and the RAMP subroutine will round it up to the next allowable frequency.

The RAMP subroutine also has the capability to check the data in real time to make sure that none of the input channels reach user specified limits during the run. For example, the user could specify a maximum speed of 2000 r/min. If the value on a specified channel reaches the specified limit, then the RAMP subroutine terminates, prints a message telling what occurred, and returns control to the calling program.

Solutions to problems of loop timing and quantization error are discussed in Reference [1].

One additional requirement is for the calling program to output the initial value and starting conditions before calling RAMP. This is done for two reasons. One is to provide a stabilization period after starting the motor. The other enables the RAMP loop to begin taking data before it starts outputting analog control values.

3.4. Induction Machine Program

The induction machine program uses the RAMP subroutine to force the induction motor through any range of speeds from zero r/min. (blocked rotor) to max. r/min. (no-load) over any time period (normally 6 to 20 seconds). The basic flow of the program, as

shown in Figure 5, is test parameter entry, system set up, ramp preparation, the ramp, system reset, and data storage.

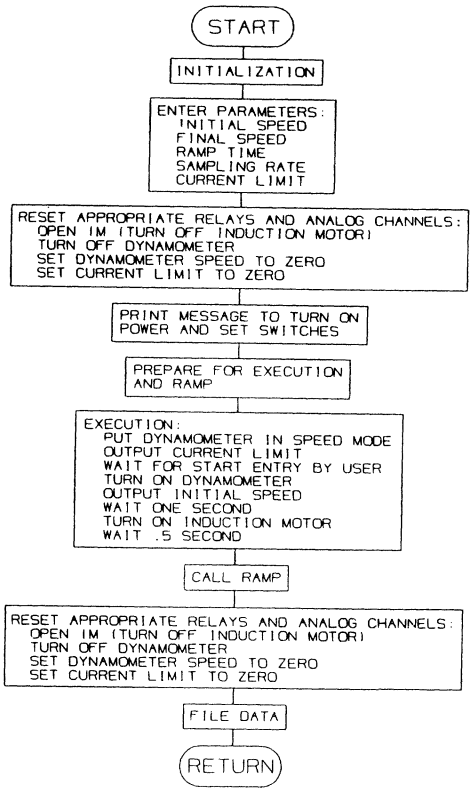

Figure 5. Flowchart for Induction Machine Program

The program uses three relays and two analog output channels. The three relays are used to turn on/off the dynamometer, put the dynamometer into the speed mode, and turn on/off the induction motor. The two analog channels are used to set the dynamometer current limit, and control the speed of the dynamometer. Upon exit from the program the relays will be left open and the analog output channels will be set to zero.

Parameter entry is the first section of the program. The parameters required are the initial speed, final speed, current limit, RAMP time, and data sampling rate. The initial and final speeds are converted to their encoded format and passed to the RAMP subroutine. The ramp time and the data sampling rate are also passed to the Ramp subroutine, but do not require conversion. The current limit is converted to its encoded format and sent to the dynamometer drive on analog channel 2.

The next section puts the system in a state ready for execution and prepares for calling RAMP. The first step in this section is to open the three pertinent relays and set appropriate analog channels to zero. This is to prepare for the next step which is for the user to set all switches and power breakers in a position ready for execution. The reason for this step is so that nothing unexpected will happen when the power is turned on. Following these two steps, the initial speed, the final speed, and the current limit are converted to the encoded format. The dynamometer is put in the speed mode by closing relay 1 and the speed is reset to zero once again in case switching on the power caused some sort of failure. After this, the dynamometer's current limit is set, making everything ready for execution. Upon command of the user (an S keystroke) the execution stage begins. The dynamometer is turned on by closing relay 2 and the initial speed is sent out on analog channel 1. The dynamometer responds by accelerating to the initial speed. A one second period is inserted to allow for machine acceleration and stabilization. After this, relay 10 is closed to start the induction motor and a .5 second stabilization period is inserted.

After the stabilization period the machines are running and ready to be ramped. Execution is turned over to the RAMP subroutine which takes the machines through the test. After returning, the machines are turned off by opening relays 2 and 10. The data acquired during the ramp is filed and the experiment is completed.

3.5. Dc Machine Program

The dc machines program is similar to the induction program. It has the same basic flow, but is designed to take a dc machine through a range of torques. The flowchart is shown in Figure 6.

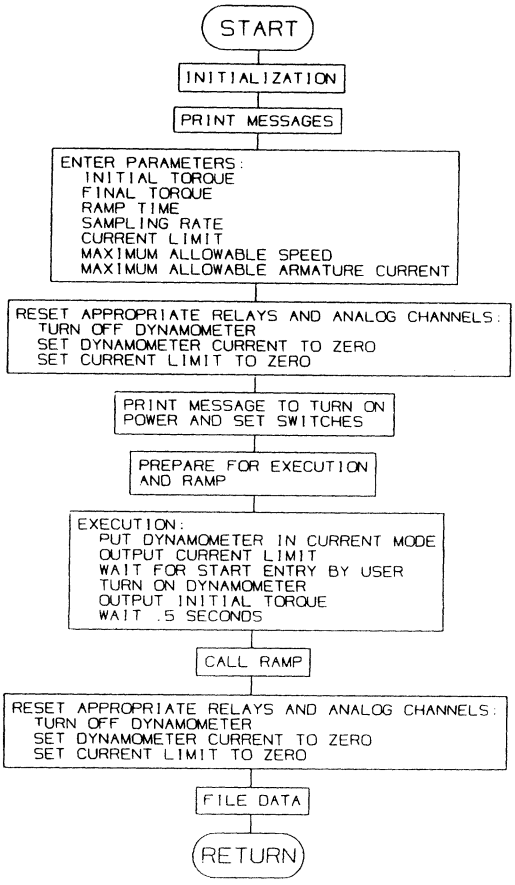

Figure 6. Flowchart for Dc Machine Program

The program uses two relays, relay 2 to turn on/off the dynamometer and relay 1 to put the dynamometer in the current mode. Two analog output channels are used; channel 2 for the dynamometer current limit and channel 1 for the dynamometer current control. By ramping the dynamometer current the dc motor is ramped through a series of torques, because torque is roughly proportional to armature current for the dc dynamometer (field excitation is constant)

The user enters the initial torque, final torque, ramp time, data sampling rate, current limit, maximum allowable speed, and maximum allowable armature current. All values, except the current limit, are passed to the RAMP subroutine. The initial torque and the final torque are converted to values that correspond to the dynamometer's current mode and then are converted to their encoded formats.

The ramp time and the data sampling rate are sent as entered. The other two values, the maximum allowable speed and the maximum allowable armature current, are used in conjunction with the RAMP subroutine's ability to check specified input channels for maximum values. The current limit is converted to the encoded format and sent to the dynamometer drive prior to calling RAMP.

The execution begins by opening relays and zeroing analog channels. Following that, the array is set up that will pass the channels and limits to be checked. Then starting and ending ramp values are converted, the dynamometer is put in current mode, and the current limit is sent to the dynamometer drive. The dynamometer is turned on, the initial value is sent to the drive, and a stabilization period of .5 seconds is inserted to allow the motors to reach steady state. Execution is then turned over to the RAMP subroutine. Upon return from the RAMP subroutine, the dynamometer is turned off and the channels are reset to zero. The data is then filed and the program is completed.

3.6. Synchronous Machine Program

The synchronous program is similar to the induction and dc programs. It also has the same basic flow, but is designed to take a synchronous machine through a range of field currents. The flowchart is shown in Figure 7. The program uses relay 7 and analog output channel 4. The relay is used to turn the field on and off, and the analog channel is used to ramp the field.

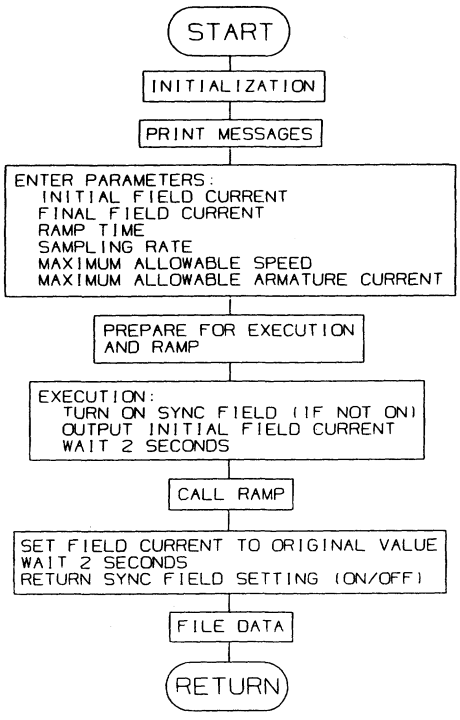

Figure 7. Flowchart for Synchronous Machine Program

The parameters entered by the user are the initial field current, final field current, ramp time, data sampling rate, maximum allowable speed, and maximum allowable armature current. All values are passed to RAMP.

Execution begins by setting up the array that passes channels and channel limits that will be checked. Then the initial and final field currents are converted to their encoded values. Just before RAMP is called, the field is set to the initial value and a stabilization period of 2 seconds is inserted. After the ramp, the field current is set to the initial value and the field relay remains on. The data is then filed and the program is completed.

4. Testing and Evaluation

Testing and evaluation of the system has revealed that most of the errors can be attributed to timing and data skew. The magnitude of the data is as correct as the accuracy of the transducers and measurement devices, but the software system allows and creates some timing problems. First the transducers and signal conditioners all have a significant step response time that does unavoidable filtering. Some of the time constants for these transducers are as long as 200ms. Two of the transducer step responses are shown in Figures 26 and 27 of Reference [1]. Without correction for this filtering the experiments must be limited to quasi-steady-state testing. This is acceptable for undergraduate laboratory experiments, where motor test runs are 10 seconds or more, but might be unsatisfactory for a more involved research effort, where transient response is of interest. [4]

The second type of timing problems are those introduced by sampling channels sequentially. This creates a data skew situation where samples from two channels that are to be compared (i.e., torque plotted against speed) do not correspond to the exact same time slice. They are shifted by a constant time factor. Adjacent channels are sampled at 600μs intervals as the programs are now written. In the RAMP subroutine loop, 16 channels are sampled consecutively. This means that the 1st and 16th channel are sampled 9ms apart. In fact, without going into detail, the remaining part of the loop takes 9ms also, which sets up the situation where 1st channel may be sampled every 18ms and the 16th channel is sampled in the middle of that interval. This may be undesirable for some experiments.

The third timing problem is the response time of the machines and drives. The control signals are calculated precisely and sent out of the computer with known accuracy and timing. The drives and machines receive these signals and respond with reasonable accuracy. Because they are mechanical devices, the response is not immediate (i.e., there is a delay due to the response time of the equipment). Even though this may or may not be considered as a problem to be solved by the software, it is a problem that the present software can be used to study. That is, the software may be used to determine the response time of the drives or the mechanical response time of machines to external control signals.

Although the problems of timing and data skew exist, the solution for this application was to make the motor test run times long enough to cause errors in the test data to be small (negligible). For example, in tests of 5hp machines (induction, dc, and synchronous) test run times of 6 seconds or more produce test data of negligible error as seen from motor characteristic curves (speed vs. torque, armature vs. field current) plotted from these data. The upper limit on test run time is a function of how much the test motor overheats.

Conclusion

The computer-aided motor test system has given the user the capability to do research and experimentation on machines with advantages of quick feedback, precise control, and intricate data acquisition. With the Monitor program the user can manually control modes and inputs of the system and have real-time feedback of the measurement data. With the Induction, Dc, and Synchronous machines programs, he/she may set up or change the parameters of an experiment easily to obtain information about the machine.

While the software gives the user almost unlimited capabilities, the problems of data skew and transducer filtering remain. At present, the user must correct for these problems himself. Further

problems in software by digital signal processing and interpolation.

Acknowledgements

The authors are very grateful to the following companies for their generous contributions: Emerson Electric Company for the bulk of the funding for this project, Digital Equipment Corporation for computer and peripherals, Furnas for the switchgear, and others including many students who received hands-on" experience as part of their education.

References

1. Hauk, Steven A., "Software Development for the Emerson Computer Controlled Machines Laboratory", MS Thesis, University of Missouri-Rolla, August 1985.
2. Digital Equipment Corporation RT-11 V4.0 Reference Manuals, Massachusetts: Digital Equipment Corporation, 1980.
3. Laramore, Robert and McPherson, George, "Computer-Aided Testing of Electric Motors for Education Purposes", IEEE IAS Conference Record Toronto, October, 1985.
4. Oppenheim, Alan V., and Schafer, Ronald W., Digital Signal Processing, New Jersey: Prentice-Hall, Inc., 1975.
5. System Manual and Drawing Set for Emerson Drives and Machines, Emerson Electric Company, Industrial Controls Division, Grand Island, NY, 1983.