Applying fiberoptic data links to instrumentation
by Michael Steven Beer

A thesis sumitted in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE in Electrical Engineering
Montana State University

Abstract:
The subject of this thesis is the application of a fiber optic data link for use in instrumentation applications where conventional information transmission techniques cannot be used due to excessive electrical noise. A technique was developed for converting a direct current voltage to a pulse string whose frequency was proportional to the voltage and transmitting this pulse string over the data link.

This technique also allowed recovery of the information at the receiving end in digital form using a microprocessor-based data acquisition system. As a demonstration of the validity of this technique, sensors were constructed to enable use of the data link to measure alternating currents, direct currents, and temperature. Mathematical transfer functions were developed for each of these sensors to aid the microprocessor-based system in providing accurate measurements.

STATEMENT OF PERMISSION TO COPY

In presenting this thesis in partial fulfillment of the requirements for an advanced degree at Montana State University, I agree that the Library shall make it freely available for inspection. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by my major professor, or, in his absence, by the Director of Libraries. It is understood that any copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Signature Michael Beer

Date July 11, 1980

APPLYING FIBEROPTIC DATA LINKS TO INSTRUMENTATION

by

MICHAEL STEVEN BEER

A thesis sumitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Electrical Engineering

Approved:

D. A. Pierre
Head, Major Department

Donald K. Weaver Jr.
Chairman, Examining Committee

Michael Malone
Graduate Dean

MONTANA STATE UNIVERSITY
Bozeman, Montana

July, 1980

ACKNOWLEDGMENT

The development and application of the fiberoptic data link draws on information learned through several other data acquisition projects developed at Montana State University. The author extends appreciation and thanks to Dr. Daniel N. March, who contributed the fiberoptics used to build the data link, and Western Telecomputing Corporation, that has provided assistance with hardware and its support. Of course, the author is especially grateful to Dr. Donald K. Weaver for his assistance and support of the project.

Michael S. Beer

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The subject of this thesis is the application of a fiber optic data link for use in instrumentation applications where conventional information transmission techniques cannot be used due to excessive electrical noise. A technique was developed for converting a direct current voltage to a pulse string whose frequency was proportional to the voltage and transmitting this pulse string over the data link. This technique also allowed recovery of the information at the receiving end in digital form using a microprocessor-based data acquisition system. As a demonstration of the validity of this technique, sensors were constructed to enable use of the data link to measure alternating currents, direct currents, and temperature. Mathematical transfer functions were developed for each of these sensors to aid the microprocessor-based system in providing accurate measurements.

CHAPTER I

INTRODUCTION

This thesis describes a research activity directed towards applying fiberoptic data links to instrumentation. The intent of this thesis is to describe the equipment necessary to achieve this, and verify that the instrumentation/data link combination operates properly. There will be no attempt to compare this method with other instrumentation techniques, and no efforts will be made to optimize the results presented here with respect to a certain set of conditions. The techniques advanced in this thesis have application in areas where high voltages and electromagnetic interference make electrical methods of information transmission impossible or impractical.

Fiberoptic data links have been much publicized by Bell Laboratories because of the large quantity of information that they can carry. Fiberoptics have useful properties other than large bandwidth. The properties of fiberoptics that this research exploits involve electrical noise immunity and electrical isolation. Another benefit that the methods developed have is that they digitally encode the information that is to be transmitted. As a consequence, these methods work particularly well in, but are not limited to, systems that employ microprocessors and other digital equipment.

Fiberoptics involves the transmission of signals using visible and near-visible light in transparent waveguides. Due to the short wave-length of light relative to the other forms of electromagnetic

radiation, these waveguides are much smaller than conventional waveguides. Typical fiberoptic fibers have cross-sections approaching that of a human hair. Because of this, the cables encasing the fibers are small diameter (1/4") and very flexible. The fiberoptics fibers themselves are constructed of either glasses or plastics, with glasses being preferred over plastics due to their lower attenuation. The fibers themselves are usually encased in several layers of plastic to form the fiberoptic cable. Since both glass and plastic are excellent insulators, the resulting fiberoptic cable provides excellent electrical isolation between transmitter and receiver. Since light waves are used rather than electric waves, the signals in the fiberoptics are immune to those type of interference that degrade electrical signals.

Information can be sent through fiberoptics in either analog or digital form. It was decided to use digital encoding in this project. The form of digital encoding chosen was pulse modulation, in which the frequency of the pulse string coming out of the analog-to-digital (A-to-D) converter is nearly a linear function of the input voltage. This type of A-to-D conversion will be referred to as voltage-to frequency conversion and the A-to-D converter will be referred to as a voltage-to-frequency converter throughout the rest of this thesis.

The instrumentation technique advanced in this thesis involves converting the quantity to be measured to a voltage. This voltage is then applied to a voltage-to-frequency converter and the pulse string

from this converter is used to turn a light emitting diode (LED) on and off. The light from this LED is then launched down the fiberoptic cable. At the receiving end of the cable, the light signal is transformed into an electrical signal that conforms to the voltage levels established for Transistor Transistor Logic (TTL) circuits. This signal's frequency is then proportional to the quantity initially measured, the exact proportionality coefficient(s) being dependent on the transfer function of the system as a whole. This type of system is diagrammed in Figure 1.

Two types of sensors were developed to interface with the fiberoptics data link. One type of sensor measures temperature and the other type measures electrical current. The temperature sensor is an integrated circuit that produces a current proportional to the temperature of the device. Additional circuitry changes this current into a voltage and allows amplification and offset to be set so that the temperature range of interest can correspond to the voltage range of the voltage-to-frequency converter.

The current sensors developed for the data link evolved along two lines of thought. One line was aimed at measuring current levels in conventional A.C. lines and the other line was aimed at measuring currents that could be either D.C. or A.C. The first sensor was developed to measure these A.C. and D.C. currents. It is based on a Hall effect device and generates an output voltage proportional to the in-

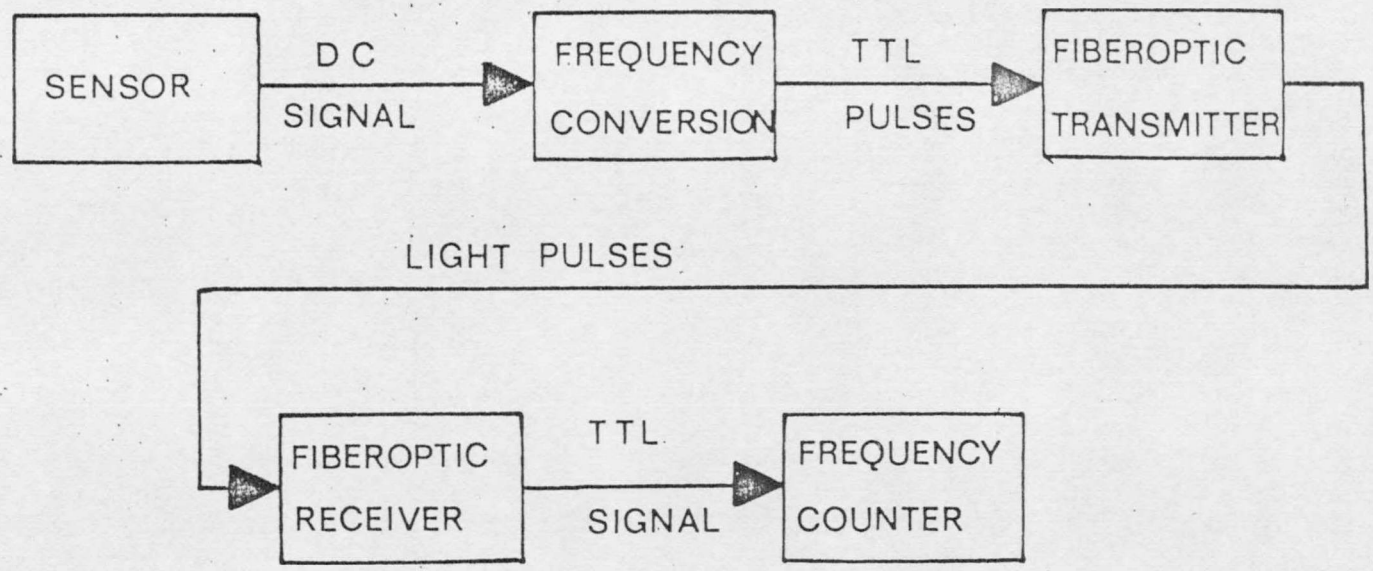


Figure 1 System Signal Flow

stantaneous intensity of the magnetic field surrounding the conductor and the excitation current supplied to the Hall effect device. The second current sensor is based around a conventional current transformer similar to those used in most power applications. This sensor is, of course, limited to A.C. currents with frequencies near 60 Hz. The current transformer outputs a current proportional to the A.C. current being measured. This A.C. current is passed through a resistor whose size was chosen so that the voltage drop generated corresponds to the voltage produced by the Hall effect device under similar conditions. The signal from either current sensor is processed by circuitry that rectifies and amplifies the signal producing an output that is compatible with the voltage-to-frequency converter.

No matter which sensor is used, the frequency of the voltage-to-frequency converter has to be determined. A microprocessor based data acquisition system (DAS) was used to determine the frequency of the pulse string generated by the fiberoptics data link receiver. The method used involved counting the number of pulses from the receiver over a known period of time. This method is generally referred to as a gated counter. The gated counter is controlled by a program that the DAS microprocessor executes. The program can direct the results to be printed on an output device or saved in memory.

This thesis describes the construction of the fiberoptic data link, the construction of the voltage-to-frequency converter, the con-

struction of temperature and current transducers compatible with this equipment, and it describes the software used to control and calibrate the system.

CHAPTER II

THE FIBEROPTIC DATA LINK

The fiberoptic data link consists of a transmitter and receiver separated by 100 meters of Valtec PC10 FIBERdata single-fiber fiberoptic cable. This data link is not a high speed data link, although the speed limitation is in the electronics and not the fiberoptics. It was decided to design the link to handle a square wave at a maximum frequency of 100 kHz. The link had to also be able to perform down to D.C. conditions. These specifications were arrived at after considering the frequency ranges available in hybrid and monolithic voltage-to-frequency converters. Since these specifications were set, several commercially available data links have come on the market with virtually identical specifications. This frequency range appears to be the industry standard for low data rate fiberoptic links.

The specification that the link be capable of operating down to D.C. completely precludes the use of capacitively coupled amplifiers at any point in the transmitter or receiver. It was also decided from a point of size, power consumption and operating environment that semiconductor sources and detectors would be used in the data link.

The light sources chosen were conventional LEDs. These sources are by far the easiest sources to bias, operate and align. Another factor in this selection was the large amount of work that has gone into improving LED reliability and operating life. LEDs are also much easier to obtain than other types of solid emitters. The main

disadvantage is that the devices require large amounts of power relative to the amount of power coupled into the optical fiber. This is a function of the broad area over which the light is emitted from the LED package and the small area that the terminated fiber optic has. This inefficiency is a trade-off that must be accepted to keep the ease of alignment attribute. The sources used in this data link nominally consume 100 milliwatts of power and the author has calculated that the amount coupled into the fiber is on the order of a few microwatts. Both red and infrared sources were used in the data link.

The detectors chosen were Positive Intrinsic Negative (PIN) diodes. These devices are readily available and operate at much lower bias voltages than avalanche type detectors. This is due to their internal construction which more effectively utilizes the voltages generated in the space-charge layer (11). The amount of current developed by one of these diodes in its reverse operating mode is linearly dependent on the incident light. This is true over the voltage range that extends from the reverse breakdown voltage of the diode to the point at which the generated current and the input resistance of the first amplifier produce enough voltage to remove the diode's reverse bias.

There was also the choice between single fiber light guides and multiple fiber light guides. The multiple fiber light guides are undoubtedly easier to use in a data link since they carry more light than single fiber guides, but they are more costly and also more difficult

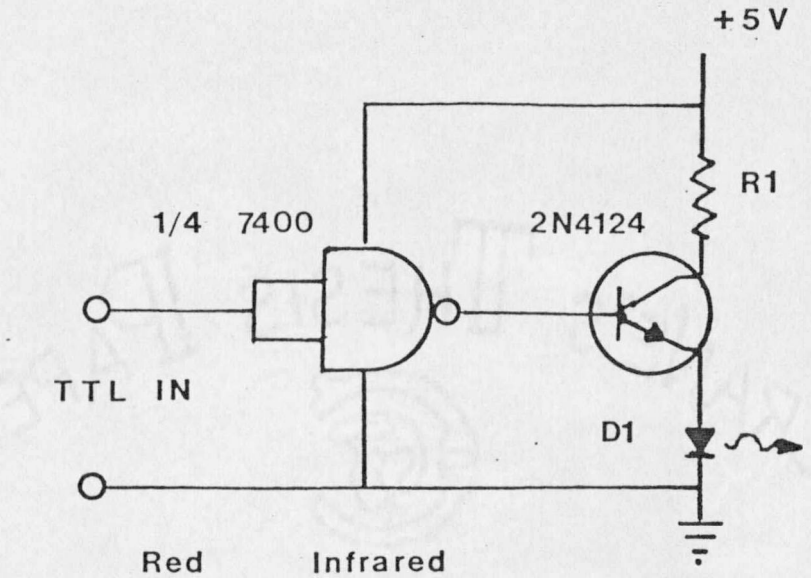
to properly terminate. Because of this, single fiber cable was chosen for this project. Again, it appears from advertising in electronics magazines that the choice of single fiber cables over multifiber cables has been made by nearly every data link manufacturer.

The Fiberoptic Transmitter

The schematic for the transmitter for the visible LED was provided by Valtec, the fiberoptic cable manufacturer. This was modified slightly for the infrared LED which required a different drive current from the red LED. These circuits are shown in Figure 2. The NAND gate provides hysteresis and drive current for the transistor, which can supply the higher current required by the light emitting diode. The different LED's used are shown along with the required current limiting resistors. The transmitter has worked well from the beginning of the evaluation, and this circuit has not been changed.

The Fiberoptic Receiver

This circuit was the most difficult of all of the circuits to design. The chief reason for this was the very small signal produced by the PIN diodes. The fiberoptics cable manufacturer had provided a schematic for a receiver with the cable that was used in the project. This receiver was constructed and it did work, but it was very sensitive to just about any change in operating conditions. Most of these pro-



	Red	Infrared	
D1	FLV104	MFOE100	
R1	33	82	OHMS

Figure 2 Transmitter

blems were found to be coupling problems between the power supply and various parts of the circuit. To avoid these problems, a more conservative design was adopted. This design is presented in Figure 3. Great care was taken to not only decouple the power supply from the operational amplifier supplies, but to also decouple the various parts of the circuit from one another.

The input signal is very small, having been measured at .5 microamperes for the red LED/detector pair. The input resistance of the op amp A1 is on the order of 70 megohms, so virtually all of this current passes through the 510 kohm resistor. An input resistance this large may be criticized as introducing thermal noise and converting any

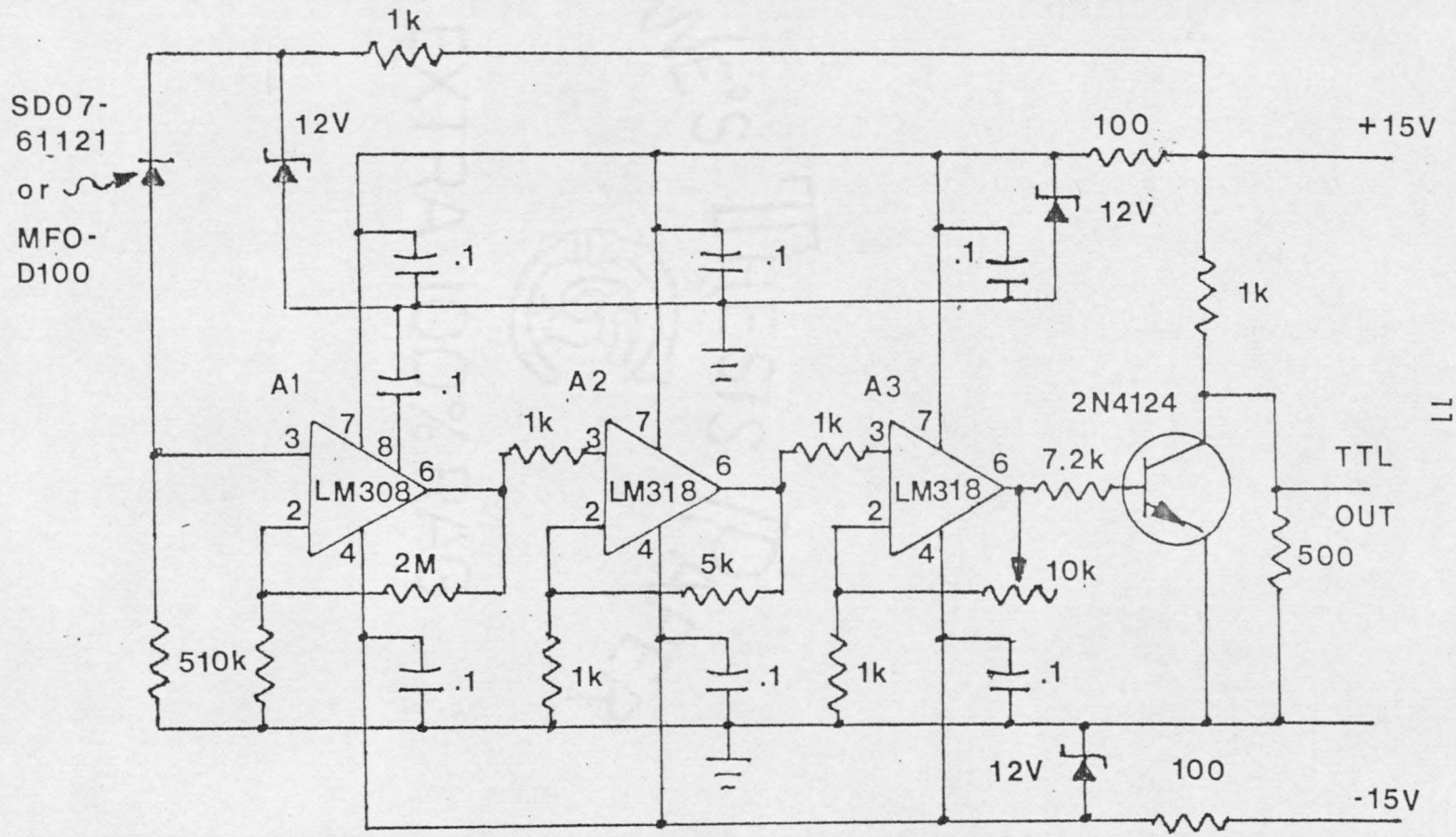


Figure 3 Fiberoptic Receiver

stray currents into voltage drops, but as long as this introduced noise does not create spurious logic transitions at the output, it is a small price to pay for decreasing the amount of voltage gain the op amps must supply. The op amp A1 must have very small offset current and bias current due to the already mentioned small signal. The LM308 op amp selected for this task meets these requirements easily with .4nA and 3nA respectively (8). The voltage gain was kept low in this stage since the LM308 does not have a gain-bandwidth product much better than 1 MHz and the circuit must have an overall bandwidth of 100 kHz. Once the signal has been amplified by the LM308, there is sufficient drive current, and the bias and offset current specifications can be relaxed somewhat. For the next two amplifiers, LM308 op amps were used. These amplifiers have a gain-bandwidth product of 15 MHz (8), and can easily be pushed to higher voltage gains than the LM308. The last of these two amplifiers has a trimpot in the feedback loop to provide an adjustable gain. This is to compensate for variable input levels caused by different detector/source pairs or losses due to variable fiberoptic lengths. The final stage of the receiver is the output transistor. This transistor accomplishes two things. It not only provides some wave shaping that removes noise introduced in the amplification and cleans up the rising and falling edges of the waveform, but provides a signal inversion to make up for the inversion introduced by the fiberoptics transmitter.

There were two source detector pairs that were used in the data link. The pair that operates in the red end of the spectrum is a Spectronics SD0761121 PIN diode and a Fairchild FLV104 LED. This was the pair supplied by the fiberoptic cable manufacturer. The other detector/source pair operates at infrared wavelengths. These are both Motorola devices and the part numbers are MFOD100 and MFOE200 respectively. The infrared pair gives a slightly stronger signal, which is probably due to the higher semiconductor efficiency at this wavelength (11). It is interesting to note that either detector will work with either LED, but it is often necessary to readjust the gain in the receiver to get maximum performance when this is done. If the detector and LED are matched, there is no need to readjust the gain when going from one detector/source pair to the other.

CHAPTER III

THE INTERFACE CIRCUITS AND TRANSDUCERS

Once the data link was operating, interface circuitry was constructed to allow generation of a pulse train from an input voltage. This pulse train could then be fed into the fiberoptic data link. The input voltage was to be derived from some physical quantity that a transducer could change into an electrical signal. This electrical signal had to be transformed into a voltage, since the only A-to-D conversion available is a voltage-to-frequency converter. Provisions had to be made to scale and offset this voltage so that the maximum output of the transducer would correspond to the maximum allowed input voltage to the voltage-to-frequency converter.

The voltage-to-frequency converter will be the first interface circuit discussed. The description of the current transducers and their interface circuitry will follow, and the chapter will close on a similar description for the temperature transducer.

The Voltage-to-Frequency Converter

The voltage-to-frequency converter used for the analog-to-digital conversion is based around an Analog Devices AD527 monolithic voltage-to-frequency converter (1). This circuit was selected over other integrated circuits because of the small number of additional components necsssary for operation (6 additional components), its good stability and linearity, and the good test and application documentation provided

provided by the manufacturer. A monolithic voltage-to frequency converter was chosen over a hybrid or discrete component device. Integrated circuits recently introduced on the market are very much less expensive in terms of cost and power consumption than either of the other types, at very little sacrifice in performance. The schematic for the circuit used is given in Figure 4. This circuit is based on circuit suggestions provided by Analog Devices.

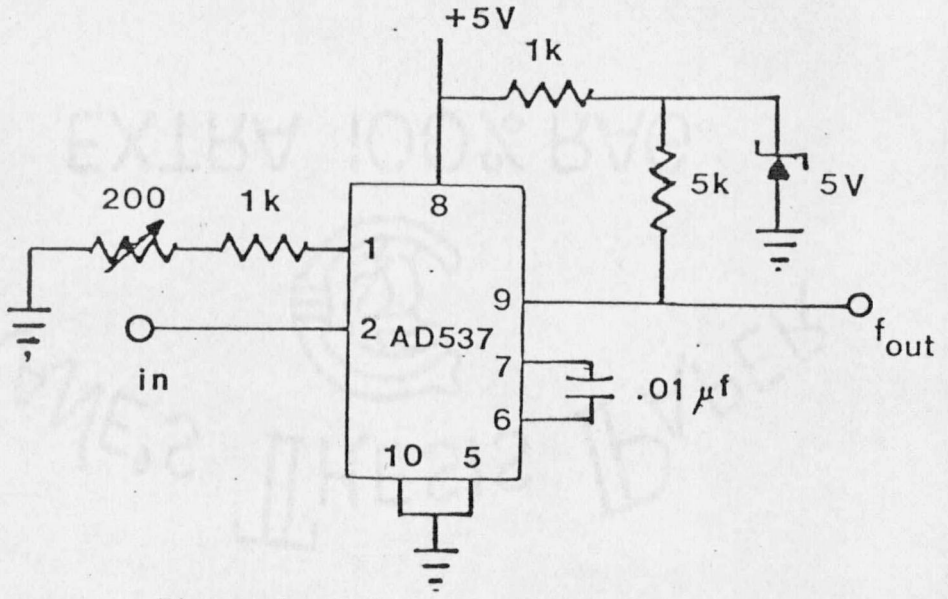


Figure 4 Voltage-to-Frequency Converter

The internal operation of the device is not dramatically different from other voltage-to-frequency converters. A current source controlled by the input voltage feeds charge into an integrator composed of an external capacitor and an internal operational amplifier. The voltage

developed across the capacitor is compared to a reference voltage derived on the chip. When this voltage exceeds the reference voltage, an astable multivibrator is triggered. The output from the astable multivibrator in turn drives an output amplifier which provides higher driver current. The frequency of operation is determined by the size of the external capacitor and also the size of the external resistors used to adjust the magnitude of the current from the voltage-controlled current source.

The linearity and temperature stability of the circuit is greatly dependent on the quality of the external resistors and capacitor. These components must have very low temperature coefficients and the timing capacitor must have a very low effective series resistance if the circuit is to meet the manufacturer's specifications.

The Current Transducers

Two types of transducers were used to make the current level measurements on a conductor. One of these transducers was an Ohio Semitronics CT50L DC to 5 kilohertz transducer (9). This transducer is based on the well known Hall Effect (4). Unlike conventional current transformers which depend on the change of the current and a change in its accompanying magnetic field, this transducer directly measures the intensity of the magnetic field. Thus, its time response is not limited by the inductance present in a sensing coil, but only by the semi-

conductor transport processes and the properties of the material used to concentrate the magnetic field around the Hall Effect sensor. This enables the transducer to have a response linear to within .5% over a range from D.C. to 5 KHz. The upper figure is a function of this particular Hall Effect device and not the Hall Effect itself.

There are two disadvantages to this type of transducer. The first disadvantage is that it must be supplied with a constant excitation current (nominally .1 A). Variations in this excitation current show up as unwanted variations in the transducer output. The second disadvantage is that the Hall Effect device used is a semiconductor device and the mobility of the charge carriers in the semiconductor is dependent on the material temperature. Since the voltage produced by the Hall Effect is dependent on the mobility of the charge carriers, variations in temperature may also cause unwanted output variations.

Neither of these disadvantages is insurmountable. The excitation current can be derived from a variety of sources. Possible sources are batteries, power supplies, or in the case of measuring high A.C. current, the excitation current could be transformed right off the power line itself. The problem of temperature dependence is also easily solved. Since the data is taken in digital form, the temperature dependence can be removed mathematically. Methods to do this are well known (3) and usually consist of fitting experimentally derived error data to a mathematical function. This function is either

directly evaluated by the microprocessor or is built into a lookup table that the microprocessor can reference. Both of these error removal methods require that the temperature of the transducer be known. Temperature measuring methods are discussed later in this section.

The second transducer used was a conventional current transformer, specifically a Westinghouse Current Transformer type CMS. This is a transformer similar to those presently used by the power companies in determining current levels in transmission equipment. The only modification required is the insertion of a small shunt resistance across the transformer output terminals. This shunt resistor (.024 ohms) was chosen so that the voltage drop across it would be compatible with the output voltage developed by the Hall Effect device.

Both transducers are operated so that a current of 50 amps causes a 30 millivolt peak output. The 50 amp figure was chosen for experimental convenience. Hall Effect transducers are available to measure currents up to 8000 amperes (9). Current transformers are available for a similar range (15).

A 50 amp A.C. transmission line was simulated by passing a 5 amp A.C. current through a ten turn coil wound through the active area of each transducer. An A.C. current source consisting of a variable transformer and a 1 ohm load resistance provided the current to the transducers. The current level in the "transmission line" was

then measured by recording the voltage dropped across the one ohm load and multiplying by ten.

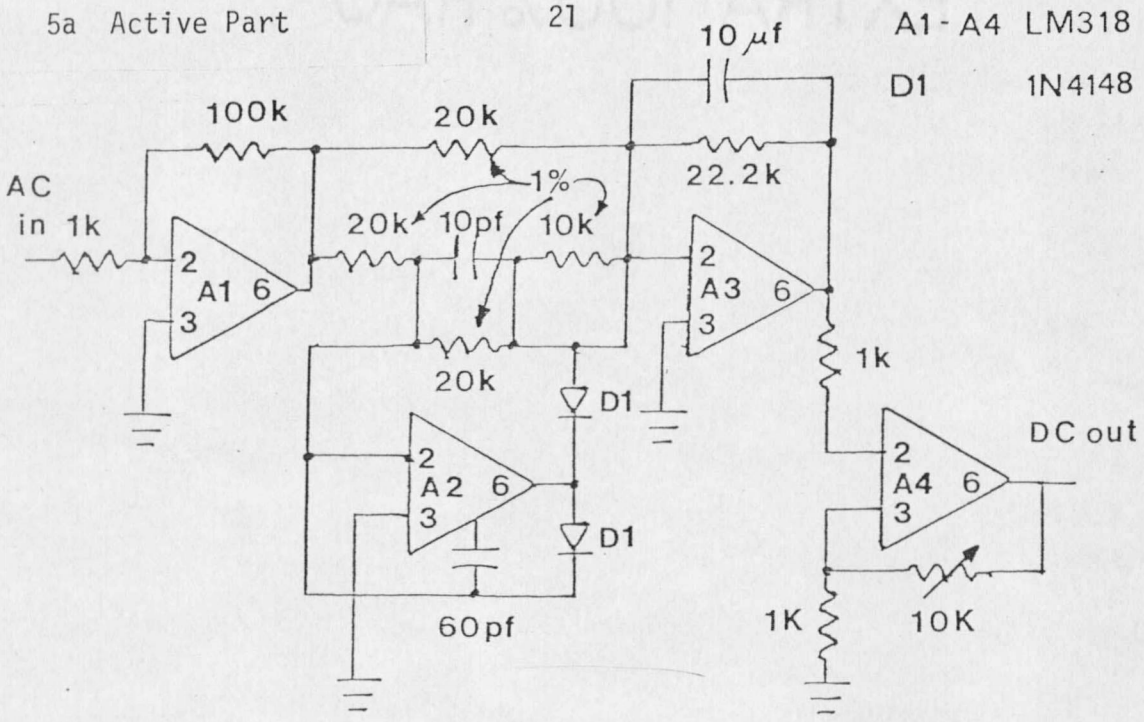
The Rectifier and Filter

The signal coming from the current transducers has to be rectified and amplified before the signal could be used by the voltage-to-frequency converter, which requires a D.C. input voltage. Since the current transducers have a 30 millivolt peak output, direct rectification by a diode bridge of any sort will introduce an error. Suppose that the signal is first amplified to 10 volts peak prior to rectifying. If the signal passes through a full wave bridge rectifier, two forward diode voltage drops are subtracted from the signal. For silicon diodes this translates to somewhere in the vicinity of 1.2 to 1.5 volts subtracted from the peak voltage. For voltages less than peak, the error will be less than this, but it will be a nonlinear proportion of the maximum error due to the nonlinear transfer characteristic of the diodes. The 1.5 volt maximum error corresponds to a 15% error, which is much to high for the accuracy desired. Amplifying the peak output voltage to 100 volts would reduce the percentage error by a factor of 10, but these higher voltages have gotten out of the voltage range of most common integrated circuit amplifiers. This argument precludes using the "brute force" methods of rectification common to power supplies. The solution to this problem is to use a rectifier that is configured to

utilize the diodes only to block the portion of the signal that is not wanted rather than block and pass the waveform. The circuit in Figure 5 is based around a circuit idea from a National Semiconductor application note (7). This circuit not only provides full wave rectification, but also contains a portion to smooth the rectified waveform and amplify the resulting D.C. voltage to match the voltage range of the output to the voltage range of the voltage-to-frequency converter. Although this circuit substantially alters A.C. waveforms, D.C. waveforms are just amplified and smoothed to an average D.C. level. The circuit operation and a derivation of its transfer function are provided in Appendix A.

The schematic for the amplifier and precision rectifier is provided in the two parts of Figure 5. One of these parts depicts the active circuitry, while the other part depicts the power supply decoupling scheme (2). This decoupling scheme may seem rather elaborate at first, but the accuracy of the entire instrument depends on the proper functioning of this one circuit. The circuit will be operating in close proximity to the voltage-to-frequency converter which can introduce switching spikes on the ground and power lines of the rectifier. The decoupling will help to absorb these spikes before they can get into the op amp's circuitry. In the case of an application such as an A.C. current monitor, this circuitry would be operating on a power line and be totally immersed in the 60 cycle

5a Active Part



5b Decoupling Part

R = 50 ohms
C1= .1μf
C2= 100 pf

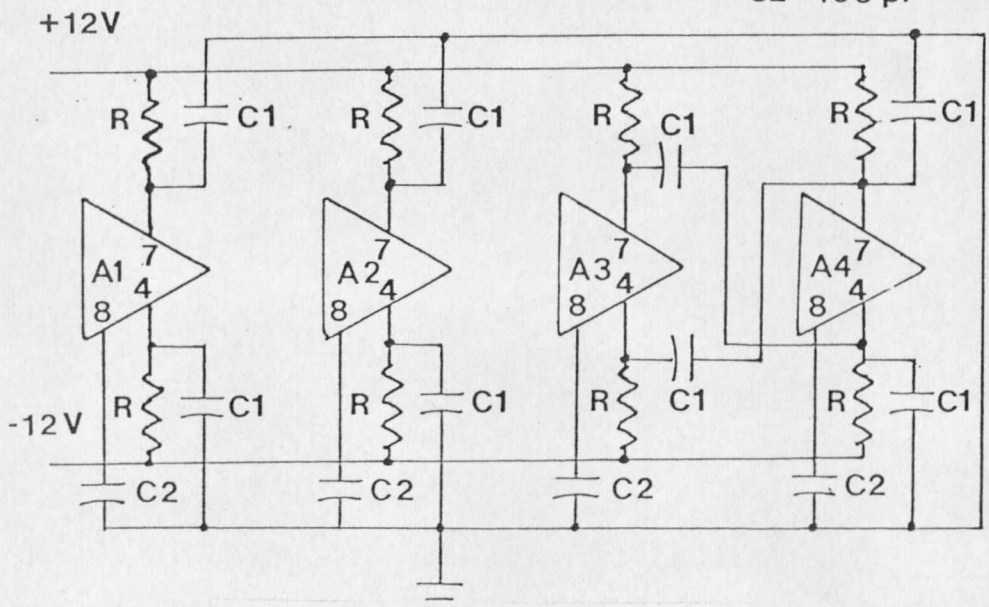


Figure 5 Rectifier/Filter

electromagnetic field of the power line. These fields may induce noise in the circuit at the 60 Hz line frequency. This noise is totally indistinguishable from a valid signal if it gets onto either the input or the output of any of the amplifiers, so it is essential that it be kept away from the active elements at all costs.

The Temperature Transducer

Since operation of the Hall Effect Device may require an accurate knowledge of the temperature, a temperature transducer would be required. Several different integrated circuit transducers were acquired, connected to whatever supporting circuitry was required, and this combination tested in an environmental chamber. On the basis of these tests, the Analog Devices AD590 two terminal IC temperature transducer was selected as the transducer to be used (1). The parameters used in selecting this transducer were ease of use with the voltage-to-frequency converter and accuracy.

The temperature transducer has only two types of errors in the current output that is the response to temperature. These errors are offset errors and nonlinearity errors. There are several grades of this device available. In general, a more expensive grade has a smaller offset voltage error than a less expensive grade. The nonlinearity errors are the same from one grade to the next.

The circuit that the AD590 is used in is shown in Figure 6.

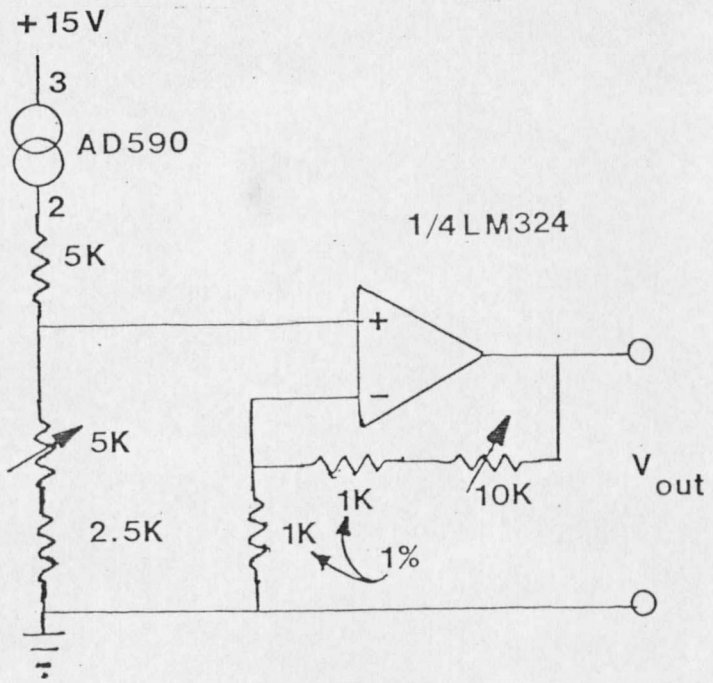


Figure 6 Temperature Sensor

There are two variable resistors in the schematic. The 5k resistor is used to remove the offset inherent in the device. The 10k variable resistor in the op amp feedback loop is provided to match the output voltage produced by the maximum anticipated temperature to the maximum input voltage that the voltage-to-frequency converter is set for. This allows the desired temperature range to be spread over a larger fraction of the voltage-to-frequency converter's operating range.

CHAPTER IV

SOFTWARE DEVELOPMENT

Before discussing the software to control the hardware that has been developed, it is necessary to describe the hardware environment in which the software operates. This environment consists of four pieces of equipment, all functioning as one microprocessor-based data acquisition system. The data acquisition system need not be this complex, this organization primarily reflects the equipment available for this project. The equipment consists of an INTEL PROMPT80 microcomputer, a Western Telecomputing Corporation WTC-800 computer system, a WTC-700 data acquisition system contains a four channel A-to-D converter (used to calibrate the data link instrumentation), a four channel event counter and the controller card that is connected to the WTC-800 system. This controller card provides all of the bus timing and addressing for the WTC-700 system. This system has a bus structure that is markedly different from the bus used in the WTC-800 system. The WTC-700 card cage contains slots for the controller card (already mentioned), two power supply cards, and 16 slots for data acquisition cards. Each of these slots is separately addressed, so the position of a specific card in the bus determines the software address of the card. Each of the cards in this system has as its output an ASCII string representing the quantity measured. The controller card enables this string to be read off one byte at a time.

Software Routines

The software written for this research project consists of two basic types of programs. These are the programs that interface directly with the hardware (called hardware drivers) and the program that coordinates these interface programs. This modular program structure (6) and how it interfaces with the hardware is illustrated in Figure 7.

This type of structure was chosen for its flexibility. If the hardware/software system is to do a different task, the hardware drivers do not need to be changed. The coordinating program is the only module that has to be altered, and then most of the changes will only be in the order that the supporting modules are called by the coordinating program. Another advantage of this type of structure from the programmer's point-of-view is that the programmer is not constrained to any one programming language. Routines that lend themselves to register manipulation, or that must execute very rapidly may be written in assembly language. Routines that involve data processing may be written in a higher level language that allows arrays and/or data structures (i.e., BASIC, PLM/80 or PASCAL). Routines that involve mathematical processing may be written in a formula oriented language such as FORTRAN or ALGOL.

The routines presented in this thesis are written in a combination of PLM/80 and INTEL 8080 assembly language. The choice of a certain language for one of the routines reflects not only the above mentioned considerations, but also the fact that at the beginning of the project

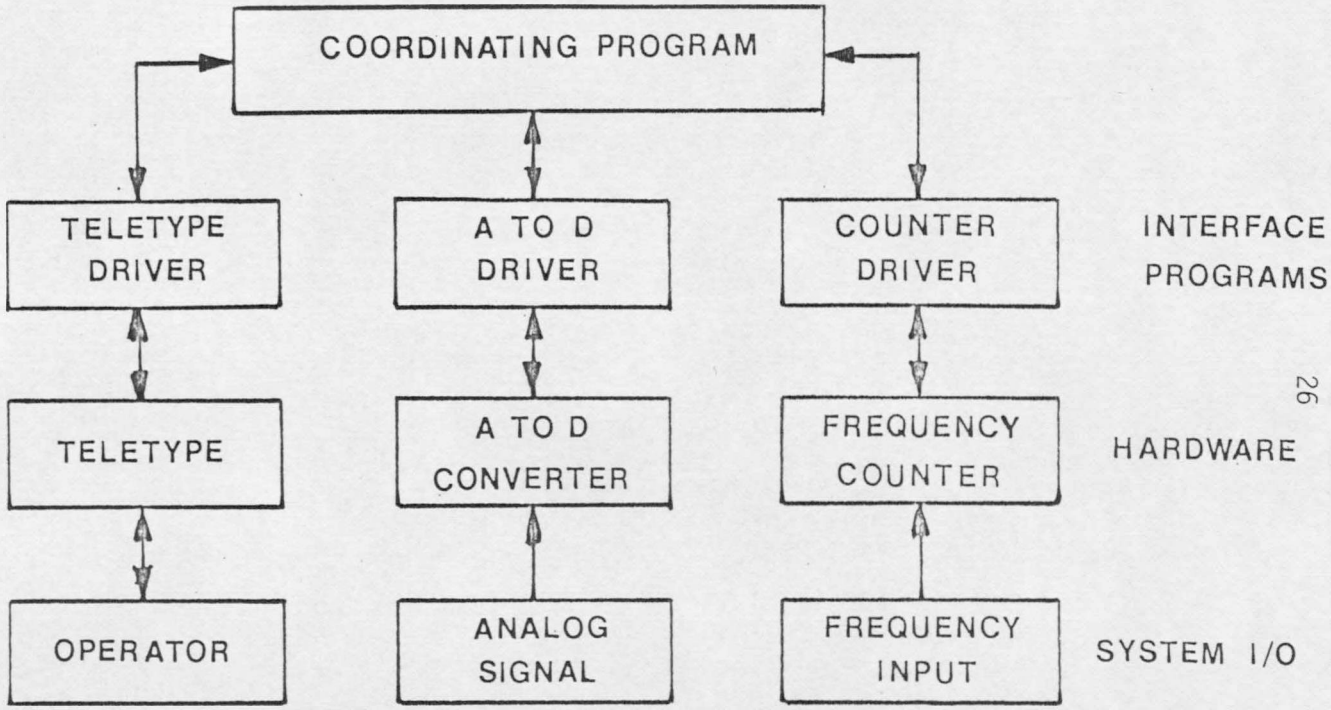


Figure 7 Software/Hardware Interface

assembly language was the only language resident in the MDS 230 software development system used to develop all the software in this project. The author also recognizes that the reader of this thesis may not have access to a PLM/80 compiler or be familiar with the PLM/80 syntax. Hence, all PLM/80 routines are presented in the appendix in two forms. The first of these forms is a listing of the PLM/80 source code. The second form is a listing of the same source code with the compiler generated assembly language mnemonics included. Appendix B also contains a description of how PLM/80 and assembly language programs are interfaced. This part of the appendix should make the register and stack manipulation used prior to and after calls easier to understand. The program listings are contained in appendices D through G.

The software routines for this project must be able to:

(a) receive input from the operator as to what sensor(s) to read and when to read them

(b) read the sensors

(c) process the readings

(d) report the results to the operator

These requirements demand that the coordinating program be able to interface with drivers for the following hardware:

(a) A teletype

(b) The frequency counter connected with the fiber optics data link

(c) An A-to-D converter (for calibration only)

Before discussing individual hardware drivers, the operation and

construction of the coordination program will be considered.

The Coordinating Program

This program first interrogates the user to set up parameters that determine the initialization of the frequency counter, the number of times the readings will be taken, and the time delay between the readings. The program checks the initialization values provided by the operator in two ways. The values are first checked to see if they correspond to the format given in the interrogation message. If the values pass this first test, the values are converted from the ASCII strings that came from the teletype to either binary numbers (for the delay and repetition parameters), or binary coded decimal (BCD) numbers (for the counter initialization parameter). The magnitudes of the binary numbers are checked to insure that they fall within the range prescribed by the interrogation message. It is not necessary to do this for the BCD numbers since if the input number string passes the first test, it will fall within the proper range. Should any of these values fail either of these tests, the appropriate interrogation message is repeated until the operator can provide a satisfactory response. There is only one assumption made in this program. It is assumed that the operator will enter numbers on the teletype and not some other character. The program will function should characters other than numbers be supplied, but no error

message will be generated and the results will in general be different from those anticipated.

Once the initilization parameters have been set, the coordinating program will call the hardware drivers in the proper order to read the sensor and print the results on the teletype. The program then enters a software timing loop to generate a delay between readings. The duration of this timing loop was one of the parameters specified in the initilization sequence. When this loop times out, the program will make another reading if it has not completed the specified number of readings. When the program has finished these readings, it re-enters the initialization sequence and waits for further operator input.

At this point, it is necessary to make some additional comments on the actual time delay between readings of the frequency counter. This delay is determined by three factors:

(1) The length of time the A-to-D converter requires to make a conversion. This also includes delays in the accompanying software routines.

(2) The length of time required to print out or store the results of a conversion.

(3) The delay introduced by the software controlled timing loops.

The user has very little control over the first of these factors. The delay caused by the second factor can be minimized, and the delay caused by the third factor is completely controllable.

The delay caused by the A-to-D conversion is directly proportional to the number placed in counter zero to specify the length of time counter one downcounts. The delay is also dependent on the clock rate fed into counter zero. The hardware configuration chosen for the WTC-520A board gives a clock period of 25.6 microseconds. At this clock rate, if counter zero is initialized with a count of 8200, counter one will contain a count corresponding to 1/4 the actual frequency of the incoming pulse string. These two factors imply that the length of time required for one conversion is .21 second. If there were no other appreciable delays in the system, the maximum sampling rate would then be approximately 4.76 conversions per second.

When the delays caused by the teletype are accounted for, this rate drops dramatically. If the teletype does no more than print out the four digits of the conversion, a carriage return and a line feed, then the total delay is .81 second and the rate is reduced to 1.235 conversions/second. It is quite likely that more information than this would be printed on the teletype, with a corresponding reduction in sampling rate.

The maximum sampling rate of 4.76 conversions per second could be very nearly approached if the data were stored directly in memory instead of being immediately printed on the teletype. It was assumed in calculating the above sampling rate that the time required by the software is negligible compared to the hardware time. A routine to

place the data directly in memory would presumably wait in a loop until the conversion was completed, read the appropriate counter, store the BCD information in memory, store a data delimiter in memory, check to see if the counter has been read the required number of times and, if not, begin another conversion and jump to the wait loop. A portion of code to do just this is presented in Appendix C. This routine has not actually been run on the DAS, as it's purpose is merely to illustrate a point. This point is to provide an idea of the length of time a similar program would require to store the information in memory, and see if this time is significant compared to the hardware conversion time.

Each pass through this routine takes a maximum of 220 clock cycles. With a clock rate of 2 MHz, this is an execution time of:

$$220 \times 1/(2 \times 10^{6}) = .11 \times 10^{-3} \text{ sec}$$

Since the A-to-D conversion time alone is 210×10^{-3} sec., this additional processing time is relatively inconsequential. This example justifies the earlier assumption that the time consumed in the hardware interfacing routines is not significant compared to the conversion time.

The Hardware Drivers

Now that the overall operation of the coordinating program is understood, the program modules that the coordinating program inter-

faces to will be considered. Details on interfacing software to these routines and the listings of the routines are contained in the Appendices D through G.

The Teletype Driver. The teletype is the main I/O device in the system. Contained in the PROMPT 80 computer used in this project is an INTEL 8251 Universal Synchronous/Asynchronous Receiver/Transmitter (USART) (5) and associated circuitry to drive a teletype on 20 ma. current loops. All of the communications to and from the teletype are done via this USART. The USART appears to the microprocessor as two input/output ports. The port at address 236 serves as the port through which the characters are transmitted to and from the teletype. These routines are derived from a set of assembly language routines prepared earlier by L. K. Smith and D. K. Weaver (10). This teletype driver does not use interrupts and thus must stay in a waiting loop during character transmission or reception. These routines are now written in a mixture of PLM/80 and assembly language and have been modified both in logic structure and buffer structure. Assembly language was used to decrease size and speed execution of some of the routines.

The routines require that a buffer area in memory have previously been defined. This buffer area is up to 256 bytes long. The first byte in the buffer (lowest address) is reserved for the total number of filled bytes in the buffer. It is zero if the buffer is empty. There are no restrictions on the contents of the rest of the buffer.

The routines themselves are presented in Appendix D, along with descriptions of the parameters required by each routine. There are provisions for calling the important routines from either assembly language or PLM/80. The routines not only enable the teletype to be used as an I/O device, but also allow rudimentary line editing to be done on the line presently in the buffer. Characters may be deleted beginning with the last character entered by pressing the rubout key. The contents of the entire buffer may be erased leaving an empty buffer by using CONTROL-X. If the line being entered exceeds the buffer length, the routines will quit echoing the characters after the buffer is full. The carriage return key terminates the line input.

The Frequency Board Driver. The next interface routine to be discussed will be the driver for the WTC-520A frequency board shown in Figure 8 (14). The WTC-520A frequency counter board is designed around the INTEL 8253 counter (5). There are three sixteen bit down-counters on this chip. Counters 0 and 1 are used on the WTC-520A board to form a gated frequency counter. Each counter has a gate enable input that controls when the input pulse string is to be counted, and an input line that indicates when the counter reaches zero. Counter zero on this chip and a D flipflop are used to control the length of time that counter one may count the incoming pulse string. To successfully operate the frequency board it is necessary to:

(1) Reset the output of counter zero.

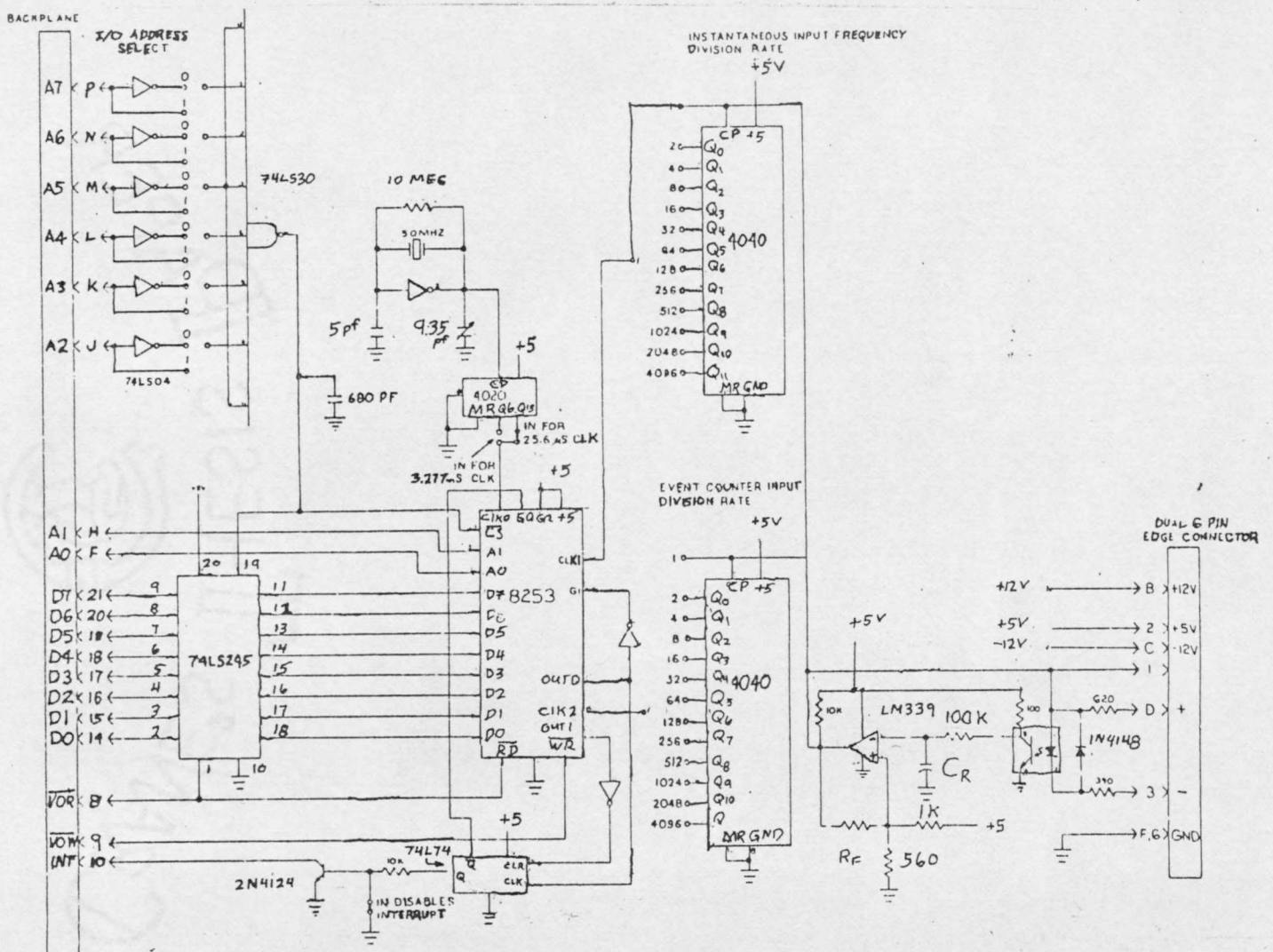


Figure 8 WTC 520 Board (Courtesy of Western Telecomputing)

(2) Preset the D flipflop by forcing counter one's output high, then low. At this point, both counters are ready to begin counting.

(3) Load counter one with the number corresponding to the maximum number the counter can hold in the mode it is being operated in.

(4) Load counter zero with the number corresponding to the time interval that counter one is to count for. This starts both of the counters.

When counter zero reaches zero, the D flipflop will be triggered, and both counters will stop counting. The counters may then be read at any time. Counter zero will, of course, contain zero, but counter one will contain a value corresponding to the number of counts subtracted from the number initially loaded into this counter. It is the responsibility of the programmer to determine that the counting interval is always short enough to insure that counter one will not count more pulses than the number initially loaded into the counter.

Since in this application the microprocessor-based system has nothing to do until the frequency counter is through making it's conversion, the driving routine merely waits in a loop checking to see when counter zero's contents are zero. When this is true, the contents of counter one are read and subtracted from the value initially loaded into counter one. The result of this subtraction is equal to the number of pulses input to the board during the time it took counter zero to reach zero. This number is changed into an ASCII string and placed in the appropriate buffer.

The A-to-D Converter Driver. The third interface driver controls a WTC-700-120A analog to digital conversion board (13). This board resides in a WTC-700 Data Interface System connected to the WTC-800 156 Microprocessor Controller (13). This controller uses the INTEL 8255 parallel interface. The 8255 must be set into the proper mode to interface with the support electronics on the WTC-700 156 board. When the mode of the 8255 is initially set, the entire WTC-700 system is reset and then the individual cards in the DAS may be communicated with through the 8255.

The WTC-700-120A analog-to-digital converter may be read by signaling the card to make a conversion, waiting for the conversion to be completed, and then reading the results of the conversion. The results are in the form of a five byte ASCII string and are read from the WTC-700-120A using a byte by byte handshake. The details of this protocol are provided in the program listing contained in Appendix G.

CHAPTER V

EXPERIMENTAL RESULTS

After the hardware had been constructed, a series of tests were run to calibrate the operation of the sensors, fiberoptic data link, and DAS operating as one system. The process used in calibration involved two stages. The first stage involved individually adjusting the response of the sensors, voltage-to-frequency converter, data link, and software parameters until they responded reasonably close to design specifications. The second state involved deriving mathematical formulas to correct for remaining errors in the system.

To accomplish the first stage of calibration, adjustments that had been built into each sensor's circuitry were used to bring the response of each sensor reasonably close to the design specifications. An exact setting was not required or attempted, as the mathematical portion of the calibration procedure compensates for any residual errors. Next, the gain of any required interface stages was set so that the maximum voltage that would be delivered to the voltage-to-frequency converter was +10 volts. Then the voltage-to-frequency converter was set so that a 10 volt input would produce a 10 kilohertz output. Finally, a 10 kHz square reference was used to adjust the gate time on the gated counter so there was a simple relation between the value in the frequency counter at the end of the counting period and the frequency being counted. At this point, the first stage of the calibration was finished.

The second stage of the adjustment procedure uses formulas to correct for nonlinearity and remaining adjustment errors since the DAS contains a mircoprocessor that is capable of performing numerical calculations. A series of measurements are made over the range of operation of the sensor. As these measurements are made, they are transferred over the data link to the DAS where they are recorded. At the same time, the same measurements are made with another set of instruments that are known to be accurate. A mathematical formula may then be constructed relating the values recorded with the DAS to the actual quantity that was measured. The derived formula is a representation of the transfer function for the system as a whole.

There are several methods to construct this formula (3). The method chosen in this case involves adjusting the coefficients of a polynomial so that the polynomial corrects for the errors in the system. The values calculated with this polynomial are expected to produce more accurate measurements than the results without correction. A BASIC program was provided by Dr. Donald Weaver to calculate the polynomial coefficients required for this method. To demonstrate the increased accuracy of the polynomial method over the linear approximation method, the author wrote a BASIC program to provide a least squares fit of the same data to a linear relation. Printouts of the data used to calculate the coefficients for this polynomial and the coefficients calculated for a seventh degree polynomial are provided in a table for each of the sensors. A graph of

the calculated transfer function is also given, along with a graph of error between the linear approximation and the seventh degree polynomial. In interpreting these graphs, it is important to remember that the polynomial representation of the transfer function is not exact, but has a waviness to it due to the fact that a finite number of terms were used in the polynomial. This waviness is apparent in the plots of the error function which is the difference between the polynomial and linear representations. While the minor waviness is a mathematical construction, the plots do indicate the increase in accuracy that is available with the polynomial method.

The two transfer functions for the current sensors (Figures 9 and 10 and Tables 1 and 2) plot the current measured in the conductor versus the frequency of the signal measured from the data link. The circuitry had been adjusted so that 5 amperes through the conductor should produce a 10 kilohertz reading with the DAS gated counter. Neither circuit produces exactly this amount. These discrepancies are due to the method used in the initial hardware adjustments. Readjusting the hardware to eliminate these discrepancies is unnecessary since the polynomial representation of the transfer function for the readjusted hardware would be just as accurate as the present polynomial.

The transfer function for the temperature sensor is given in Figure 11 and Table 3. This transfer function differs from the other transfer functions in that the output of the device was not adjusted so

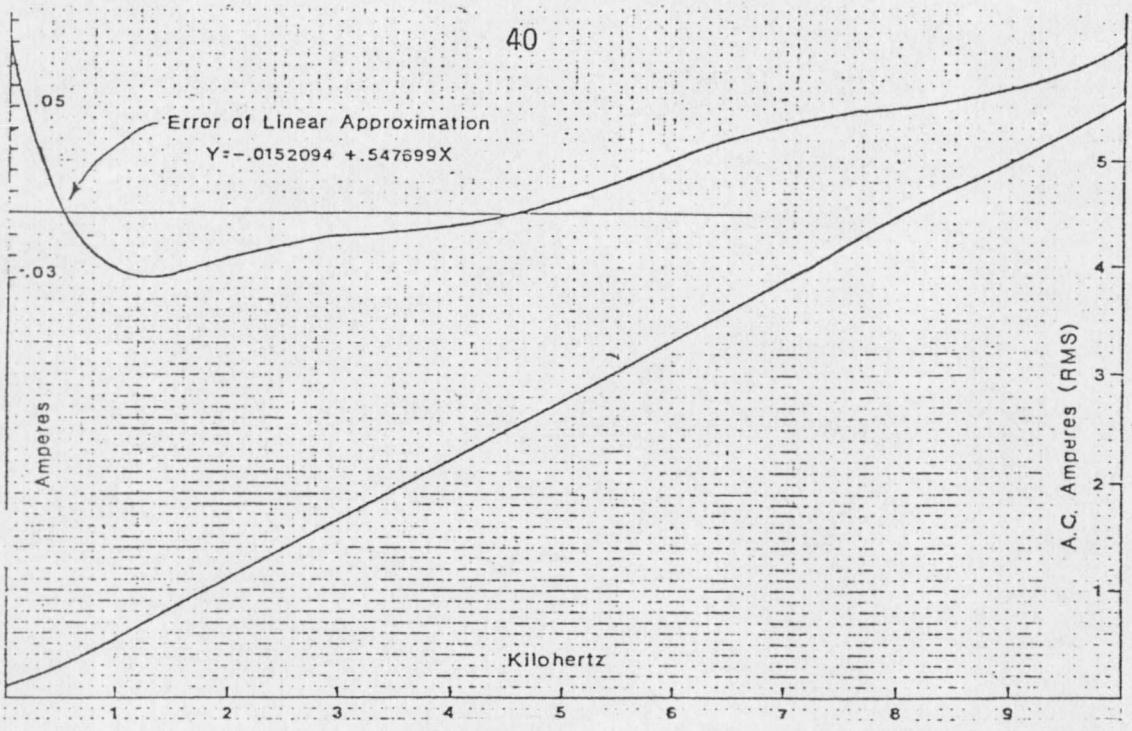


Figure 9 Transfer Function for Hall Effect Device & Data Link

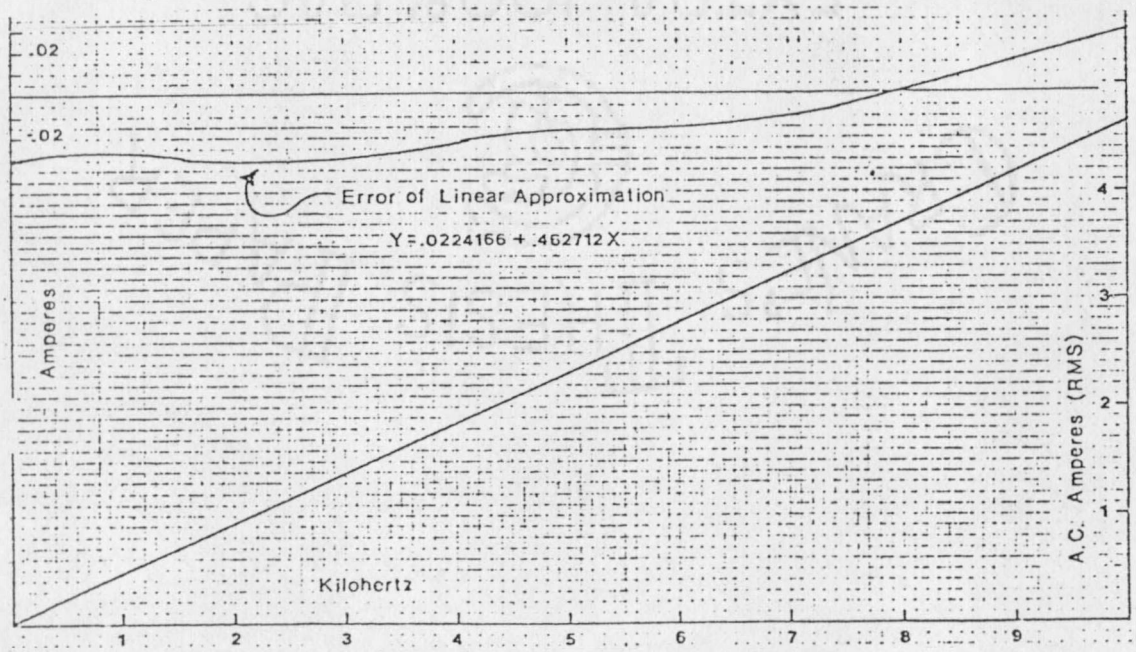


Figure 10 Transfer Function for Current Transformer & Data Link

DATA COEFFICIENTS ARE COMPUTED FROM

I	X(I)	Y(I)	I	X(I)	Y(I)
1	.29	.214	2	.544	.292
3	.984	.5355	4	1.494	.8168
5	2	1.094	6	2.538	1.373
7	2.988	1.644	8	3.48	1.915
9	4.048	2.231	10	4.504	2.485
11	5.044	2.787	12	5.51	3.045
13	6.04	3.355	14	6.514	3.613
15	6.98	3.884	16	7.52	4.181
17	7.978	4.439	18	8.528	4.735
19	9.038	5.03	20	9.524	5.29
21	9.944	5.548	22	10.508	5.858

COEFFICIENTS OF POLYNOMIAL SUMMATION A(I)*X**I

I	A(I)
0	.106290860395553
1	.27675797880948
2	.230824938867842
3	-.0944872305830564
4	.020908734981913
5	-2.5305582678523D-03
6	1.57327492499476D-04
7	-3.9254887314146D-06

Table 1 Hall Effect Device Coefficients

DATA COEFFICIENTS ARE COMPUTED FROM

I	X(I)	Y(I)	I	X(I)	Y(I)
1	.274	.1187	2	.538	.2413
3	1.034	.4723	4	1.55	.7123
5	2.08	.9535	6	2.53	1.156
7	3.04	1.403	8	3.518	1.623
9	4.04	1.87	10	4.5	2.081
11	5.024	2.329	12	5.564	2.581
13	6.054	2.813	14	6.504	3.019
15	7.104	3.303	16	7.518	3.497
17	7.988	3.716	18	8.51	3.974
19	9.05	4.219	20	9.46	4.426
21	10.04	4.697	22	10.512	4.916

COEFFICIENTS OF POLYNOMIAL SUMMATION A(I)*X**I

I	A(I)
0	-.0142191448561888
1	.489367371176304
2	-.0279904147401379
3	.0118919854596125
4	-2.38428291814793D-03
5	2.407419265383D-04
6	-1.14098596465538D-05
7	1.88465239576853D-07

Table 2 Current Transformer Coefficients

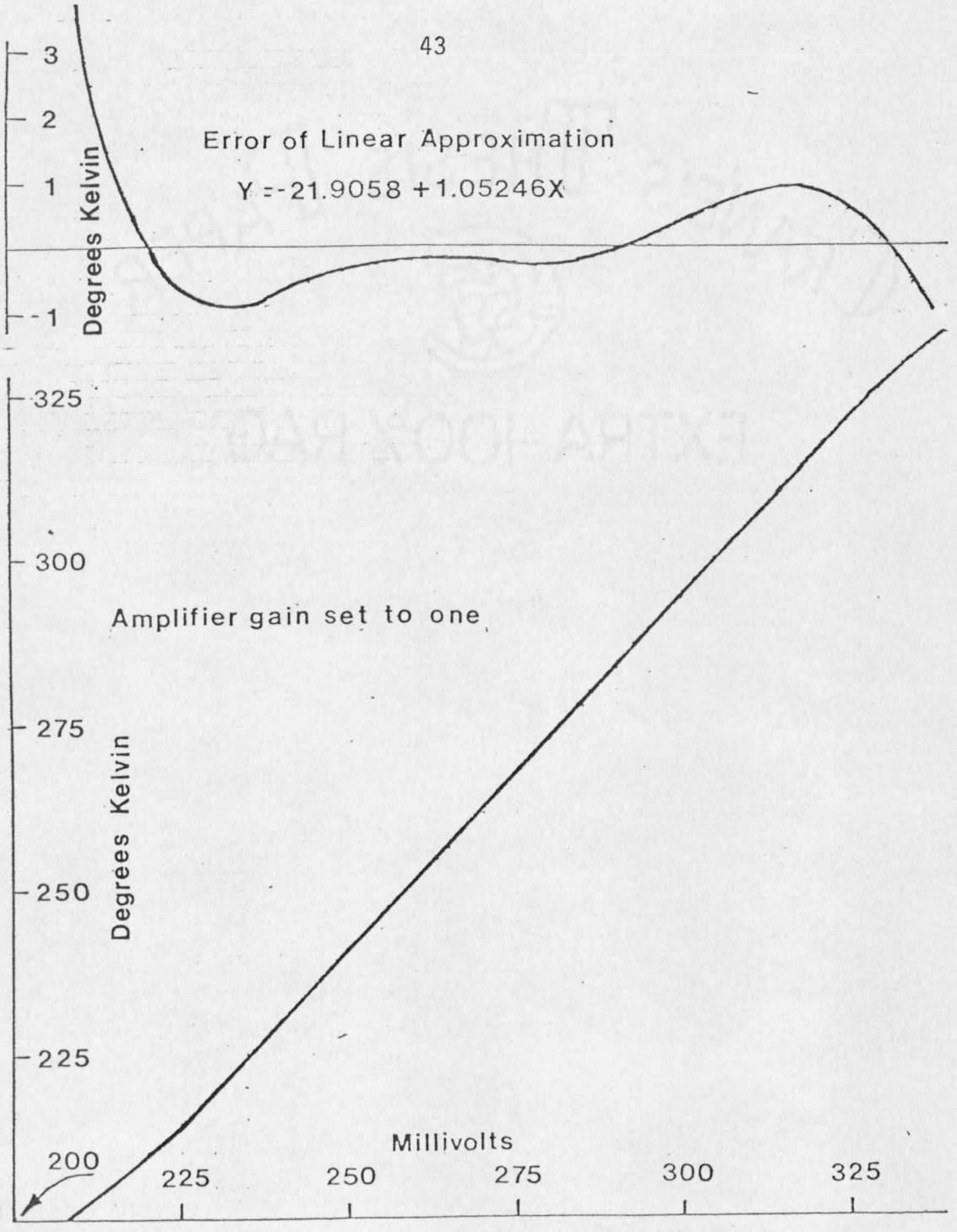


Figure 11 Transfer Function for Temperature Sensor

DATA COEFFICIENTS ARE COMPUTED FROM

I	X(I)	Y(I)	I	X(I)	Y(I)
1	212.8	204.25	2	219.4	209N52
3	223.8	213.5	4	229.8	219.15
5	234.6	224.1	6	241	230.7
7	245	234.4	8	251.3	241.2
9	256.1	246.1	10	261.6	251.45
11	252	245.95	12	262.1	255.85
13	268.7	261.35	14	274.1	267.05
15	278.3	270.75	16	284.6	277.15
17	280.9	272.6	18	293.2	285.75
19	297.8	291.9	20	302.8	297.8
21	308.7	304.45	22	315.3	309.1
23	314.1	310.6	24	314.9	311.15
25	316.1	312.2	26	320.9	316.35
27	325.9	321.35	28	329.99	325.75
29	337.8	332.4	30	344	338.9
31	349.4	344.2			

COEFFICIENTS OF POLYNOMIAL SUMMATION A(I)*X**I

I	A(I)
0	-75614.7574477737
1	2209.42819228466
2	-27.0187201840786
3	.180187711624059
4	-7.0942821931079D-04
5	1.65235955637477D-06
6	-2.11131637812401D-09
7	1.14303444626439D-12

Table 3 Temperature Sensor Coefficients

that it would produce a maximum of 10 volts. This was done because there would be few applications in which the full temperature range of the device would be needed. The offset and gain adjustments are used to spread the range of interest over a larger portion of the voltage-to-frequency converter input range. An offset (measured in millivolts) adds or subtracts this quantity from the constant term of the polynomial approximation. Changing the gain multiplies all of the coefficients by a like amount. Neither of these operations changes the general form of the transfer function. This one plot will suffice for any temperature in the AD590's rated range. The curvatures visible at each end of the transfer function are believed to occur since the LM324 op amp is at or past the edge of its specified temperature range, and performance is beginning to deteriorate.

The transfer function for the voltage-to-frequency converter is given in Figure 12 and Table 4. This transfer function displays more curvature than the integrated circuit's manufacturer had specified. This is due to the timing capacitor used in the device. The effective series resistance of this capacitor and temperature coefficient are higher than the specified amount. Tests were done that demonstrated a relationship between capacitor quality and degree of curvature of the voltage-to-frequency converter's transfer function. These tests showed that the voltage-to-frequency converter's performance was repeatable, and with the polynomial method of representing the transfer

function, that is all that is required of any sensor.

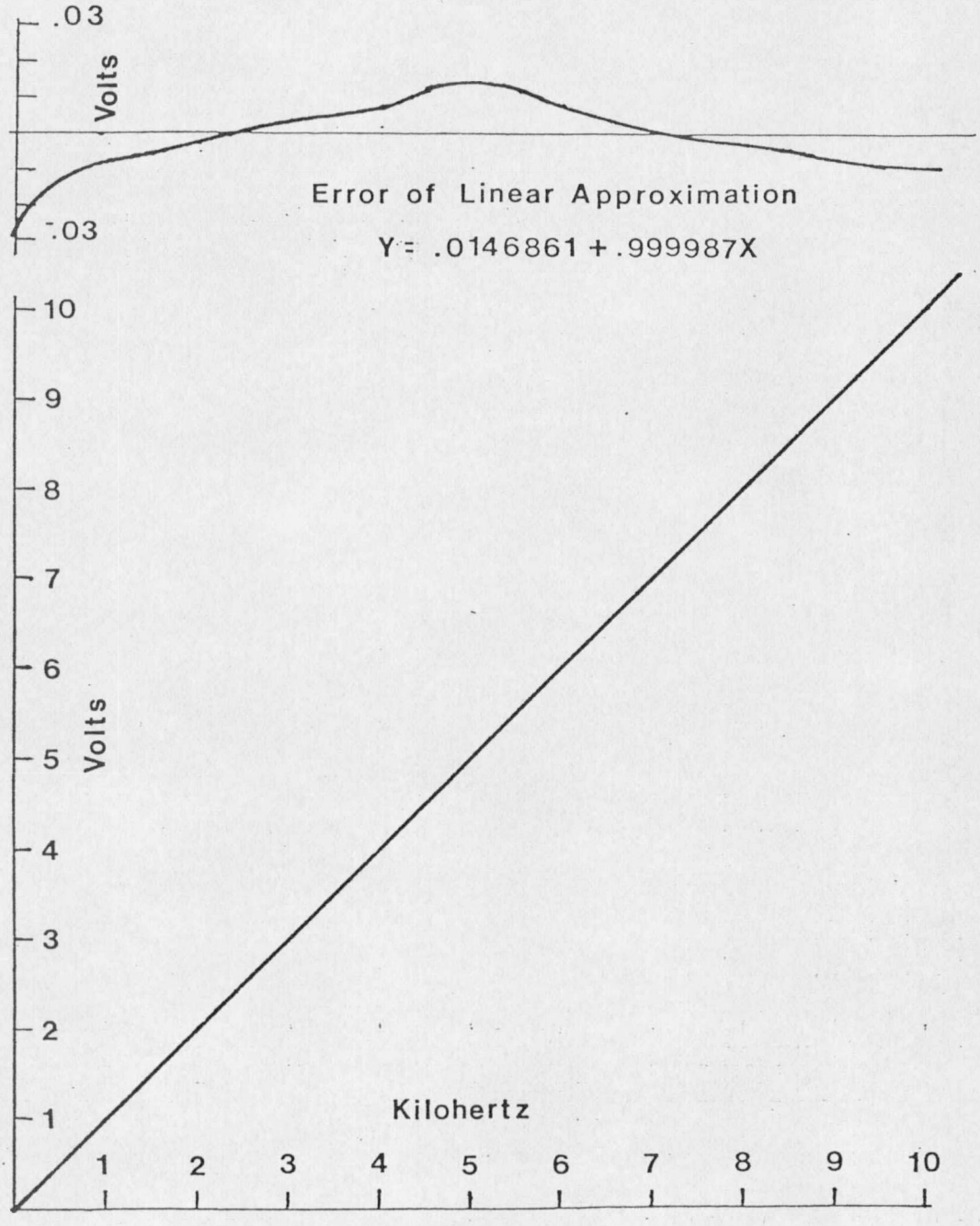


Figure 12 Transfer Function for Voltage-to-Frequency Converter

DATA COEFFICIENTS ARE COMPUTED FROM

I	X(I)	Y(I)	I	X(I)	Y(I)
1	.29	.29	2	.543	.543
3	.982	.988	4	1.495	1.504
5	2.001	2.02	6	2.536	2.54
7	2.987	3.02	8	3.48	3.49
9	4.047	4.06	10	4.505	4.572
11	5.043	5.06	12	5.51	5.53
13	6.04	6.07	14	6.52	6.513
15	6.981	7	16	7.52	7.54
17	7.977	8	18	8.526	8.52
19	9.039	9.04	20	9.522	9.53
21	9.945	9.95	22	10.509	10.52

COEFFICIENTS OF POLYNOMIAL SUMATION A(I)*X**I

I	A(I)
0	-.0145017747016736
1	1.05706590453779
2	-.0590085615643065
3	.0302319055502486
4	-7.70089530823416D-03
5	1.01726923287447D-03
6	-6.72024426283903D-05
7	1.76006946398493D-06

Table 4 Coefficients for Voltage-to-Frequency Converter

CHAPTER VI

SUMMARY

The purpose of this investigation was to build a low data rate fiberoptic data link and demonstrate methods of interfacing both sensors and digital equipment to the data link. To accomplish this, data links were constructed and operated for both infrared and red wavelengths. These data links had enough bandwidth to be able to handle square waves with frequencies up to 100 kHz, although the highest rate actually used for data transmission was only 10 kHz. A method of converting analog information to a digital quantity using a voltage-to-frequency converter was developed to convert D.C. voltages to square waves that could be fed into the fiberoptic data link. Recovery of the signal at the other end was accomplished with a microprocessor-based data acquisition system. This data acquisition system contains a frequency counter that was used to determine the frequency of the signal that had been transmitted over the fiberoptic data link. A teletype was also connected to and controlled by the data acquisition system. This teletype enabled the operator to specify parameters that controlled the way the frequency counter was read and also allowed results to be printed to the operator.

All of the data acquisition functions are controlled by the microprocessor in the system, which in turn is directed by programs stored in the data acquisition system memory. It is these programs that determine what the system as a whole will do.

There were two types of sensors that were developed and interfaced to the fiberoptic data link/data acquisition system described above. These sensors enabled measurements of current and temperature to be made. All of the sensors converted the particular quantity that they measured into a voltage that was compatible with the voltage-to-frequency converter.

Two types of current sensors were developed. The first type of current sensor was constructed using a conventional current transformer. This sensor could measure A.C. currents at power line frequencies. The second type of current sensor was constructed using a Hall Effect device. This sensor allowed currents to be measured from D.C. to 5000 Hz. Both of the current transducers used the same circuitry to interface to the voltage-to-frequency converter.

The temperature sensor was constructed using an integrated circuit temperature transducer. The output of this transducer was converted to a D.C. voltage by an interface circuit that allowed a large amount of latitude in selecting the temperature range that was to be measured. When these sensors had been constructed, they were attached to the voltage-to-frequency converter and the output of the voltage-to-frequency converter was fed into the fiberoptic data link/data acquisition system. The software written for the data acquisition system could then be used to obtain accurate calibration curves and functions for each of the sensors. These curves allow the output of

the frequency counter to be translated into the quantity that was measured.

It is felt that this experiment has demonstrated a valid instrumentation technique that allows accurate measurements to be made via a fiberoptic data link. These methods allow measurements to be made in an environment where electrical noise can mask or alter the information being transmitted.

The techniques developed in this thesis can be expanded into instruments that could be used in several measurement problems. An area that has already been suggested is the measurement of current levels in high voltage transmission lines. The specific problem suggested developing a current sensor as one of the demonstration sensors. Such measurements are presently done with current transformers combined with an analog meter or recorder. There is presently an effort to automate these measurements using computer-based equipment, which would benefit if the ideas presented in this thesis were applied to the automation equipment. Transmission of data through fiberoptics would provide an added measure of system reliability by isolating sensors from the microprocessor-based system. Transmission of the information on wires does not necessarily have this advantage.

Other areas where fiberoptics could be applied with some advantage are:

(1) Near machinery that has large alternating magnetic fields

such as motors, generators, or transformers.

(2) Areas such as meteorological towers that are exposed and susceptable to lightning strokes.

(3) Electrically noisy areas such as telephone switching facilities.

(4) Transmission of measurements over long distances (several tens or hundreds of meters) where ground loops and differences in potential between receiver and transmitter can occur.

(5) Devices like electrostatic particle accelerators or magnetohydrodynamic channels where great voltages exist between the sensor and the computer.

(6) Transmission of information through corrosive or potentially explosive areas like those occurring in chemical plants.

Some of these applications might multiplex several sensors on one data link. Other applications may require bidirectional communication between the sensor package and the DAS. In these cases, the chief difficulty is in devising a protocol that allows communication to occur on one data link, or with a minimum number of data links.

There are several areas outside the scope of this thesis that warrant further investigation. The system developed in this thesis should be compared with other methods of transmitting data through electrically hostile environments. This would give a better indication of when a fiberoptic system of this nature is cost effective.

Studies of this nature have been done for communication applications of fiberoptics (12), but there does not appear to be a similar study comparing fiberoptics with other isolation techniques.

Other areas that warrant further consideration are directed towards problems that would be encountered in converting the ideas presented in this thesis to a practical instrument capable of operating in a non-laboratory environment. One problem is that of supplying power to the sensor and fiberoptic transmitter outside of a laboratory environment. The reason for using fiberoptics is to provide electrical isolation between the transmitting and receiving ends of the data link. This dictates separate power supplies at each end of the data link. The transmitter may not always be located near a conventional power outlet. Operation on a high voltage transmission line is an excellent example of one of these cases. Here, the most practical method of obtaining power would be by transforming the power directly from the transmission line. Other applications may dictate solar cells or a rechargeable battery backup. The separate source of power for the sensors and transmitter will depend on the application of the data link. In some of these applications, there may only be a limited amount of power. In these cases, an effort would have to be made to reduce the amount of power consumed by the sensor and transmitter. The circuitry could be redesigned using lower power technologies such as CMOS, but the largest difference

could be made by reducing the power consumption of the transmitter, as the LED alone consumes 1/2 watt. Most of the light generated by the LED is not coupled into the fiberoptic cable. A better arrangement for coupling the LED's light into the fiberoptic cable would reduce the total amount of light that had to be generated, with a corresponding reduction in power consumption.

Another area that has not been investigated is the temperature dependence of the system. The system response will vary somewhat with temperature variations, but the magnitude of these variations was not checked in the prototype. These variations would have to be quantified before an instrument could operate dependably over a wide range of temperature.

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Analog Devices, Data Acquisition Products Catalog Supplement, Route 1, Industrial Park, P.O. Box 280, Norwood, Massachusetts 02062, 1979

2. Brokaw, Paul, An I.C. Amplifiers User's Guide to Decoupling, Grounding, and Making Things Go Right For a Change, Analog Devices, Route 1, Industrial Park, P.O. Box 280, Norwood, Massachusetts 02062

3. Gerald, Curtis F., Applied Numerical Analysis, Addison-Wesley Publishing Company, Reading, Massachusetts, 1970

4. Hall, E.H., American Journal of Mathematics, 2, 287 (1879)

5. Intel Corporation, Component Data Catalog, 3065 Bowers Avenue, Santa Clara, California 95051, 1979

6. Intel Corporation, ISIS II User's Guide, 3065 Bowers Avenue, Santa Clara, California 95051, 1978

7. National Semiconductor Corporation, AN 31, Linear Applications Handbook, 2900 Semiconductor Drive, Santa Clara, California 95051, 1978

8. National Semiconductor Corporation, AN 31, Linear Integrated Circuits Catalog, 2900 Semiconductor Drive, Santa Clara, California 95051, 1977

9. Ohio Semitronics, Incorporated, Product Catalog, 1205 Chesapeake Avenue, Columbus, Ohio 43212

10. Smith, Lois K., 8080 Software Development Package (unpublished), Department of Electrical Engineering, Montana State University, Bozeman, Montana 59717, 1978

11. Sze, S.M., Physics of Semiconductor Devices, John Wiley and Sons, New York, New York, 1969

12. Uradnisheck, Jay, Estimating When Fiberoptics Will Offer Greater 'Value in Use', Electronics, November 9, 1978

13. Western Telecomputing Corporation, WTC-700 System Hardware Manual, Bozeman, Montana 59715

14. Western Telecomputing Corporation, WTC-800 System Hardware Manual, Bozeman, Montana 59715

15. Westinghouse Electric Corporation, Type CMS Current Transformers Data Sheet, Meter and LVIT Division, Raleigh, North Carolina 27603, 1978

APPENDIX

APPENDIX A

ANALYSIS OF FILTER/RECTIFIER STAGE

ANALYSIS OF FILTER/RECTIFIER STAGE

The circuit in Figure 1A was taken from a National Semiconductor applications note (7) and provides the rectification and filtering action in the rectifier and filter. This is not actually the circuit that was used in the hardware, as it has had all of the frequency compensating elements and extra amplification elements removed to clarify circuit operation. Amplifier A1 and the feedback network consisting of R1, R2, D1, D2 and C4 provide the rectification. To analyze the circuit operation, the following assumptions need to be made:

(1) Capacitor C4 is small enough to have little effect in the circuit's passband.

(2) Resistors R4 and R5 are small compared to the input resistance of the operational amplifiers (approximately 400K ohms). Hence, they are going to have little effect at raising the noninverting inputs off ground, and can be neglected in the analysis.

When these two considerations have been made, it is possible to show that the voltage transfer function in the complex frequency plane (s plane) for the circuit in Figure 1 has the form:

$$\frac{Vout}{Vin} = \frac{R7}{(C2 \times R7 \times s + 1)} \frac{(R1 \times R3 - R6 \times R2)}{(R6 \times R3 \times R1)} \qquad \text{eq(A1.1)}$$

for Vin positive and

$$\frac{Vout}{Vin} = \frac{-R7}{(C2 \times R7 \times s + 1) \times R6} \qquad \text{eq(A1.2)}$$

for Vin negative. If R1 = R2 = R6 = R and if R3 = 1/2 R then eq(A1.1)

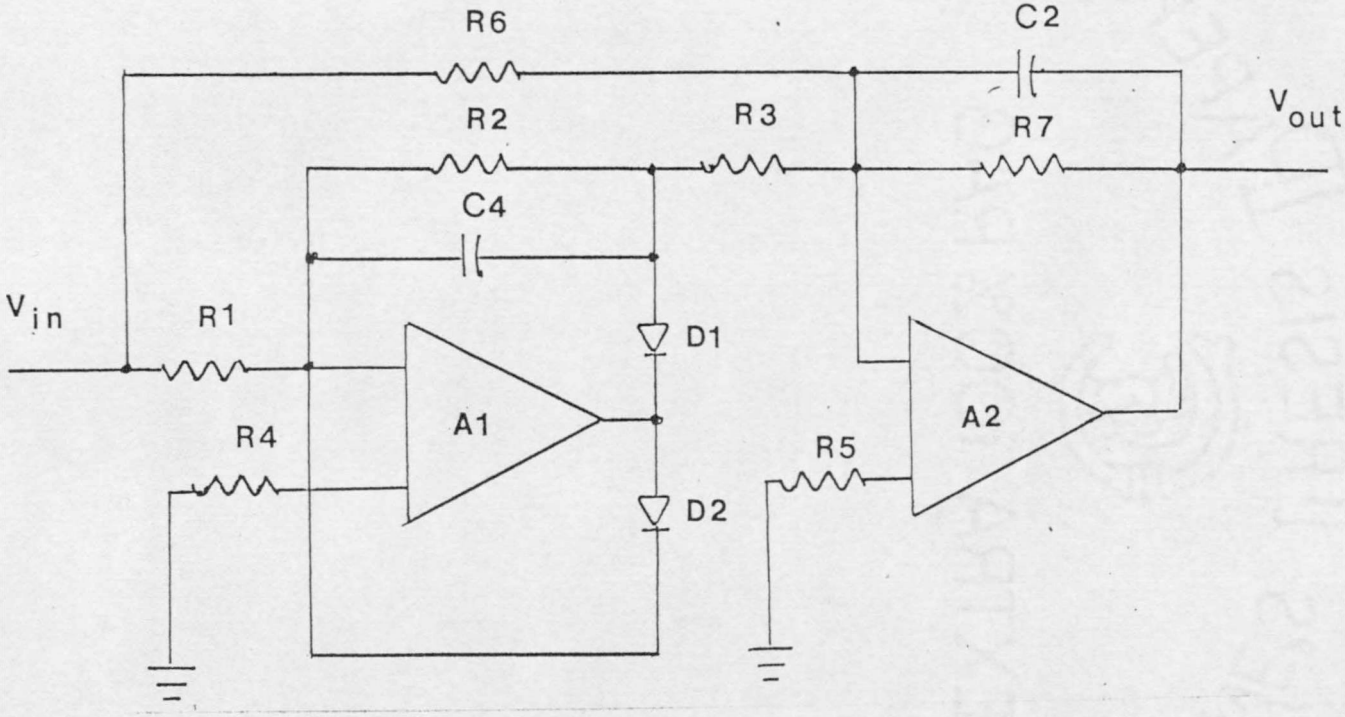


Figure 1A Rectifier/Filter

reduces to:

$$\frac{Vout}{Vin} = \frac{R7}{(C2 \times R7 \times s + 1) \times R} \qquad \text{eq(A1.3)}$$

and eq(A1.2) reduces to the negative of eq(A1.3). The signal inversion caused by this minus sign for negative input voltages causes negative voltages to become positive. The diode combination also blocks the output of the first amplifier for negative input voltages. These two actions insure that the rectification is fullwave as was anticipated.

The (C2 x R7 x s + 1) part in the denominator of eq(A1.3) exhibits a pole. If the vaules of the resistances and capacitor are substituted from the schematic in the main part of the text, the pole occurs at:

$$wp = 1/(C2xR7) = 1/((22.2 \times 10^{3})(10 \times 10^{-6})) =$$

$$4.5 \text{ rad/sec or } .717 \text{ Hz.} \qquad \text{eq(A1.4)}$$

Above this frequency the circuit tends to smooth the output voltage and below this frequency the signals are passed unchanged. This smoothing operation is akin to an integration and averaging over the period of the wave form. The pole at a finite frequency causes D.C. to be passed without this "integration". This is convenient, since if the D.C. were integrated, a ramp function would be generated which would be interpreted by the voltage-to-frequency converter as a constantly changing input voltage rather than a constant input voltage.

APPENDIX B

INTERFACING PLM/80 AND ASSEMBLY LANGUAGE PROGRAMS

INTERFACING PLM/80 AND ASSEMBLY LANGUAGE PROGRAMS

The main consideration in calling a PLM/80 program from assembly language or vice versa is the problem of parameter passing from one language to another. There is a convention that has been established (6). When a PLM/80 procedure calls a subroutine and parameters are passed to the subroutine, the first parameter is passed in the 8080's BC register pair, and the second parameter is passed in the 8080's DE register pair. If there are more than two parameters to be passed, then the last two are passed as above, with all others being placed on the stack. The first parameter in the CALL statement parameter list is the first on the stack, and the last parameter in the list is in the DE register pair. If the parameter being passed is a byte, then it is passed in the low order register of the register pair (i.e., if the byte is to be passed as the BC register pair, then it is placed in the C register). When addresses are passed to the called routine, the most significant byte of the address is passed in the high order byte of the register pair. When a subroutine is to return a parameter to the calling routine, there are only two cases to consider. The routine either returns a byte or an address. Bytes are returned in the A register, and addresses are returned in the HL register pair.

APPENDIX C

PROGRAM FOR DETERMINING MINIMUM SOFTWARE DELAY

PROGRAM FOR DETERMINING MINIMUM SOFTWARE DELAY

This program is assumed to be entered just after the frequency counter has completed a conversion. The loop that the processor sits in while waiting for the conversion is assumed to be labeled WAIT. This loop is not presented here, but would be very similar to the one in the routine that reads the frequency board. It is also assumed that the register pair HL contains the address of the block in memory that the data is to be stored in and that register C contains the number of times the loop is to be executed.

```
            DSEG
CTR1        EQU     XXH         ;ADDRESS OF THE DATA COUNTER
CTR0        EQU     XXH         ;ADDRESS OF THE TIMING COUNTER
READ1       EQU     XXH         ;COMMAND TO PREPARE COUNTER ONE FOR READING
CMD         EQU     XXH         ;ADDRESS OF THE COUNTER COMMAND PORT
SC1MD4      EQU     XXH         ;SETS COUNTER 1 TO MODE FOUR
SC1MD0      EQU     XXH         ;SETS COUNTER 1 TO MODE ZERO
COMMA       EQU     ','         ;DATA DELIMITER
            ;
            ;
            ;
            CSEG
            ;HL MUST CONTAIN THE CURRENT MEMORY ADDRESS FOR THE DATA
            ;C MUST CONTAIN THE NUMBER OF TIMES FREQUENCY COUNTER IS TO
            ;BE READ
            ;
            ;

            JNZ     WAIT        ;END OF WAIT LOOP
            MVI     A,READ1
            OUT     CMD         ;PREPARE COUNTER 1 FOR READ
            IN      CTR1
            MOV     M,A         ;SAVE LSB RESULT IN MEMORY
            INX     H           ;MOVE MEMORY POINTER
            IN      CTR1
            MOV     M,A         ;SAVE MSB RESULT IN MEMORY
```

```
INX     H
MVI     M,COMMA     ;DATA DELIMITER IN MEMORY
INX     H
DCR     C           ;DECREMENT LOOP COUNT
JZ      DONE        ;EXIT IF THRU
MVI     A,SC1MD4    ;RESET COUNTERS AND FLIPFLOPS
OUT     CMD
MVI     A,SC1MD0
OUT     CMD
MVI     A,SC0MD0
OUT     CMD
MVI     A,99H       ;LOAD MAX COUNT IN COUNTER ONE
OUT     CTR1
OUTCTR1
MVIA,00
OUTCTR0             ;LOAD COUNT PER IN CTR ZERO
MVI     A,82H
OUT     CTR0        ;THIS BEGING CONVERSION
JMP     WAIT        ;WAIT TILL CONVERSION DONE
DONE:
;
;
```

APPENDIX D

THE COORDINATING PROGRAM

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CURRENTMONITORDRI
        -    VER
OBJECT MODULE PLACED IN DVRTST.OBJ
COMPILER INVOKED BY:  :F1:PLM80 DVRTST.PLM NOPAGING
        -    TITLE('DRIVER TESTING PROGRAM')


    1           CURRENT$MONITOR$DRIVER:
                  DO;
    2   1         DECLARE (PROMPT1,PROMPT2,PROMPT3) LABEL PU
        -         BLIC;
                  /*
                   EXTERNAL PROCEDURES DESCRIBED IN OTHER MODULES
                  */
    3   1             KEYBOARD$INPUT: PROCEDURE (BUFR) EXTERNAL;
    4   2                 DECLARE BUFR ADDRESS;
    5   2                 END KEYBOARD$INPUT;
    6   1             LINOUT: PROCEDURE (BUFR) EXTERNAL;
    7   2                 DECLARE BUFR ADDRESS;
    8   2                 END LINOUT;
    9   1             STROUT: PROCEDURE (BUFR) EXTERNAL;
   10   2                 DECLARE BUFR ADDRESS;
   11   2                 END STROUT;
   12   1             CARLNF: PROCEDURE EXTERNAL;
   13   2                 END CARLNF;
   14   1             FREQ:   PROCEDURE (DABUFR,INTPER) EXTERNAL;
   15   2                 DECLARE (DABUFR,INTPER) ADDRESS;
   16   2                 END FREQ;
   17   1             DVRATD: PROCEDURE (BUFF,DEV) EXTERNAL;
   18   2                 DECLARE BUFF ADDRESS, DEV BYTE;
   19   2                 END DVRATD;
                  /*
                   PROCEDURE TO PACK 2 ASCII NUMBERS INTO ONE BC
        -          D BYTE. REQUIRES THE ADDRESS OF MS ASCII BYTE.
                  */
   20   1             PACK:   PROCEDURE (FIRST$BYTE) BYTE PUBLIC;
```

```
PLM80 COMPILER    DRIVER TESTING PROGRAM

  21   2              DECLARE FIRSTBYTE ADDRESS, CHAR BYTE;
  22   2              DECLARE (CONTENT BASED FIRSTBYTE) (1)
         -  BYTE;
  23   2              CHAR = 0H;
  24   2              CHAR = ROR((CONTENT (0) AND 00001111B)
         -  ,4);
  25   2              CHAR = CHAR OR (CONTENT (1) AND 00001
         -  111B);
  26   2              RETURN CHAR;
  27   2              END PACK;
                    /*
                     PROCEDURE TO CALCULATE BINARY VALUE OF FOUR D
                    -  IGIT ASCII STRING.
                    */
                    /*
                      REQUIRES THE ADDRESS OF THE BYTE PRECEEDING
                    -  MS ASCII BYTE
                    */
  28   1            FOURVALUE: PROCEDURE (STRING) ADDRESS PUBL
                    -  IC;
  29   2              DECLARE (STRING,VALUE) ADDRESS;
  30   2              DECLARE (ARRAY BASED STRING) (4) BYTE;
  31   2              VALUE = (ARRAY (1) AND 0FH)*1000 + (AR
                    -  RAY (2) AND 0FH)*100 + (ARRAY (3) AND 0FH)*10
                    -  + (ARRAY (4) AND 0FH);
  32   2              RETURN VALUE;
  33   2            END FOURVALUE;
                    /*
                    SIGN ON AND PROMPTING MESSAGES
                    */
  34   1              DECLARE CONTENT1 (32) BYTE PUBLIC DATA
                    -  (31,'CURRENT MONITOR READING ROUTINE');
  35   1              DECLARE CONTENT2 (48) BYTE PUBLIC DATA
                    -  (47,'ENTER 4 DIGIT INTEGRATION COUNT (DEFAUL
                    -  T 8200)?');
```

PLM80 COMPILER DRIVER TESTING PROGRAM

```
                  36   1          DECLARE CONTENT3 (35) BYT
                  -    E PUBLIC DATA (
                  -    34,'NUMBER OF SAMPLES (0001 TO 9999)? ');
           37     1          DECLARE CONTENT4 (51) BYTE PUBLIC DATA
                  -     (50,'NUMBER OF SECONDS BETWEEN SAMPLES (0000
                  -     TO 9999)? ');
           38     1          DECLARE CONTENT5 (9) BYTE PUBLIC DATA
                  -    (8,' COUNT= ');
           39     1          DECLARE CONTENT6 (16) BYTE PUBLIC DATA
                  -     (15,' D.C. VOLTAGE= ');
           40     1          DECLARE DABUFR(256) BYTE PUBLIC;
           41     1          DECLARE BUFR(256) BYTE PUBLIC /*ALOCAT
                  -    E BUFFER STORAGE*/;
           42     1          DECLARE (COUNT1,TEMPB,K,DEVICE,BUFL) B
                  -    YTE PUBLIC;
           43     1          DECLARE INTPER STRUCTURE(HIGHBYTE BYTE
                  -    ,LOWBYTE BYTE) PUBLIC;
           44     1          DECLARE (I,J,PERIOD,SAMPLENUMBER,SECON'
                  -    DS) ADDRESS PUBLIC;
                             /*
                             CALL DRIVERS AND INTERACT WITH TTY OPERATOR
                             */
                        /*
                        GET INTEGRATION PERIOD FROM OPERATOR
                        */
           45     1          PROMPT1: CALL LINOUT(.CONTENT1(0));
           46     1          CALL STROUT(.CONTENT2(0));
           47     1          CALL KEYBOARD$INPUT(.BUFR(0));
           48     1          IF BUFR(0) = 0 THEN
           49     1               DO;
           50     2                    INTPER.LOWBYTE = 00H;
           51     2                    INTPER.HIGHBYTE = 82H;
           52     2               END;
                             ELSE
           53     1               DO;
```

```
PLM80 COMPILER    DRIVER TESTING PROGRAM

 54   2                        IF BUFR(0) <> 4 THEN GOTO PROM
          -    PT1;
 56   2                        ELSE DO;
 57   3                            INTPER.LOWBYTE = PACK(.BUF
          -    R(3));
 58   3                            INTPER.HIGHBYTE = PACK(.BU
          -    FR(1));
 59   3                        END;
 60   2                    END;
                /*
                 GET NUMBER OF SAMPLES TO BE TAKEN
                */
 61   1            PROMPT2: CALL STROUT(.CONTENT3(0));
 62   1                CALL KEYBOARD$INPUT(.BUFR(0));
 63   1                IF BUFR(0) <> 4 THEN GOTO PROMPT2;
 65   1                ELSE DO;
 66   2                    SAMPLE$NUMBER = FOURVALUE(.BUFR(0)
          -    );
 67   2                    IF SAMPLE$NUMBER > 9999 THEN GOTO
          -    PROMPT2;
 69   2                END;
                /*
                 GET TIME DELAY BETWEEN END OF TTY OUTPUT & NE
          -      XT SAMPLE
                */
 70   1            PROMPT3: CALL STROUT(.CONTENT4(0));
 71   1                CALL KEYBOARD$INPUT(.BUFR(0));
 72   1                IF BUFR(0) <> 4 THEN GOTO PROMPT3;
 74   1                ELSE DO;
 75   2                    SECONDS = FOURVALUE(.BUFR(0));
 76   2                    IF SECONDS > 9999 THEN GOTO PROMPT
          -    3;
 78   2                END;
 79   1                DEVICE=4AH;/* ATO D CARD, CHANNEL 0*/
 80   1                    PERIOD=(SHL(DOUBLE(INTPER.HIGHBYTE
          -    ),8) OR (DOUBLE(INTPER.LOWBYTE)));
 81   1                DO I=1 TO SAMPLENUMBER;
```

```
PLM80 COMPILER     DRIVER TESTING PROGRAM


  82   2                    CALL FREQ(.DABUFR(0),PERIOD);/*GET
                 FREQUENCY COUNT*/
  83   2                    CALL STROUT(.CONTENT5(0));
  84   2                    CALL STROUT(.DABUFR(0));
  85   2                    CALL DVRATD(.DABUFR(0),DEVICE);/*G
                 ET AD CONVERSION*/
  86   2                    CALL STROUT(.CONTENT6(0));
  87   2                    CALL LINOUT(.DABUFR(0));
  88   2                    DO J = 1 TO SECONDS;
  89   3                         DO K = 1 TO 40;
  90   4                              CALL TIME(250);
  91   4                         END;
  92   3                    END;
  93   2               END;
  94   1          GOTO PROMPT1;
  95   1     END CURRENT$MONITOR$DRIVER;

MODULE INFORMATION:

     CODE AREA SIZE.     = 0286H    646D
     VARIABLE AREA SIZE = 0218H    536D
     MAXIMUM STACK SIZE = 0008H      8D
     119 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE CURRENTMONITORDRI
        -    VER
OBJECT MODULE PLACED IN DVRTST.OBJ
COMPILER INVOKED BY: :F1:PLM80 DVRTST.PLM NOPAGING CODE PRIN
        -    T(DVRTST.LLT) TITLE('DRIVER TESTING PROGRAM')
  1          CURRENT$MONITOR$DRIVER:
               DO;
  2  1         DECLARE (PROMPT1,PROMPT2,PROMPT3) LABEL PU
        -     BLIC;
             /*
              EXTERNAL PROCEDURES DESCRIBED IN OTHER MODULE
        -     S
             */
  3  1           KEYBOARD$INPUT: PROCEDURE (BUFR) EXTERNAL;
  4  2             DECLARE BUFR ADDRESS;
  5  2             END KEYBOARD$INPUT;
  6  1           LINOUT: PROCEDURE (BUFR) EXTERNAL;
  7  2             DECLARE BUFR ADDRESS;
  8  2             END LINOUT;
  9  1           STROUT: PROCEDURE (BUFR) EXTERNAL;
 10  2             DECLARE BUFR ADDRESS;
 11  2             END STROUT;
 12  1           CARLNF: PROCEDURE EXTERNAL;
 13  2             END CARLNF;
 14  1           FREQ:   PROCEDURE (DABUFR,INTPER) EXTERNAL
        -     ;
 15  2             DECLARE (DABUFR,INTPER) ADDRESS;
 16  2             END FREQ;
 17  1           DVRATD: PROCEDURE (BUFF,DEV) EXTERNAL;
 18  2             DECLARE BUFF ADDRESS, DEV BYTE;
 19  2             END DVRATD;
             /*
              PROCEDURE TO PACK 2 ASCII NUMBERS INTO ONE BC
        -     D BYTE. REQUIRES THE ADDRESS OF MS ASCII BYTE.
             */
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

  20  1           PACK:  PROCEDURE (FIRST$BYTE) BYTE PUBLIC
          -     ;
                                                    ; STATEMENT
          -       # 20
                          ; PROC  PACK
           0210  210100             LXI     H,FIRSTBYTE+1H
           0213  70                 MOV     M,B
           0214  2B                 DCX     H
           0215  71                 MOV     M,C
  21   2                 DECLARE FIRST$BYTE ADDRESS, CHAR BYTE;
  22   2                 DECLARE (CONTENT BASED FIRSTBYTE) (1)
          -     BYTE;
  23   2                 CHAR = 0H;
                                                    ; STATEMENT
          -       # 23
           0216  210200             LXI     H,CHAR
           0219  3600               MVI     M,0H
  24   2                 CHAR = ROR((CONTENT (0) AND 00001111B)
          -     ,4);
                                                    ; STATEMENT
          -       # 24
           021B  2A0000             LHLD    FIRSTBYTE
           021E  3E0F               MVI     A,0FH
           0220  A6                 ANA     M
           0221  0F                 RRC
           0222  0F                 RRC
           0223  0F                 RRC
           0224  0F                 RRC
           0225  320200             STA     CHAR
  25   2                 CHAR = CHAR OR (CONTENT (1) AND 00001
          -     11B);
                                                    ; STATEMENT
          -       # 25
           0228  2A0000             LHLD    FIRSTBYTE
           022B  23                 INX     H
           022C  3E0F               MVI     A,0FH
           022E  A6                 ANA     M
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

            022F  210200          LXI     H,CHAR
            0232  B6              ORA     M
            0233  77              MOV     M,A
   26   2              RETURN CHAR;
                                                    ; STATEMENT
            -       # 26
            0234  C9              RET
   27   2              END PACK;
                                                    ; STATEMENT
            -       # 27
                  /*
                  - PROCEDURE TO CALCULATE BINARY VALUE OF FOUR D
                  -   IGIT ASCII STRING.
                  */
                  /*
                    REQUIRES THE ADDRESS OF THE BYTE PRECEEDING
                  -  MS ASCII BYTE
                  */
   28   1            FOURVALUE: PROCEDURE (STRING) ADDRESS PUBL
                  -  IC;
                                                    ; STATEMENT
                  -    # 28
                                ; PROC  FOURVALUE
            0235  210400          LXI     H,STRING+1H
            0238  70              MOV     M,B
            0239  2B              DCX     H
            023A  71              MOV     M,C
   29   2              DECLARE (STRING,VALUE) ADDRESS;
   30   2              DECLARE (ARRAY BASED STRING) (4) BYTE;
   31   2               VALUE = (ARRAY (1) AND 0FH)*1000 + (AR
                  -  RAY (2) AND 0FH)*100 + (ARRAY (3) AND 0FH)*10
                  -    + (ARRAY (4) A
                  -
                  -  ND 0FH);
                                                    ; STATEMENT
                  -    # 31
            023B  2A0300          LHLD    STRING
```

PLM80 COMPILER TESTING PROGRAM WITH MNEMONICS

```
023E  23          INX   H
023F  3E0F        MVI   A,0FH
0241  A6          ANA   M
0242  6F          MOV   L,A
0243  2600        MVI   H,0
0245  11E803      LXI   D,3E8H
0248  CD0000      CALL  @P0034
024B  E5          PUSH  H       ; 1
024C  2A0300      LHLD  STRING
024F  23          INX   H
0250  23          INX   H
0251  3E0F        MVI   A,0FH
0253  A6          ANA   M
0254  6F          MOV   L,A
0255  2600        MVI   H,0
0257  116400      LXI   D,64H
025A  CD0000      CALL  @P0034
025D  C1          POP   B       ; 1
025E  09          DAD   B
025F  010300      LXI   B,3H
0262  E5          PUSH  H       ; 1
0263  2A0300      LHLD  STRING
0266  09          DAD   B
0267  3E0F        MVI   A,0FH
0269  A6          ANA   M
026A  6F          MOV   L,A
026B  2600        MVI   H,0
026D  CD0000      CALL  @P0033
0270  C1          POP   B       ; 1
0271  09          DAD   B
0272  010400      LXI   B,4H
0275  E5          PUSH  H       ; 1
0276  2A0300      LHLD  STRING
0279  09          DAD   B
027A  3E0F        MVI   A,0FH
027C  A6          ANA   M
027D  5F          MOV   E,A
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                  027E  1600            MVI     D,0
                  0280  E1              POP     H       ; 1
                  0281  19              DAD     D
                  0282  220500          SHLD    VALUE
  32   2                  RETURN VALUE;
                                                        ; STATEMENT
                -    # 32
                  0285  C9              RET
  33   2           END FOURVALUE;
                                                        ; STATEMENT
                -    # 33
                     /*
                     SIGN ON AND PROMPTING MESSAGES
                     */
  34   1                   DECLARE CONTENT1 (32) BYTE PUBLIC DATA
                -      (31,'CURRENT MONITOR READING ROUTINE');
  35   1                   DECLARE CONTENT2 (48) BYTE PUBLIC DATA
                -      (47,'ENTER 4 DIGIT INTEGRATION COUNT (DEFAUL
                -     T 8200)?');
                -     36   1                DECLARE CONTENT3 (35) BYT
                -     E PUBLIC DATA (
                -     34,'NUMBER OF SAMPLES (0001 TO 9999)? ');
  37   1                   DECLARE CONTENT4 (51) BYTE PUBLIC DATA
                -      (50,'NUMBER OF SECONDS BETWEEN SAMPLES (0000
                -      TO 9999)? ');
  38   1                   DECLARE CONTENT5 (9) BYTE PUBLIC DATA
                -     (8,' COUNT= ');
  39   1                   DECLARE CONTENT6 (16) BYTE PUBLIC DATA
                -      (15,' D.C. VOLTAGE= ');
  40   1                   DECLARE DABUFR(256) BYTE PUBLIC;
  41   1                   DECLARE BUFR(256) BYTE PUBLIC /*ALOCAT
                -     E BUFFER STORAGE*/;
  42   1                   DECLARE (COUNT1,TEMPB,K,DEVICE,BUFL) B
                -     YTE PUBLIC;
  43   1                   DECLARE INTPER STRUCTURE(HIGHBYTE BYTE
                -     ,LOWBYTE BYTE) PUBLIC;
  44   1                   DECLARE (I,J,PERIOD,SAMPLENUMBER,SECON
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS


          -     DS) ADDRESS PUBLIC;
                   /*
                    CALL DRIVERS AND INTERACT WITH TTY OPERATO
          -     R
                   */
                /*
                 GET INTEGRATION PERIOD FROM OPERATOR
                */
   45   1            PROMPT1: CALL LINOUT(.CONTENT1(0));
                                                    ; STATEMENT
          -          # 45
           00BF  310000              LXI     SP,@STACK$ORIGIN
                                PROMPT1:
           00C2  310000              LXI     SP,@STACK$ORIGIN
           00C5  010000              LXI     B,CONTENT1
           00C8  CD0000              CALL    LINOUT
   46   1               CALL STROUT(.CONTENT2(0));
                                                    ; STATEMENT
          -          # 46
           00CB  012000              LXI     B,CONTENT2
           00CE  CD0000              CALL    STROUT
   47   1               CALL KEYBOARD$INPUT(.BUFR(0));
                                                    ; STATEMENT
          -          # 47
           00D1  010701              LXI     B,BUFR
           00D4  CD0000              CALL    KEYBOARDINPUT
   48   1               IF BUFR(0) = 0 THEN
                                                    ; STATEMENT
          -          # 48
           00D7  3A0701              LDA     BUFR
           00DA  FE00                CPI     0H
           00DC  C2EA00              JNZ     @1
   49   1                       DO;
   50   2                           INTPER.LOWBYTE = 00H;
                                                    ; STATEMENT
          -          # 50
           00DF  210D02              LXI     H,INTPER+1H
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                00E2  3600            MVI     M,0H
51    2                           INTPER.HIGHBYTE = 82H;
                                                  ; STATEMENT
              -     # 51
                00E4  2B              DCX     H
                00E5  3682            MVI     M,82H
52    2                     END;
                                                  ; STATEMENT
              -     # 52
                00E7  C30701          JMP     @2
                              @1:
                          ELSE
                              DO;
53    1
54    2                         IF BUFR(0) <> 4 THEN GOTO PROM
              -     PT1;
                                                  ; STATEMENT
              -     # 54
                00EA  3A0701          LDA     BUFR
                00ED  FE04            CPI     4H
                00EF  CAF500          JZ      @3
                                                  ; STATEMENT
              -     # 55
                00F2  C3C200          JMP     PROMPT1
                              @3:
56    2                         ELSE DO;
57    3                           INTPER.LOWBYTE = PACK(.BUF
              -     R(3));
                                                  ; STATEMENT
              -     # 57
                00F5  010A01          LXI     B,BUFR+3H
                00F8  CD1002          CALL    PACK
                00FB  320D02          STA     INTPER+1H
58    3                           INTPER.HIGHBYTE = PACKH.BU
              -     FR(1));
                                                  ; STATEMENT
              -     # 58
                00FE  010801          LXI     B,BUFR+1H
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                0101  CD1002            CALL    PACK
                0104  320C02            STA     INTPER
 59   3                          END;
                                                    ; STATEMENT
              -     # 59
                              @4:
 60   2                         END;
                                                    ; STATEMENT
              -     # 60
                              @2:
                    /*
                     GET NUMBER OF SAMPLES TO BE TAKEN
                    */
 61   1                  PROMPT2: CALL STROUT(.CONTENT3(0));
                                                    ; STATEMENT
              -     # 61
                              PROMPT2:
                0107  310000            LXI     SP,@STACK$ORIGIN
                010A  015000            LXI     B,CONTENT3
                010D  CD0000            CALL    STROUT
 62   1                       CALL KEYBOARD$INPUT(.BUFR(0));
                                                    ; STATEMENT
              -     # 62
                0110  010701            LXI     B,BUFR
                0113  CD0000            CALL    KEYBOARDINPUT
 63   1                       IF BUFR(0) <> 4 THEN GOTO PROMPT2;
                                                    ; STATEMENT
              -     # 63
                0116  3A0701            LDA     BUFR
                0119  FE04              CPI     4H
                011B  CA2101            JZ      @5
                                                    ; STATEMENT
              -     # 64
                011E  C30701            JMP     PROMPT2
                              @5:
 65   1                       ELSE DO;
 66   2                          SAMPLE$NUMBER = FOURVALUE(.BUFR(0)
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

          -     );
                                                    ; STATEMENT
          -       # 66
           0121  010701            LXI     B,BUFR
           0124  CD3502            CALL    FOURVALUE
           0127  221402            SHLD    SAMPLENUMBER
 67   2                    IF SAMPLE$NUMBER > 9999 THEN GOTO
          -     PROMPT2;
                                                    ; STATEMENT
          -       # 67
           012A  110F27            LXI     D,270FH
           012D  211402            LXI     H,SAMPLENUMBER
            0130  CD0000            CALL    @P0104
           0133  D23901            JNC     @7
                                                    ; STATEMENT
          -       # 68
           0136  C30701            JMP     PROMPT2
                              @7:
 69   2                  END;
                                                    ; STATEMENT
          -       # 69
                              @6:
                /*
                 GET TIME DELAY BETWEEN END OF TTY OUTPUT & NE
          -      XT SAMPLE
                */
 70   1              PROMPT3: CALL STROUT(.CONTENT4(0));
                                                    ; STATEMENT
          -       # 70
                            PROMPT3:
           0139  310000            LXI     SP,@STACK$ORIGIN
           013C  017300            LXI     B,CONTENT4
            013F  CD0000            CALL    STROUT
 71   1                    CALL KEYBOARD$INPUT(.BUFR(0));
                                                    ; STATEMENT
          -       # 71
           0142  010701            LXI     B,BUFR
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                0145  CD0000            CALL    KEYBOARDINPUT
 72   1                    IF BUFR(0) <> 4 THEN GOTO PROMPT3;
                                                        ; STATEMENT
                -     # 72
                0148  3A0701            LDA     BUFR
                014B  FE04              CPI     4H
                014D  CA5301            JZ      @8
                                                        ; STATEMENT
                -     # 73
                0150  C33901            JMP     PROMPT3
                                  @8:
 74   1                    ELSE DO;
 75   2                      SECONDS = FOURVALUE(.BUFR(0));
                                                        ; STATEMENT
                -     # 75
                0153  010701            LXI     B,BUFR
                0156  CD3502            CALL    FOURVALUE
                0159  221602            SHLD    SECONDS
 76   2                      IF SECONDS > 9999 THEN GOTO PROMPT
                -       3;
                                                        ; STATEMENT
                -     # 76
                015C  110F27            LXI     D,270FH
                015F  211602            LXI     H,SECONDS
                0162  CD0000            CALL    @P0104
                0165  D26B01            JNC     @10
                                                        ; STATEMENT
                -     # 77
                0168  C33901            JMP     PROMPT3
                                  @10:
 78   2                    END;
                                                        ; STATEMENT
                -     # 78
                                  @9:
 79   1                    DEVICE=4AH;/* ATO D CARD, CHANNEL 0*/
                                                        ; STATEMENT
                -     # 79
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                016B  210A02          LXI     H,DEVICE
                016E  364A            MVI     M,4AH
80    1                         PERIOD=(SHL(DOUBLE(INTPER.HIGHBYTE
                -     ),8) OR (DOUBLE(INTPER.LOWBYTE)));
                                                        ; STATEMENT
                -       # 80
                0170  2A0C02          LHLD    INTPER
                0173  2600            MVI     H,0
                0175  0E08            MVI     C,8H
                0177  CD0000          CALL    @P0088
                 017A  E5              PUSH    H       ; 1
                017B  2A0D02          LHLD    INTPER+1H
                017E  2600            MVI     H,0
                0180  D1              POP     D       ; 1
                0181  CD0000          CALL    @P0049
                0184  221202          SHLD    PERIOD
81    1                   DO I=1 TO SAMPLENUMBER;
                                                        ; STATEMENT
                -       # 81
                0187  210100          LXI     H,1H
                018A  220E02          SHLD    I
                                @11:
                018D  111402          LXI     D,SAMPLENUMBER
                0190  010E02          LXI     B,I
                0193  CD0000          CALL    @P0098
                 0196  DA0B02          JC      @12
82    2                         CALL FREQ(.DABUFR(0),PERIOD);/*GET
                -       FREQUENCY COUNT*/
                                                        ; STATEMENT
                -       # 82
                0199  2A1202          LHLD    PERIOD
                019C  EB              XCHG
                019D  010700          LXI     B,DABUFR
                 01A0  CD0000          CALL    FREQ
83    2                         CALL STROUT(.CONTENT5(0));
                                                        ; STATEMENT
                -       # 83
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS

                01A3  01A600          LXI     B,CONTENT5
                01A6  CD0000          CALL    STROUT
 84    2                      CALL STROUT(.DABUFR(0));
                                                  ; STATEMENT
               -     # 84
                01A9  010700          LXI     B,DABUFR
                01AC  CD0000          CALL    STROUT
 85    2                      CALL DVRATD(.DABUFR(0),DEVICE);/*G
               -    ET AD CONVERSION*/
                                                  ; STATEMENT
               -     # 85
                01AF  2A0A02          LHLD    DEVICE
                01B2  EB              XCHG
                01B3  010700          LXI     B,DABUFR
                01B6  CD0000          CALL    DVRATD
 86    2                      CALL STROUT(.CONTENT6(0));
                                                  ; STATEMENT
               -     # 86
                01B9  01AF00          LXI     B,CONTENT6
                01BC  CD0000          CALL    STROUT
 87    2                      CALL LINOUT(.DABUFR(0));
                                                  ; STATEMENT
               -     # 87
                01BF  010700          LXI     B,DABUFR
                01C2  CD0000          CALL    LINOUT
 88    2                      DO J = 1 TO SECONDS;
                                                  ; STATEMENT
               -     # 88
                01C5  210100          LXI     H,1H
                01C8  221002          SHLD    J
                                @13:
                01CB  111602          LXI     D,SECONDS
                01CE  011002          LXI     B,J
                01D1  CD0000          CALL    @P0098
                01D4  DAFE01          JC      @14
 89    3                        DO K = 1 TO 40;
                                                  ; STATEMENT
```

PLM80 COMPILER TESTING PROGRAM WITH MNEMONICS

```
                -     # 89
                01D7  210902          LXI     H,K
                01DA  3601            MVI     M,1H
                                @15:
                01DC  3E28            MVI     A,28H
                01DE  210902          LXI     H,K
                01E1  BE              CMP     M
                01E2  DAF101          JC      @16
90    4                                CALL TIME(250);
                                                      ; STATEMENT
                -     # 90
                01E5  3EFA            MVI     A,0FAH
                01E7  CD0000          CALL    @P0105
91    4                             END;
                                                      ; STATEMENT
                -     # 91
                01EA  210902          LXI     H,K
                01ED  34              INR     M
                01EE  C2DC01          JNZ     @15
                                @16:
92    3                              END;
                                                      ; STATEMENT
                -     # 92
                01F1  110100          LXI     D,1H
                01F4  2A1002          LHLD    J
                01F7  19              DAD     D
                01F8  221002          SHLD    J
                01FB  D2CB01          JNC     @13
                                @14:
93    2                       END;
                                                      ; STATEMENT
                -     # 93
                01FE  110100          LXI     D,1H
                0201  2A0E02          LHLD    I
                0204  19              DAD     D
                0205  220E02          SHLD    I
                0208  D28D01          JNC     @11
```

```
PLM80 COMPILER    TESTING PROGRAM WITH MNEMONICS


                        @12:
  94   1                GOTO PROMPT1;
                                                        ; STATEMENT
                  # 94
              020B  C3C200          JMP     PROMPT1
  95   1          END CURRENT$MONITOR$DRIVER;
                                                        ; STATEMENT
                  # 95
              020E  FB              EI
              020F  76              HLT

MODULE INFORMATION:

     CODE AREA SIZE     = 0286H    646D
     VARIABLE AREA SIZE = 0218H    536D
     MAXIMUM STACK SIZE = 0008H      8D
     119 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

APPENDIX E

THE TELETYPE DRIVERS

PARAMETER PASSING SPECIFICATIONS FOR THE TELETYPE DRIVERS

The following routines are written in assembly language and are contained in a file named DVRSUP.SRC:

STROUT: Outputs the contents of a buffer without appending a carriage return and line feed to the buffer. The address of the first byte in the buffer must be placed in register pair BC prior to calling this routine.

LINOUT: Outputs the buffer appending a carriage return and line feed to the buffer contents. The address of the first byte in the buffer must be placed in BC prior to the routine call.

CHIN: Reads a character from the teletype. The character is returned to the calling routine in the A register.

CHOUT: Outputs a character to the teletype. The address of the character to be output must be in the BC register pair prior to the routine call.

The following routines are written in PLM/80 and are contained in the file DVRPTY.PLM:

ECHO: Receives a character from the teletype keyboard and prints the character on the teletype. Returns ASCII equivalent of the character in the A register.

CHARACTEROUT: Prints a character on the teletype. The routine must be entered with the ASCII equivalent of the character in the C register.

CHARACTERINPUT: Gets a character from the teletype keyboard. Character is returned to the calling routine in the A register.

CARLNF: Outputs a carriage return and a line feed to the teletype.

KEYBOARDINPUT: Fills a buffer in memory with characters from the teletype. The routine is exited with a carriage return which is not placed in the buffer. The routine ignores linefeeds as input characters. Procedures are provided to allow the previously entered character to be deleted from the buffer using the rubout key or to discard all of the characters in the buffer by entering a CONTROL-X. When the buffer is full, the routine will not accept new characters until a carriage return is used. A carriage return at any time will cause the routine to be exited with the buffer conforming to the standard stated earlier.

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TELETYPEDRIVER
OBJECT MODULE PLACED IN DVRPTY.OBJ
COMPILER INVOKED BY:  :F1:PLM80 DVRPTY.PLM NOPAGING TITLE('D
          -    RIVER TESTING PROGRAM')


  1          TELETYPEDRIVER:
               DO;
             /*
              DATA INITIALIZATION FOR CONSTANTS
              */
  2   1          DECLARE (CRET,FEED,RESET,SET,RUBOUT,DASH,C
          -     NTRLX,SLASH) BYTE PUBLIC DATA(0DH,0AH,40H,0CF
          -     H,7FH,5FH,18H,5CH);
             /*
             PROCEDURE TO RESET 8251
             */
  3   1          RETMD:  PROCEDURE PUBLIC;
  4   2              OUTPUT(0EDH)=RESET;
  5   2              RETURN;
  6   2              END RETMD;
             /*
             PROCEDURE TO SET 8251 MODE (DO FIRST)
             */
  7   1          SETMD:  PROCEDURE PUBLIC;
  8   2              OUTPUT(0EDH)=SET;
  9   2              RETURN;
 10   2              END SETMD;
             /*
             GET CHARACTER FROM 8251 OR USE ANOTHER ROUTINE
          -    WITH SAME NAME
             */
 11   1          CHIN:   PROCEDURE BYTE EXTERNAL;
 12   2              END CHIN;
             /*
             OUTPUT CHAR TO 8251 GIVEN ITS ADDRESS IN REG B
```

```
PLM80 COMPILER    TELETYPE DRIVERS

                  C
                */
  13    1       CHOUT: PROCEDURE(CHAROUT) EXTERNAL;
  14    2           DECLARE CHAROUT ADDRESS;
  15    2           END CHOUT;
                /*
                 ROUTINE TO GET CHAR FROM TTY KEYBOARD AND ECH
                 O IT TO TTY PRINTER
                */
  16    1       ECHO:  PROCEDURE BYTE PUBLIC;
  17    2           DECLARE CHAR BYTE;
  18    2           CHAR = CHIN;
  19    2           CALL CHOUT(.CHAR);
  20    2           RETURN CHAR;
  21    2           END ECHO;
                /*
                ROUTINE TO OUTPUT CHARACER GIVEN ONLY THE CHAR
                 ACTER
                */
  22    1       CHARACTER$OUT: PROCEDURE (CHAROUTPUT) PUBL
                 IC;
  23    2           DECLARE CHAROUTPUT BYTE;
  24    2           CALL CHOUT (.CHAROUTPUT);
  25    2           RETURN;
  26    2       END CHARACTER$OUT;
                /*
                 ROUTINE TO CHARACTER INPUT RETURNS CHARACTER
                 RECEIVED FROM TTY
                */
  27    1       CHARACTERINPUT: PROCEDURE BYTE PUBLIC;
  28    2           DECLARE TEMP BYTE;
  29    2           TEMP =  CHIN;
  30    2           RETURN TEMP;
  31    2           END CHARACTERINPUT;
                /*
                 GENERATES A CARRIAGE RETURN AND LINE FEED
                */
```

```
PLM80 COMPILER    TELETYPE DRIVERS

  32   1          CARLNF: PROCEDURE PUBLIC;
                  /*OUTPUT CARRIAGE RETURN*/
  33   2              CALL CHARACTER$OUT (CRET);
                      /*OUTPUT LINEFEED*/
  34   2              CALL CHARACTER$OUT (FEED);
  35   2              RETURN;
  36   2              END CARLNF;
                /*
                PRINT CONTENTS OF BUFFER TO OUTPUT DEVICE. RE
              -  QUIRES ADDRESS OF BUFFER
                */
  37   1          LINOUT: PROCEDURE (BUFR) EXTERNAL;
  38   2              DECLARE BUFR ADDRESS;
  39   2              END LINOUT;
                /*
                PRINT BUFFER CONTENTS WITHOUT CARRIAGE RETURN.
              -    REQUIRES BUFFER ADDRESS
                */
  40   1          STROUT: PROCEDURE (BUFR) EXTERNAL;
  41   2              DECLARE BUFR ADDRESS;
  42   2              END STR$OUT;
                /*
                FILL BUFFER FROM KEYBOARD. REQUIRES BUFFER AD
              -  DRESS. MAX 255 CHARACTERS
                */
  43   1          KEYBOARDINPUT: PROCEDURE (BUFR) REENTRANT
              -  PUBLIC;
  44   2              DECLARE BUFR ADDRESS;
  45   2              DECLARE (BUFL,COUNT,CHARINPUT) BYTE;
  46   2              DECLARE (CONTENT BASED BUFR) (255) BYT
              -  E;
  47   2              BUFL=255;
  48   2              LOOP0: COUNT=0;
  49   2              LOOP1: CHARINPUT = ECHO;
  50   2                  IF CHARINPUT = CRET  THEN GOTO EXI
              -  T;
                          /*EXIT IF CARRIAGE RETURN*/
```

```
PLM80 COMPILER    TELETYPE DRIVERS

 52   2                    IF CHARINPUT = FEED  THEN GOTO LOO
        -    P1;
                           /*LOOP BACK IF LINEFEED*/
 54   2                    IF CHARINPUT= RUBOUT  THEN
                           /*RUBOUT ROUTINE*/
 55   2                    RUB:    DO;
 56   3                        IF COUNT > 0 THEN COUNT = COUN
        -    T -1;
 58   3                        CALL CHARACTER$OUT (DASH);
 59   3                        GOTO LOOP1;
 60   3                        END RUB;
 61   2                    IF CHARINPUT = CNTRLX THEN
                           /*WIPEOUT ROUTINE, ENTERED BY CONT
        -    ROL X*/
 62   2                    NEW:    DO;
                               /*OUTPUT SLASH*/
 63   3                        CALL CHARACTER$OUT (SLASH);
 64   3                        CALL CARLNF;
 65   3                        GOTO LOOP0;
 66   3                        END NEW;
 67   2                    CONTENT(1 + COUNT) = CHARINPUT;
 68   2                    COUNT = COUNT + 1;
 69   2                    IF COUNT > BUFL-1 THEN GOTO WAIT;
 71   2                IF CHARINPUT = CRET THEN GOTO EXIT;
                          /*EXIT ON CARRIAGE RETURN*/
 73   2                ELSE GOTO LOOP1;
 74   2                WAIT:    CHARINPUT = CHIN;
                               /*WAIT FOR CARRIAGE RETURN*/
 75   2                        IF CHARINPUT = CRET THEN GOTO EXIT
        -    ;
 77   2                        ELSE GOTO WAIT;
 78   2                EXIT:   CONTENT(0) = COUNT;
 79   2                        CALL CARLNF;
 80   2                RETURN;
 81   2                END KEYBOARDINPUT;
 82   1            END TELETYPEDRIVER;
```

```
PLM80 COMPILER - TELETYPE DRIVERS


MODULE INFORMATION:

     CODE AREA SIZE      = 0118H    280D
     VARIABLE AREA SIZE  = 0003H      3D
     MAXIMUM STACK SIZE  = 0008H     11D
     122 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
A SM80 DVRSUP.SRC SYMBOLS NOPAGING TITLE('TELETYPE DRIVER SU
                  PPORT')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          MODULE
TELETYPE DRIVER SUPPORT

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1          DSEG
  00ED                 2 TTYC     EQU     0EDH     ;TTY COMMAND
                         CHANNEL
  00EC                 3 TTYD     EQU     0ECH     ;TTY DATA CH
                         ANNEL
                       4 ;
                       5 ;
                       6          CSEG
                       7 ;
                       8 ;-----------------------------------
                         -------
                       9 ; ROUTINE TO OUTPUT A STRING. STRING
                          ADDRESS IN BC. FIRST BYTE OF STRING
                      10 ; IS THE NUMBER OF CHAR IN REST OF S
                         TRING.
                      11 ; USES REG HL,DE,BC,AF.  HL,DE RETUR
                         NED INTACT.
                      12 ;-----------------------------------
                         -------
                      13 ;
  0000 D5             14 STROUT: PUSH    D
  0001 E5             15         PUSH    H
  0002 60             16         MOV     H,B
  0003 69             17         MOV     L,C
  0004 5E             18         MOV     E,M      ;PUT CAHR CO
                         UNT IN E
  0005 7B             19         MOV     A,E      ;CHAR COUNT
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER TELETYPE DRIVER SUPPORT

                        IN A
0006 FE00              20       CPI     0H      ;SEE IF NO C
                        HAR
0008 CA1900    C       21       JZ      SCR2    ;JUMP IF NO
                        CHAR
000B 03                22 SCR1: INX     B       ;INC BUF PTR
                       23 ;
                       24 ;-------------------------------------
                        -----
                       25 ; ENTER THE CHARACTER OUT ROUTINE WI
                        TH THE BUFFERS SET FOR PLM CONVENTIO
                        N
                       26 ; BC CONTAINS THE ADDRESS OF THE BYT
                        E TO BE OUTPUT. E CONTAINS THE NUMBE
                        R OF
                       27 ; BYTES REMAINING, BUT IS NOT
                        PASSED TO THE ROUTINE CH
                        OUT.
                       28 ;-------------------------------------
                        -----
                       29 ;
000C CD2F00    C       30       CALL    CHOUT   ;OUTPUT CHAR
                        ACTER
000F 1D                31       DCR     E       ;DEC CHAR CO
                        UNT.
0010 CA1900    C       32       JZ      SCR2    ;OUT LOOP
0013 FA1900    C       33       JM      SCR2    ;EXIT IF MIN
                        US # OF CHAR
0016 C30B00    C       34       JMP     SCR1    ;GET ANOTHER
                        CHAR
0019 E1                35 SCR2: POP     H
001A D1                36       POP     D
001B C9                37       RET
                       38 ;
                       39 ;-------------------------------------
                       40 ;ROUTINE TO OUTPUT A STRING AND SUPP
                        LY CARRIAGE RETURN AND LINE FEED
```

```
                        41 ;NO REGISTERS USED
                        42 ;---------------------------------
                        43 ;
001C CD0000   C         44 LINOUT: CALL    STROUT
001F CD0000   E         45         CALL    CARLNF
0022 C9                 46         RET
                        47 ;
                        48 ;---------------------------------
                        49 ; ROUTINE TO READ A CHARACTER FROM T
                           TY.  RETURNS BYTE IN A.  NO OTHER
                        50 ; REGISTERS USED. ALL OTHERS RETURNE
                           D INTACT.
                        51 ;---------------------------------
                        52 ;
0023 DBED               53 CHIN:   IN      TTYC    ;READ STATUS
                            WORD
0025 E602               54         ANI     2H      ;MASK & CHEC
                           K STATUS WORD
0027 CA2300   C         55         JZ      CHIN    ;WAIT IF NO
                           DATA
002A DBEC               56         IN      TTYD    ;INPUT DATA
                           WORD
002C E67F               57         ANI     07FH    ;REMOVE PARI
                           TY BIT
002E C9                 58         RET
                        59 ;
                        60 ;---------------------------------
                        61 ; ROUTINE TO OUTPUT CHARACTER.  ADDR
                           ESS OF CHARACTER IN BC.  USES AF,BC,H
                           L.
                        62 ; HL,DE RETURNED INTACT.
                        63 ;---------------------------------
                        64 ;
002F E5                 65 CHOUT:  PUSH    H
0030 60                 66         MOV     H,B
0031 69                 67         MOV     L,C     ;PUT CHARACT
                           ER ADDRESS IN HL
```

```
ISIS-II 8080/8085 MACRO ASSEMBLER TELETYPE DRIVER SUPPORT

0032 DBED          68 WAITLP: IN      TTYC    ;READ STATUS
                      WORD
0034 E601          69         ANI     01H     ;MASK & CHEC
                   K STATUS WORD
0036 CA3200    C   70         JZ      WAITLP  ;WAIT FOR TT
                   Y TO CATCH UP IF ZERO
0039 7E            71         MOV     A,M     ;WHEN TTY RE
                   ADY, PUT CHAR IN A
003A D3EC          72         OUT     TTYD    ;OUTPUT CHAR
003C E1            73         POP     H
003D C9            74         RET
                   75 ;
                   76 ;
                   77         EXTRN   CARLNF
                   78         PUBLIC  CHIN,TTYC,TTYD,CHOUT
                   ,LINOUT,STROUT
                   79         END


PUBLIC SYMBOLS
CHIN   C 0023    CHOUT  C 002F    LINOUT C 001C    STROUT C
                        0000    TTYC   A 00ED    TTYD   A 00
                        EC

EXTERNAL SYMBOLS
CARLNF E 0000

USER SYMBOLS
CARLNF E 0000    CHIN   C 0023    CHOUT  C 002F    LINOUT C
                        001C    SCR1   C 000B    SCR2   C 00
                        19     STROUT C 0000

                        TTYC   A 00ED    TTYD   A 00EC    W
                        ITLP C 0032

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE TELETYPEDRIVER
OBJECT MODULE PLACED IN DVRPTY.OBJ
COMPILER INVOKED BY:  :F1:PLM80 DVRPTY.PLM NOPAGING CODE PRI
        -    NT(DVRPTY.LLT) TITLE('TELETYPE DRIVER')


  1          TELETYPEDRIVER:
               DO;
             /*
              DATA INITIALIZATION FOR CONSTANTS
             */
  2   1          DECLARE (CRET,FEED,RESET,SET,RUBOUT,DASH,C
             -    NTRLX,SLASH) BYTE PUBLIC DATA(0DH,0AH,40H,0CF
             -    H,7FH,5FH,18H,5CH);
             /*
             PROCEDURE TO RESET 8251
             */
  3   1          RETMD:  PROCEDURE PUBLIC;
                                                    ; STATEMENT
             -       # 3
                           ; PROC  RETMD
  4   2                  OUTPUT(0EDH)=RESET;
                                                    ; STATEMENT
             -       # 4
             0008  3A0200              LDA     RESET
             000B  D3ED                OUT     0EDH
  5   2                  RETURN;
                                                    ; STATEMENT
             -       # 5
             000D  C9                  RET
  6   2                  END RETMD;
                                                    ; STATEMENT
             -       # 6
             /*
             PROCEDURE TO SET 8251 MODE (DO FIRST)
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS -

                  */
   7   1          SETMD:  PROCEDURE PUBLIC;
                                                    ; STATEMENT
                - # 7
                  ) ; PROC  SETMD
   8   2              OUTPUT(0EDH)=SET;
                                                    ; STATEMENT
                - # 8
                0000E  3A0300          LDA     SET
                00011  D3ED            OUT     0EDH
   9   2              RETURN;
                                                    ; STATEMENT
                - # 9
                00013  C9              RET
  10   2          END SETMD;
                                                    ; STATEMENT
                - # 10
                /*
                GET CHARACTER FROM 8251 OR U
                - SE ANOTHER ROUTINE WITH SAME NAME
                */
  11   1          CHIN:   PROCEDURE BYTE EXTERNAL;
  12   2              END CHIN;
                /*
                OUTPUT CHAR TO 8251 GIVEN ITS ADDRESS IN REG B
                - C
                */
  13   1          CHOUT:  PROCEDURE(CHAROUT) EXTERNAL;
  14   2              DECLARE CHAROUT ADDRESS;
  15   2              END CHOUT;
                /*
                ROUTINE TO GET CHAR FROM TTY KEYBOARD AND ECH
                - O IT TO TTY PRINTER
                */
  16   1          ECHO:   PROCEDURE BYTE PUBLIC;
                                                    ; STATEMENT
                - # 16
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS


                          ; PROC  ECHO
  17   2                  DECLARE CHAR BYTE;
  18   2                  CHAR = CHIN;
                                                    ; STATEMENT
                -     # 18
                  0014  CD0000          CALL    CHIN
                  0017  320000          STA     CHAR
  19   2                  CALL CHOUT(.CHAR);
                                                    ; STATEMENT
                -     # 19
                  001A  010000          LXI     B,CHAR
                  001D  CD0000          CALL    CHOUT
  20   2                  RETURN CHAR;
                                                    ; STATEMENT
                -     # 20
                  0020  3A0000          LDA     CHAR
                  0023  C9              RET
  21   2                END ECHO;
                                                    ; STATEMENT
                -     # 21
                     /*
                     ROUTINE TO OUTPUT CHARACER GIVEN ONLY THE CHAR
                -     ACTER
                     */
  22   1             CHARACTERSOUT: PROCEDURE (CHAROUTPUT) PUBL
                -    IC;
                                                    ; STATEMENT
                -     # 22
                          ; PROC  CHARACTEROUT
                  0024  210100          LXI     H,CHAROUTPUT
                  0027  71              MOV     M,C
  23   2                DECLARE CHAROUTPUT BYTE;
  24   2                CALL CHOUT (.CHAROUTPUT);
                                                    ; STATEMENT
                -     # 24
                  0028  010100          LXI     B,CHAROUTPUT
                  002B  CD0000          CALL    CHOUT
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

  25   2                RETURN;
                                                        ; STATEMENT
           -      # 25
         002E  C9                  RET
  26   2              END CHARACTERSOUT;
                                                        ; STATEMENT
           -      # 26
                 /*
                  ROUTINE TO CHARACTER INPUT RETURNS CHARACTER
           -      RECEIVED FROM TTY
                 */
  27   1              CHARACTERINPUT: PROCEDURE BYTE PUBLIC;
                                                        ; STATEMENT
           -      # 27
                          ; PROC  CHARACTERINPUT
  28   2                DECLARE TEMP BYTE;
  29   2                TEMP = CHIN;
                                                        ; STATEMENT
           -      # 29
         002F  CD0000              CALL    CHIN
         0032  320200              STA     TEMP
  30   2                RETURN TEMP;
                                                        ; STATEMENT
           -      # 30
         0035  3A0200              LDA     TEMP
         0038  C9                  RET
  31   2              END CHARACTERINPUT;
                                                        ; STATEMENT
           -      # 31
                 /*
                  GENERATES A CARRIAGE RETURN AND LINE FEED
                 */
  32   1              CARLNF: PROCEDURE PUBLIC;
                                                        ; STATEMENT
           -      # 32
                          ; PROC  CARLNF
                  /*OUTPUT CARRIAGE RETURN*/
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

  33   2              CALL CHARACTER$OUT (CRET);
                                                    ; STATEMENT
                  # 33
          0039  2A0000          LHLD    CRET
          003C  4D              MOV     C,L
          003D  CD2400          CALL    CHARACTEROUT
                      /*OUTPUT LINEFEED*/
  34   2              CALL CHARACTER$OUT (FEED);
                                                    ; STATEMENT
                  # 34
          0040  2A0100          LHLD    FEED
          0043  4D              MOV     C,L
          0044  CD2400          CALL    CHARACTEROUT
  35   2              RETURN;
                                                    ; STATEMENT
                  # 35
          0047  C9              RET
  36   2              END CARLNF;
                                                    ; STATEMENT
                  # 36
                /*
                PRINT CONTENTS OF BUFFER TO OUTPUT DEVICE.  RE
                QUIRES ADDRESS OF BUFFER
                */
  37   1            LINOUT: PROCEDURE (BUFR) EXTERNAL;
  38   2              DECLARE BUFR ADDRESS;
  39   2              END LINOUT;
                /*
                PRINT BUFFER CONTENTS WITHOUT CARRIAGE RETURN.
                  REQUIRES BUFFER ADDRESS
                */
  40   1            STROUT: PROCEDURE (BUFR) EXTERNAL;
  41   2              DECLARE BUFR ADDRESS;
  42   2              END STR$OUT;
                /*
                FILL BUFFER FROM KEYBOARD. REQUIRES BUFFER AD
                DRESS. MAX 255 CHARACTERS
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS


          */
43   1        KEYBOARDINPUT: PROCEDURE (BUFR) REENTRANT
           -    PUBLIC;
                                                    ; STATEMENT
           -      # 43
                            ; PROC  KEYBOARDINPUT
                  0048  3B                  DCX     SP
                  0049  E5                  PUSH    H
                  004A  C5                  PUSH    B
44   2                  DECLARE BUFR ADDRESS;
45   2                  DECLARE (BUFL,COUNT,CHARINPUT) BYTE;
46   2                  DECLARE (CONTENT BASED BUFR) (255) BYT
           -    E;
47   2                  BUFL=255;
                                                    ; STATEMENT
           -      # 47
                  004B  210200              LXI     H,2H      ; BUFL
                  004E  39                  DAD     SP
                  004F  36FF                MVI     M,0FFH
48   2                  LOOP0: COUNT=0;
                                                    ; STATEMENT
           -      # 48
                            LOOP0:
                  0051  210300              LXI     H,3H      ; COUNT
                  0054  39                  DAD     SP
                  0055  3600                MVI     M,0H
49   2                  LOOP1: CHARINPUT = ECHO;
                                                    ; STATEMENT
           -      # 49
                            LOOP1:
                  0057  CD1400              CALL    ECHO
                  005A  210400              LXI     H,4H      ; CHARINPUT
           -
                  005D  39                  DAD     SP
                  005E  77                  MOV     M,A
50   2                  IF CHARINPUT = CRET THEN GOTO EXI
           -    T;
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

                                                        ; STATEMENT
                # 50
 005F  3A0000           LDA     CRET
 0062  210400           LXI     H,4H     ; CHARINPUT
 0065  39               DAD     SP
 0066  BE               CMP     M
 0067  C26D00           JNZ     @1
                                                        ; STATEMENT
                # 51
 006A  C30401           JMP     EXIT
                @1:
                 /*EXIT IF CARRIAGE RETURN*/
52  2                    IF CHARINPUT = FEED  THEN GOTO LOO
          P1;
                                                        ; STATEMENT
                # 52
 006D  3A0100           LDA     FEED
 0070  210400           LXI     H,4H     ; CHARINPUT
 0073  39               DAD     SP
 0074  BE               CMP     M
 0075  C27B00           JNZ     @2
                                                        ; STATEMENT
                # 53
 0078  C35700           JMP     LOOP1
                @2:
                 /*LOOP BACK IF LINEFEED*/
54  2                    IF CHARINPUT= RUBOUT  THEN
                                                        ; STATEMENT
                # 54
 007B  3A0400           LDA     RUBOUT
 007E  210400           LXI     H,4H     ; CHARINPUT
 0081  39               DAD     SP
 0082  BE               CMP     M
 0083  C29F00           JNZ     @3
                 /*RUBOUT ROUTINE*/
55  2                    RUB:    DO;
                                                        ; STATEMENT
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

          -      # 55
                              RUB:
   56     3                       IF COUNT > 0 THEN COUNT = COUN
          -      T -1;
                                                          ; STATEMENT
          -      # 56
           0086   3E00                 MVI     A,0H
           0088   210300               LXI     H,3H      ; COUNT
           008B   39                   DAD     SP
           008C   BE                   CMP     M
           008D   D29500               JNC     @4
                                                          ; STATEMENT
          -      # 57
           0090   210300               LXI     H,3H      ; COUNT
           0093   39                   DAD     SP
           0094   35                   DCR     M
                              @4:
   58     3                       CALL CHARACTER$OUT (DASH);
                                                          ; STATEMENT
          -      # 58
           0095   2A0500               LHLD    DASH
           0098   4D                   MOV     C,L
           0099   CD2400               CALL    CHARACTEROUT
   59     3                       GOTO LOOP1;
                                                          ; STATEMENT
          -      # 59
           009C   C35700               JMP     LOOP1
   60     3                       END RUB;
                                                          ; STATEMENT
          -      # 60
                              @3:
   61     2                       IF CHARINPUT = CNTRLX THEN
                                                          ; STATEMENT
          -      # 61
           009F   3A0600               LDA     CNTRLX
           00A2   210400               LXI     H,4H      ; CHARINPUT
          -
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS


          00A5  39              DAD     SP
          00A6  BE              CMP     M
          00A7  C2B700          JNZ     @5
                            /*WIPEOUT ROUTINE, ENTERED BY CONT
          -     ROL X*/
 62   2                   NEW:    DO;
                                                ; STATEMENT
          -     # 62
                            NEW:
                              /*OUTPUT SLASH*/
 63   3                     CALL CHARACTER$OUT (SLASH);
                                                ; STATEMENT
          -     # 63
          00AA  2A0700          LHLD    SLASH
          00AD  4D              MOV     C,L
          00AE  CD2400          CALL    CHARACTEROUT
 64   3                     CALL CARLNF;
                                                ; STATEMENT
          -     # 64
          00B1  CD3900          CALL    CARLNF
 65   3                     GOTO LOOP0;
                                                ; STATEMENT
          -     # 65
          00B4  C35100          JMP     LOOP0
 66   3                   END NEW;
                                                ; STATEMENT
          -     # 66
                            @5:
 67   2                     CONTENT(1 + COUNT) = CHARINPUT;
                                                ; STATEMENT
          -     # 67
          00B7  210300          LXI     H,3H    ; COUNT
          00BA  39              DAD     SP
          00BB  7E              MOV     A,M
          00BC  3C              INR     A
          00BD  4F              MOV     C,A
          00BE  0600            MVI     B,0
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

            00C0  EB              XCHG
            00C1  1B              DCX     D
            00C2  1B              DCX     D
            00C3  1B              DCX     D
            00C4  CD0000          CALL    @P0013
            00C7  13              INX     D
            00C8  13              INX     D
            00C9  13              INX     D
            00CA  1A              LDAX    D
            00CB  77              MOV     M,A
 68   2                    COUNT = COUNT + 1;
                                                    ; STATEMENT
          -     # 68
            00CC  EB              XCHG
            00CD  2B              DCX     H
            00CE  34              INR     M
 69   2                    IF COUNT > BUFL-1 THEN GOTO WAIT;
                                                    ; STATEMENT
          -     # 69
            00CF  2B              DCX     H
            00D0  7E              MOV     A,M
            00D1  3D              DCR     A
            00D2  23              INX     H
            00D3  BE              CMP     M
            00D4  D2DA00          JNC     @6
                                                    ; STATEMENT
          -     # 70
            00D7  C3EB00          JMP     WAIT
                            @6:
 71   2                IF CHARINPUT = CRET THEN GOTO EXIT;
                                                    ; STATEMENT
          -     # 71
            00DA  3A0000          LDA     CRET
            00DD  210400          LXI     H,4H      ; CHARINPUT
            00E0  39              DAD     SP
            00E1  BE              CMP     M
            00E2  C2E800          JNZ     @7
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

                                                        ; STATEMENT
        -     # 72
        00E5  C30401              JMP     EXIT
                          @7:
                        /*EXIT ON CARRIAGE RETURN*/
 73   2                 ELSE GOTO LOOP1;
                                                        ; STATEMENT
        -     # 73
        00E8  C35700              JMP     LOOP1
                          @8:
 74   2               WAIT:  CHARINPUT = CHIN;
                                                        ; STATEMENT
        -     # 74
                        WAIT:
                        /*WAIT FOR CARRIAGE RETURN*/
        00EB  CD0000              CALL    CHIN
        00EE  210400              LXI     H,4H          ; CHARINPUT
        00F1  39                  DAD     SP
        00F2  77                  MOV     M,A
 75   2                 IF CHARINPUT = CRET THEN GOTO EXIT
        -     ;
                                                        ; STATEMENT
        -     # 75
        00F3  3A0000              LDA     CRET
        00F6  210400              LXI     H,4H          ; CHARINPUT
        00F9  39                  DAD     SP
        00FA  BE                  CMP     M
        00FB  C20101              JNZ     @9
                                                        ; STATEMENT
        -     # 76
        00FE  C30401              JMP     EXIT
                          @9:
 77   2                 ELSE GOTO WAIT;
                                                        ; STATEMENT
        -     # 77
        0101  C3EB00              JMP     WAIT
                          @10:
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

 78   2                EXIT:  CONTENT(0) = COUNT;
                                                        ; STATEMENT
          -     # 78
                         EXIT:
          0104  210000              LXI     H,0H        ; BUFR
          0107  39                  DAD     SP
          0108  4E                  MOV     C,M
          0109  23                  INX     H
          010A  46                  MOV     B,M
          010B  23                  INX     H
          010C  23                  INX     H
          010D  7E                  MOV     A,M
          010E  60                  MOV     H,B
          010F  69                  MOV     L,C
          0110  77                  MOV     M,A
 79   2                    CALL CARLNF;
                                                        ; STATEMENT
          -     # 79
          0111  CD3900              CALL    CARLNF
 80   2                RETURN;
                                                        ; STATEMENT
          -     # 80
          0114  33                  INX     SP
          0115  E1                  POP     H
          0116  E1                  POP     H
          0117  C9                  RET
 81   2                END KEYBOARDINPUT;
                                                        ; STATEMENT
          -     # 81
 82   1              END TELETYPEDRIVER;


MODULE INFORMATION:

     CODE AREA SIZE     = 0118H    280D
     VARIABLE AREA SIZE = 0003H      3D
     MAXIMUM STACK SIZE = 000BH     11D
```

```
PLM80 COMPILER    TELETYPE DRIVER WITH ASSEMBLY MNEMONICS

      122 LINES READ
      0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

APPENDIX F

THE FREQUENCY BOARD DRIVER

PARAMETER PASSING SPECIFICATIONS FOR THE FREQUENCY BOARD DRIVER

The routines and labels of the module DVRFRQ.SRC:

FREQ: The label of the entry point for the frequency card reading routine. The routine must be called with the address of the data buffer where the count is to be placed in register pair BC and the counting period in register pair DE. The contents of all other registers are returned intact to the calling program.

TSTMSB: Tests the most significant byte of counter zero to see if it is zero. This routine calls READ0 and destroys the contents of the HL register pair. THIS ROUTINE IS NOT PLM/80 COMPATIBLE.

TSTLSB: Tests least significant byte of counter zero to see if it is zero. This routine calls READ0 and destroys the contents of the HL register pair. THIS ROUTINE IS NOT PLM/80 COMPATIBLE.

GETC1: Label of entry point for portion of program that reads counter one and converts the results to an ASCII string.

READ0: Routine to read counter zero. Returns count in HL register pair. NOT PLM/80 COMPATIBLE.

READ1: Routine to read counter one. Returns count in HL register pair. NOT PLM/80 COMPATIBLE.

ASCII: Converts a four digit BCD number into an ASCII string. The routine must be entered with the BCD number in register pair DE and the address of the buffer that the ASCII string is to be placed in in register pair HL. NOT PLM/80 COMPATIBLE.

```
ASM80 DVRFRQ.SRC SYMBOLS NOPAGING TITLE('WTC-520A DRIVER')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          MODULE
WTC-520A DRIVER

  LOC  OBJ          LINE     SOURCE STATEMENT

                       1          DSEG
  0030                 2  SC0BIN  EQU     00110000B       ;COU
                         NTER 0,2BYTE R/W,MODE 0,BINARY
  0031                 3  SC0BCD  EQU     00110001B       ;
                         "    "   "   "     "   " BCD
  0070                 4  SC1BIN  EQU     01110000B       ;
                         "    1   "   "     "     BINARY
  0071                 5  SC1BCD  EQU     01110001B       ;
                         "    "   "   "     "     BCD
  00B0                 6  SC2BIN  EQU     10110000B       ;
  00B1                 7  SC2BCD  EQU     10110001B
  0079                 8  SC1MD4  EQU     01111001B       ;SET
                         COUNTER 1 MODE 4
  0000                 9  LC0     EQU     00000000B
  0040                10  LC1     EQU     01000000B
  0080                11  LC2     EQU     10000000B
  0024                12  RWC0    EQU     36D             ;I/O
                         PORT ADR COUNTER 0
  0025                13  RWC1    EQU     37D             ;I/O
                         PORT ADR COUNTER 1
  0026                14  RWC2    EQU     38D
  0027                15  P8253   EQU     39D             ;I/O
                         PORT ADR CONTROL WORD
                      16         CSEG
                      17  ;
                      18  ;-----------------------------------

                      19  ; ROUTINE TO READ THE WTC520 FREQUEN
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

                        CY COUNTER CARD. PARAMETERS PASSED
                     20 ; ARE THE INTEGRATION PERIOD IN REGI
                        STER DE AND THE ADDRESS OF THE DATA
                     21 ; BUFFER THAT THE COUNT IS TO BE PLA
                        CED IN IN REG BC. RETURNS TO THE
                     22 ; CALLING PROGRAM WITH THE ASCII REP
                        RESENTATION OF THE COUNT IN THE DES-
                     23 ; IGNATED BUFFER.
                     24 ;-----------------------------------
                     25 ;
0000 E5              26 FREQ:    PUSH    H
                     27 ; TO USE AS BINARY COUNTER LOAD A WI
                        TH SC0BIN
0001 3E71            28 INITAL: MVI      A,SC1BCD        ;SET
                        COUNTER 0 WORD
0003 D327            29          OUT     P8253           ;SET
                        COUNTER 0
0005 3E31            30          MVI     A,SC0BCD        ;SET
                        COUNTER 0 WORD
0007 D327            31          OUT     P8253
0009 3E79            32          MVI     A,SC1MD4        ;SET
                        COUNTER 0 MD 4 WORD
000B D327            33          OUT     P8253
                     34 ; TO USE AS BINARY CTR, LOAD A WITH
                        SC1BIN
000D 3E71            35          MVI     A,SC1BCD        ;RES
                        ET COUNTER 0
000F D327            36          OUT     P8253
0011 3E99            37          MVI     A,99H           ;BCD
                        99
0013 D325            38          OUT     RWC1
0015 D325            39          OUT     RWC1            ;CTR
                        1 NOW CONTAINS MAX COUNT
                     40 ; IF USING AS BINARY CTR CHANGE 99 T
                        O FF
0017 7B              41          MOV     A,E             ;GET
                        INTPER LSB
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

0018 D324          42          OUT     RWC0            ;WRI
                   TE LSB TO CTR 0
001A 7A            43          MOV     A,D             ;MSB
                   YTE INTPER IN A
001B D324          44          OUT     RWC0            ;WRI
                    TE TO CTR 0 CTRS NOW COUNTING
                   45 ; NOW SEE IF COUNT DOWN TO ZERO
001D CD4200    C   46 TSTMSB: CALL     READ0           ;REA
                   D COUNTER 0
0020 AF            47          XRA     A               ;ZER
                   O A
0021 BC            48          CMP     H               ;SEE
                    IF MSBYTE 0
0022 C21D00    C   49          JNZ     TSTMSB          ;JUM
                   P IF NOT ZERO
0025 CD4200    C   50 TSTLSB: CALL     READ0
0028 AF            51          XRA     A               ;ZER
                   O A
0029 BD            52          CMP     L               ;SEE
                    IF LSB 0
002A C22500    C   53          JNZ     TSTLSB          ;JUM
                   P IF NOT ZERO
                   54 ;NOW GET COUNT FROM COUNTER 1
                   55 ;
002D CD4D00    C   56 GETC1:  CALL     READ1
                   57 ;-------------------
                   58 ; NOW SUBTRACT FROM BCD 9999
                   59 ;       NOTE: SINGE ALL SUBTRACTS AR
                   E DONE FROM 9999 THERE CAN NEVER
                   60 ;BE A CARRY OR AUXILLAY CARRY, AND T
                   HERE IS NO NEED FOR DECIMAL ADJUST.
                   61 ;THIS IS A PROPERTY OF THE NUMBER SY
                   STEM AND THE PLACE 9999 HAS IN A FOUR

                   62 ;DIGIT SYSTEM.
                   63 ;
0030 AF            64          XRA     A               ;CLE
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

                        AR CARRIES
0031 3E99               65          MVI     A,99H           ;INI
                        TIAL COUNT LSBYTE
0033 95                 66          SUB     L               ;SUB
                         COUNT FROM INITIAL
0034 6F                 67          MOV     L,A             ;RES
                        TORE TO HL
0035 AF                 68          XRA     A
0036 3E99               69          MVI     A,99H           ;INI
                        TIAL COUNT MSBYTE
0038 94                 70          SUB     H               ;SUB
                        COUNT FROM INITIAL W/ BORROW
0039 67                 71          MOV     H,A             ;RES
                        TORE TO HL
                        72 ; NOW SAVE IN DE REG PAIR
003A EB                 73          XCHG
                        74 ; BRING BUFFER ADDRESS FROM BC TO HL
003B 60                 75          MOV     H,B
003C 69                 76          MOV     L,C
003D CD5800    C        77          CALL    ASCII
0040 E1                 78          POP     H               ;RES
                        TORE ENVIRONMENT
0041 C9                 79          RET
                        80 ;
                        81 ;-----------------------------------
                        82 ;ROUTINE TO READ COUNTER 0 "ON THE F
                        LY" RETURNS COUNT IN HL
                        83 ;DESTROYS THE CONTENTS OF A REG
                        84 ;-----------------------------------
                        85 ;
0042 3E00               86 READ0:   MVI     A,LC0           ;SAM
                        PLE COMMAND
0044 D327               87          OUT     P8253           ;LAT
                        CH COUNTER
0046 DB24               88          IN      RWC0            ;REA
                        D LSBCOUNTER
0048 6F                 89          MOV     L,A
```

ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

```
0049 DB24          90          IN      RWC0            ;REA
                       D MSBYTE
004B 67            91          MOV     H,A             ;NOW
                       COUNT IN HL
004C C9            92          RET
                   93 ;
                   94 ;--------------------------------
                   95 ;ROUTINE TO READ COUNTER 1 ON THE FL
                       Y. RETURNS COUNT IN HL
                   96 ;DESTROYS CONTENTS OF A REG
                   97 ;--------------------------------
                   98 ;
004D 3E40          99 READ1:   MVI     A,LC1           ;SAM
                       PLE COMMAND
004F D327         100          OUT     P8253           ;LAT
                       CH COUNTER
0051 DB25         101          IN      RWC1            ;REA
                       D LSBCOUNTER
0053 6F           102          MOV     L,A
0054 DB25         103          IN      RWC1            ;REA
                       D MSBYTE
0056 67           104          MOV     H,A             ;NOW
                       COUNT IN HL
0057 C9           105          RET
                  106 ;
                  107 ;--------------------------------
                  108 ; ROUTINE TO CONVERT 4 DIGIT BCD COU
                       NT TO ASCII IN THE DATA BUFFER.
                  109 ; REQUIRES THE BUFFER ADDRESS IN HL
                       AND THE BCD NUMBER IN DE.
                  110 ; A REGISTER DESTROYED.
                  111 ;--------------------------------
                  112 ;
0058 3604         113 ASCII:  MVI     M,04H           ;PUT
                       CHAR COUNT IN 1ST BYTE
005A 23           114          INX     H
005B 7A           115          MOV     A,D             ;GET
```

ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

```
                                2 MSNIBBLES
005C 0F          116            RRC                              ;EXC
                              HANGE NIBBLES
005D 0F          117            RRC
005E 0F          118            RRC
005F 0F          119            RRC
0060 E60F        120            ANI     00001111B                ;CLE
                              AR UPPER NIBBLE
0062 F630        121            ORI     30H                      ;FOR
                              M ASCII CHAR
0064 77          122            MOV     M,A                      ;SAV
                              E IN DABUFR
0065 23          123            INX     H
0066 7A          124            MOV     A,D                      ;GET
                                2 MSNIBBLES
0067 E60F        125            ANI     00001111B                ;DIS
                              CARD MSNIBBLE
0069 F630        126            ORI     30H                      ;FOR
                              M ASCII CHAR
006B 77          127            MOV     M,A
006C 23          128            INX     H
006D 7B          129            MOV     A,E                      ;GET
                                LSNIBBLE
006E 0F          130            RRC
006F 0F          131            RRC
0070 0F          132            RRC
0071 0F          133            RRC
0072 E60F        134            ANI     0FH                      ;CLE
                              AR UPPER NIBBLE
0074 F630        135            ORI     30H                      ;FOR
                              M ASCII
0076 77          136            MOV     M,A
0077 23          137            INX     H
0078 7B          138            MOV     A,E
0079 E60F        139            ANI     0FH
007B F630        140            ORI     30H
007D 77          141            MOV     M,A
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER FREQUENCY BOARD DRIVER

                        142 ; ALL OF COUNT NOW ASCII IN THE BUFE
                            R
   007E C9              143         RET
                        144 PUBLIC  FREQ,READ0,READ1,ASCII,TSTMS
                            B,TSTLSB,GETC1
                        145         END

PUBLIC SYMBOLS
ASCII  C 0058   FREQ   C 0000   GETC1  C 002D   READ0  C
                       0042   READ1  C 004D   TSTLSB C 00
                       25   TSTMSB C 001D

EXTERNAL SYMBOLS

USER SYMBOLS
ASCII  C 0058   FREQ   C 0000   GETC1  C 002D   INITAL C
                        0001    LC0    A 0000    LC1    A 00
                       40    LC2    A 0080  0042

READ1  C 004D   RWC0   A 0024    RWC1   A 0025   RWC2   A
                       0026    SC0BCD A 0031

                       SC0BIN A 0030   SC1BCD A 0071   SC
                       1BIN A 0070   SC1MD4 A 0079   SC2B
                       CD A 00B1   SC2BIN A 00B0   TSTLSB
                       C 0025
ASSEMBLY COMPLETE,  NO ERRORS
```

APPENDIX G

THE A TO D CONVERTER DRIVER

PARAMETER PASSING SPECIFICATIONS FOR THE A-TO-D DRIVER

The routines and labels in the file DVRDCV.SRC:

MSET: Sets the mode of the 8255 parallel interface chip. This routine destroys the A register.

RSET: Resets the WTC-700 system. This routine destroys the A register.

RDRDY: Initializes the WTC-700 system by calling MSET and RSET. All registers returned intact.

RDAD: Reads the WTC-700 120A analog to digital conversion card. The card number and device number on the card must be placed in register A. The address of the buffer that the results are to be placed in is in register pair HL. THIS ROUTINE IS NOT PLM/80 COMPATIBLE.

DVRATD: Calls RDAD. The register pair BC must contain the address of the buffer that the data is to be placed in. The card and source number must be in register E. This routine serves as the calling routine when the A-to-D converter is called from PLM/80.

```
ASM80 DVRDCV.SRC SYMBOLS NOPAGING TITLE('WTC-700 A TO D DRI
                      VER')


ISIS-II 8080/8085 MACRO ASSEMBLER, V3.0          MODULE
WTC-700 A TO D DRIVER

  LOC  OBJ          LINE      SOURCE STATEMENT

                       1          DSEG
  0098                 2 MDSET    EQU     098H     ;8255 RESET
  0001                 3 READ     EQU     01H      ;READ WTC700
                         COMMAND
  0004                 4 PRTA     EQU     04H      ;8255 CONTRO
                         LER PORT A
  0005                 5 PRTB     EQU     05H      ;  "      "
                               "   B
  0006                 6 PRTC     EQU     06H      ;  "      "
                             "     C
  0007                 7 PRTD     EQU     07H      ;8255 COMMAN
                         D PORT
                       8          CSEG
                       9 ;
                      10 ;-------------------------------------
                         ------
                      11 ;THESE ROUTINES ARE DRIVERS FOR THE
                         WTC-700 DATA INTERFACE SYSTEM.  THEY
                      12 ;PERMIT ANY CHANNEL ON ANY SOURCE CA
                         RD TO BE READ.  THESE DRIVERS REQUIR
                         E
                      13 ;TWO EXTERNAL PARAMETERS TO OPERATE.
                          THESE PARAMETERS ARE A STORAGE BUFF
                         ER
                      14 ;CALLED DABUFR AND A CONSTANT, DABUF
                          L, WHICH IS THE LENGTH OF DABUFR MIN
                         US
```

ISIS-II 8080-8085 MACRO ASSEMBLER A-TO-D DRIVER

```
                        15 ;ONE.  TO SUCCESFULLY LINK ANOTHER
                          PROGRAM TO THESE DRIVERS THE OTHER P
                          RO-
                        16 ;GRAM MUST HAVE THESE TWO PARAMETERS
                          DECLARED AS PUBLICS.  ALL ENTRYPOIN
                          T
                        17 ;LABLES HAVE BEEN DECLARED PUBLIC IN
                          THIS PROGRAM AND ARE ACCESSABLE TO
                        18 ;THE PROGRAM USER.  FOR REGISTER USE
                          AGE, CONSULT THE COMMENTS WITH THE
                         19 ;INDIVIDUAL ROUTINES.
                        20 ;-----------------------------------
                          ------
                        21 ;
                        22 ;
                        23 ;-----------------------------------
                          ------
                        24 ; THIS ROUTINE CALLS READS ANY CARD
                          AND DEVICE WHEN PASSED THE DEVICEOCA
                          RD
                        25 ; NUMBER AND THE ADDRESS OF THE BUFF
                          ER THE DATA IS TO BE PLACED IN.
                        26 ;-----------------------------------
                          ------
                        27 ;
0000 C5                 28 DVRATD: PUSH    B
0001 E1                 29         POP     H                ;BUF
                          R ADDRESS NOW IN HL
0002 CD1800   C         30         CALL    RDRDY            ;GET
                           WTC700 SYSTEM READY
0005 7B                 31         MOV     A,E              ;PUT
                          DEVICE & CARD # IN C
0006 CD2300   C         32         CALL    RDAD             ;GET
                           READING INTO DABUFR
0009 C9                 33         RET
                        34 ;
                        35 ;
```

```
                        36 ;-----------------------------------
                           ------
                        37 ;THIS ROUTINE SETS THE MODE OF THE 8
                           255 SYSTEM CONTROLER IN THE WTC-700
                        38 ;IT IS USUALLY CALLED PRIOR TO ANY A
                           TTEMPT AT I/O WITH THE WTC700 SYSTEM
                           .
                        39 ;THE ROUTINE DESTROYS REGISTER A, BU
                           T ALL OTHER ROUTINES ARE RETURNED IN
                           TACT
                        40 ;-----------------------------------
                           ------
                        41 ;
000A 3E98               42 MSET:    MVI     A,MDSET ;GET 8255 RE
                           SET
000C D307               43          OUT     PRTD    ;SEND COMMAN
                           D
000E C9                 44          RET
                        45 ;
                        46 ;-----------------------------------
                           -------
                        47 ;THIS ROUTINE RESETS THE ENTIRE WTC-
                           700 SYSTEM. IT USUALLY FOLLOWS THE
                        48 ;ROUTINE MSET.  THE CONTENTS OF THE
                           A REGISTER ARE DESTROYED, BUT ALL OT
                           HER
                        49 ;REGISTERS ARE RETURNED INTACT.
                        50 ;-----------------------------------
                           -------
                        51 ;
000F 3E08               52 RSET:    MVI     A,8H    ;WTC700 RESE
                           T COMMAND
0011 D306               53          OUT     PRTC
0013 3E00               54          MVI     A,0H    ;REMOVE WTC7
                           00 RESET
0015 D306               55          OUT     PRTC
0017 C9                 56          RET
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER A-TO-D DRIVER

                        57 ;
                        58 ;-----------------------------------
                           ----------
                        59 ;THIS ROUTINE INITIALIZES THE 8255 S
                           YSTEM CONTROLER AND THE WTC-700
                        60 ;ALL REGISTERS ARE RETURNED INTACT.
                        61 ;-----------------------------------
                           ----------
                        62 ;
0018 D5                 63 RDRDY:  P
                           USH     D
0019 57                 64         MOV     D,A     ;SAVE CARD &
                            SOURCE ADR.
001A CD0A00   C         65         CALL    MSET    ;SET 8255 MO
                           DE
001D CD0F00   C         66         CALL    RSET
0020 7A                 67         MOV     A,D     ;CARD,SOURCE
                            ADR IN A
0021 D1                 68         POP     D
0022 C9                 69         RET
                        70
                           ;
                        71 ;-----------------------------------
                           ----
                        72 ;THIS ROUTINE READS A SOURCE CARD A
                           ND CHANNEL IN THE WTC-700
                        73 ;SYSTEM.  THE CARD ADDRESS AND SOURC
                           E ADDRESS ARE PLACED IN REGISTER
                        74 ;A PRIOR TO CALLING THE ROUTINE.  TH
                           E ADDRESSES IN THE A REGISTER FOLLOW
                        75 ;THIS CONVENTION:  THE MOST SIGNIFIC
                           ANT NIBBLE CONTAINS THE CHANNEL ADDR
                           ESS
                        76 ;PERMISSSBLE ADDRESSES ARE BINARY 4
                           THRU 7.  THE LEAST
                           SIGNIFICANT NIBBLE
                        77 ;CONTAINS THE CARD ADDRESS.  PERMISS
```

```
                        ABLE ADDRESSES ARE BINARY 0 - 15.
                     78 ;THE ROUTINE DESTROYS THE A REGISTER
                        , BUT ALL OTHER REGISTERS ARE RETURN
                        ED
                     79 ;INTACT.  THE OUTPUT FROM THE DATA C
                        ARD IS PLACED AT THE LOCATION SPECIF
                        IED
                     80 ;BY REGISTER PAIR HL.  THE CHARACTER
                         AT THIS LOCATION BEING THE BINARY
                     81 ;REPRESENTATION OF THE NUMBER OF CHA
                        RACTERS IN THE BUFFER.
                     82 ;-------------------------------------
                        -----
                     83 ;
0023 D305            84 RDAD:    OUT     PRTB    ;OUTPUT CARD
                        ,SOURCE TO 8255
0025 C5              85          PUSH    B
0026 E5              86          PUSH    H
0027 01FF00          87          LXI     B,255D  ;BUF LENGTH
                        & CHAR COUNT
002A 23              88          INX     H       ;ROOM FOR CH
                        AR COUNT
002B 3E01            89 RADL1:   MVI     A,READ  ;GET READ WT
                        C700 COMMAND
002D D306            90          OUT     PRTC    ;ISSUE COMMA
                        ND
002F DB06            91 STLOOP:  IN      PRTC    ;BRING IN ST
                        ATUS WORD
0031 E610            92          ANI     10H     ;MASK OFF BI
                        T 4
0033 CA2F00    C     93          JZ      STLOOP  ;TRY AGAIN I
                        F BIT 4 0
0036 DB06            94          IN      PRTC    ;BRING IN ST
                          ATUS WORD
0038 E620            95          ANI     020H    ;MASK OFF BI
                           T 5
003A C24D00    C     96          JNZ     ENDST   ;JUMP IF LAS
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER A-TO-D DRIVER

                        T CHAR
003D DB04               97          IN      PRTA    ;ACCEPT DATA
003F 77                 98          MOV     M,A     ;PUT DATA IN
                         BUFFER
0040 04                 99          INR     B       ;INCREMENT C
                        HAR COUNT
0041 0D                 100         DCR     C       ;DECREMENT B
                        UFFER LENGTH
0042 CA5300     C       101         JZ      FULL    ;EXIT IF BUF
                        FER FULL
0045 23                 102         INX     H       ;INCREMENT B
                        UFFER POINTER
0046 3E00               103         MVI     A,0H    ;SIGNAL FOR
                        ANOTHER CHAR
0048 D306               104         OUT     PRTC    ;ISSUE COMMA
                        ND
004A C32B00     C       105         JMP     RADL1   ;GET ANOTHER
                         CHARACTER
004D DB04               106 ENDST:  IN      PRTA    ;ACCEPT LAST
                         CHAR OF STRING
004F 77                 107         MOV     M,A     ;PUT DATA IN
                         BUFFER
0050 04                 108         INR     B       ;INCREMENT C
                        HAR COUNT
0051 0D                 109         DCR     C       ;DECREMENT B
                        UFFER LENGTH
0052 23                 110         INX     H       ;INCREMENT B
                        UFFER POINTER
0053 78                 111 FULL:   MOV     A,B     ;PUT CHAR CO
                        UNT IN A
0054 2F                 112         CMA
0055 5F                 113
                            MOV     E,A
0056 16FF               114         MVI     D,0FFH  ;HAVE FORMED
                        2'S COMPLEMENT OF (CHARCOUNT + 1)
0058 19                 115         DAD     D       ;NOW HAVE RE
```

```
ISIS-II 8080-8085 MACRO ASSEMBLER A-TO-D DRIVER


                        CREATED BUFFER BEG ADR
  0059 70           116         MOV     M,B     ;CHAR COUNT
                        IN BUFFER TO B
  005A C1           117         POP     B
  005B E1           118         POP     H
  005C C9           119         RET
                    120         PUBLIC DVRATD,RSET,RDRDY,RD
                        AD,MSET
                    121         END


PUBLIC SYMBOLS
DVRATD C 0000    MSET   C 000A    RDAD   C 0023    RDRDY  C
                        0018    RSET   C 000F

EXTERNAL SYMBOLS


USER SYMBOLS
DVRATD C 0000    ENDST  C 004D    FULL   C 0053    MDSET  A
                        0098    MSET   C 000A    PRTA   A 00
                        04    PRTB   A 0005 0023

                        RDRDY  C 0018    READ   A 0001    RS
                        ET     C 000F

                        STLOOP C 002F

ASSEMBLY COMPLETE,   NO ERRORS
```

[redacted]