NBS Technical Note 1221

NBS Wideband Sampling Wattmeter

G. N. Stenbakken, O. B. Laug, T. H. Kibalo, B. A. Bell, and A. G. Perrey

Electrosystems Division
Center for Electronics and Electrical Engineering
National Engineering Laboratory
National Bureau of Standards
Gaithersburg, MD 20899

May 1987

U.S. Department of Commerce
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

National Bureau of Standards
Technical Note 1221
Natl. Bur. Stand. (U.S.),
Tech. Note 1221
103 pages (May 1987)
CODEN: NBTNAE

U.S. Government Printing Office
Washington: 1987

For sale by the Superintendent
of Documents,
U.S. Government Printing Office,
Washington, DC 20402

Table of Contents

List of Figures

List of Tables

Program Listings

NBS WIDEBAND SAMPLING WATTMETER

G. N. Stenbakken, O. B. Laug, T. H. Kibalo
B. A. Bell, and A. G. Perrey

The design and operation of a wideband sampling wattmeter capable of measuring distorted power signals with fundamental frequencies from 1 Hz to 10 kHz and harmonics up to 100 kHz is described. The microcomputer controlled wattmeter uses asynchronous sampling of the voltage and current signals. The errors associated with this type of operation are described, as are various methods of correcting some of these errors. A hardware multiplier-accumulator allows a large number of power signal samples to be integrated for each measurement. Sampling rates are variable up to a maximum of 300 kHz. A direct-memory-access unit is used to capture 4096 samples of both the voltage and current signals. These data are used to calculate the average and rms values of these signals.

A special feature of the sampling wattmeter is the use of programmable time delay circuits to compensate for differential time delays between the two input channels. Performance tests of the wattmeter show that it has a measurement uncertainty of less than ± 0.1 percent of full-scale amplitude over the above described frequency range.

This technical note gives schematic diagrams of the circuits used in this wattmeter and describes their operation. The software is also described, and flow charts and selected program listings are provided for the programs written in Pascal. The results of calibration of the instrument over the past year are also presented.

Key words: calibration; power measurement; sampling voltmeter; sampling wattmeter; signal analyzer; truncation error; wideband sampling wattmeter.

1. INTRODUCTION

The design, operation and performance of a new wideband sampling wattmeter are described in this report. The wattmeter was developed to meet the growing need for accurate methods of measuring highly distorted power signals in the 1 kHz to 100 kHz frequency range. The development of this instrument was carried out at the National Bureau of Standards (NBS) and received substantial financial support from the Department of Energy (DOE). DOE's interest in this project is related to their efforts to accurately characterize the efficiency of dc/ac converter systems used in powering electric vehicles. Such systems operate by chopping the dc power source at frequencies ranging from 1 to 10 kHz [1]. These systems produce signals with significant harmonics above 50 kHz on both the ac output as well as on the chopped dc input power. Many other DOE technology programs also require the accurate measurement of distorted power signals. These efforts include more widespread use of dc/ac conversion as a result of expanded use of HVDC

transmission, use of large battery storage technology by power utilities, and solar cell power generation systems.

Based on these needs, NBS began the development of a wideband wattmeter which could accurately measure distorted power in systems with fundamental signal frequencies up to 10 kHz and harmonics up to 100 kHz with an uncertainty of less than ± 0.1 percent of full scale amplitude. Several techniques were considered as candidates for this new wattmeter. Time-division-multipliers and thermal-converter-based techniques have very good accuracies at power line frequencies, but their accuracy drops rapidly at frequencies above 1 kHz. Analog multipliers, on the other hand, have the necessary frequency response but not the desired accuracy. The approach selected was a dual-channel sampling wattmeter using custom designed input amplifiers, commercial track-and-hold (T/H) amplifiers, and analog-to-digital (A/D) converters. An earlier sampling wattmeter that measured signals up to 5 kHz had already been successfully developed at NBS [2]. This new wideband NBS sampling wattmeter operates at sampling rates up to 300 kHz using 12-bit A/D converters. To accommodate future upgrading of the wattmeter, the input sections of the wattmeter have been designed as plug-in modules. The digital portions of the wattmeter can accommodate 16-bit data at sampling rates of up to 1 MHz.

The design of the NBS wideband sampling wattmeter makes accurate operation possible over a wide frequency range. Power measurements can be made on dc signals and ac signals from 1 Hz to over 100 kHz. A high speed digital multiplier-accumulator is used to calculate the power in real time from a large number of input signal samples. Real time computations, together with the fixed sampling approach, accounts for the wide frequency range of the wattmeter. The use of trigger and counter circuits provides an automatic summation interval control which accounts for the high accuracy, independent of the frequency, from 1 Hz to 10 kHz. Response from 10 kHz to over 100 kHz is limited by the input amplifiers. A feature unique to the NBS sampling wattmeter is the use of programmable time delay circuits to compensate for differential time delays between the two input channels. Without this correction the accuracy of the wattmeter would be seriously affected at higher frequencies.

A microcomputer is used to control the operation of the wattmeter. Direct-memory-access (DMA) channels are used to store data from the two channels which allows measurements to be made on the input voltage and current waveforms. New algorithms were developed which improve the accuracy with which the average and rms values of these two signals are determined.

Chapter 2 describes the basic operating principles of the NBS wideband sampling wattmeter. A brief description is given of the circuits and software used in the wattmeter. The operator interface with the sampling wattmeter is covered in chapter 3. Each of the settable parameters and their range of possible values is described. Chapter 4 provides a detailed description of the hardware and circuits used in the wattmeter. Simplified circuit diagrams are given and the operation of the circuits is explained. Chapter 5 presents flow charts for the major software and explains the operation of the software. The results of the many calibration tests performed on the sampling wattmeter

are presented in chapter 6. Appendix A shows the complete circuit diagrams of the sampling wattmeter and appendix B provides the major software programs.

2. SYSTEM DESCRIPTION

This section describes the basic principles of operation of sampling wattmeters with emphasis on the approach taken for the NBS wideband sampling wattmeter. Details on the theory of sampling wattmeters can be found in references [3], [4] and [5]. A brief overview of the hardware and software used in the NBS wideband sampling wattmeter is also given in this chapter.

2.1. Basic Sampling Principles

For periodic power signals having a period of T seconds, voltage v(t), and current i(t), the true power P is given by

$$P = \frac{1}{T} \int_0^T v(t)\ i(t)\ dt\ . \tag{1}$$

Sampling wattmeters measure power by averaging many samples of instantaneous power. These instantaneous power values are obtained by sampling the voltage and current waveforms simultaneously, converting these samples to digital values, and calculating their products. If the voltage at time t_k is denoted as $V(t_k)$ and the current as $I(t_k)$, then the average power W is given by

$$W = \frac{1}{n} \sum_{k=0}^{n-1} V(t_k)\ I(t_k)\ , \tag{2}$$

where n is the number of samples used in the average. The major error in determining W comes from two sources: first, the limitations inherent in the sampling process, and second, the imperfections of the hardware used to make the measurements. The errors from the sampling process are described next, with reference to the sampling approach used in the NBS sampling wattmeter. The errors due to hardware imperfections are described in chapter 6.

2.1.1. Sampling Errors

The easiest way to look at the errors associated with sampling wattmeters is to consider measuring simple sinusoidal power. Figure 2.1.1 shows the relationship between the sample times and the associated voltage and current signals. To calculate the power in such signals, the sample values from a finite interval of the waveforms are used. This interval is called the summation interval and is represented in figure 2.1.1 by the circled sample values. The sample interval can be normalized by expressing it in terms of the phase angle change which either signal makes during this interval. Thus, for a signal of period T seconds and a sampling rate of r samples per second, the sample interval γ in radians per sample is given by

$$\gamma = \frac{2\pi}{rT}\ . \tag{3}$$

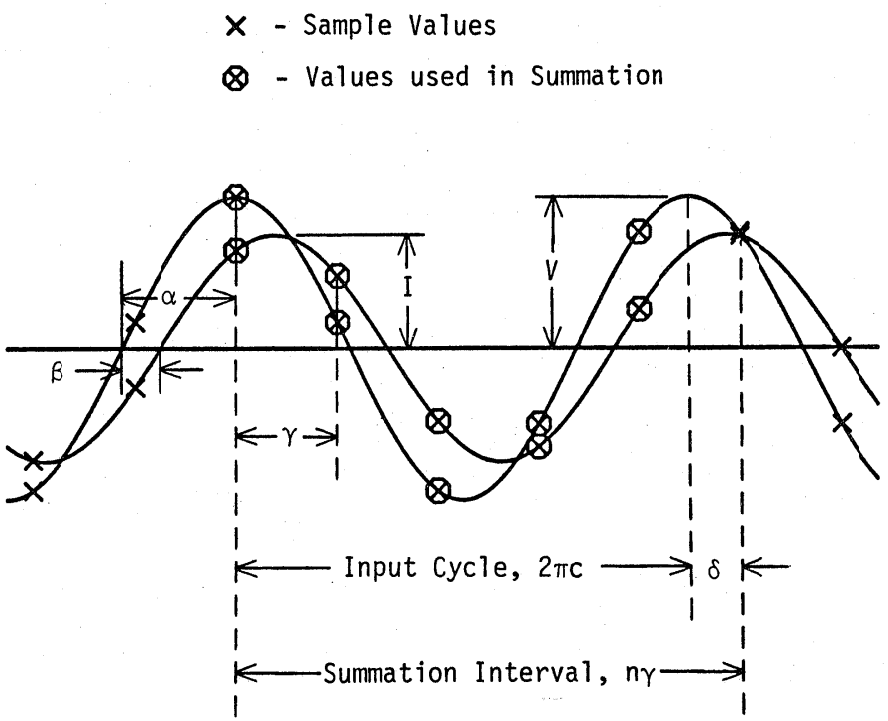


Figure 2.1.1 Sampling of Voltage and Current Sinewaves for Measuring Simple Sinusoidal Power

The voltage and current sample values are given by

$$V(t_k) = V_k = V \sin(k\gamma + \alpha) \tag{4a}$$

and

$$I(t_k) = I_k = I \sin(k\gamma + \alpha + \beta) \ , \qquad k = 0 \text{ to } n-1 \tag{4b}$$

where k is the sample number, α is the phase angle of the first voltage sample relative to the zero crossing, β is the relative phase angle between the current and voltage, and V and I are the peak values. Thus, if the power W is calculated as above, this gives,

$$W = \frac{VI}{2} \left[\cos \beta - \frac{1}{n} \sum_{k=0}^{n-1} \cos(2k\gamma + 2\alpha + \beta)\right] \ . \tag{5}$$

For sine waves the true power P is equal to the first term: thus, the summation terms can be interpreted as the error E. This can be expressed in terms of the truncation angle δ which is the difference between the summation interval $n\gamma$ and the nearest number of whole cycles c of the signal, that is $\delta = 2\pi c - n\gamma$. Using these factors the error is given by [6]

$$E = \frac{VI}{2n} \frac{\sin \delta}{\sin \gamma} \cos(2\alpha + \beta - \delta - \gamma) \ , \tag{6}$$

or as

$$E = \frac{VI\gamma}{4\pi c'} \frac{\sin \delta}{\sin \gamma} \cos(2\alpha + \beta - \delta - \gamma) \ , \tag{7}$$

where c′ is the number of cycles and partial cycles of the signal in the summation interval, and is given by

$$c' = \frac{n\gamma}{2\pi} \ . \tag{8}$$

The error given in (6) and (7) is called the truncation error and arises because the summation interval does not exactly match an integral number of cycles of the signal. Equation 7 shows that if the truncation angle δ is zero this error is zero. This is the reason many sampling wattmeters use synchronous sampling for which δ is zero.

For the NBS sampling wattmeter to achieve its desired accuracy, the truncation error must be much less than ± 0.1 percent of the full scale amplitude (FSA). Equation (6) shows that this error can be reduced by keeping δ small and n large. Consider the problem caused by not controlling δ. In this case, (7) shows that the truncation error is inversely proportional to c′ the number of signal cycles in the summation interval. Thus, at low signal frequencies the summation interval must be very long. For example, to keep this error to less than ± 0.1 percent FSA requires a summation interval of at

least 160 cycles. For a 10 Hz signal this means a measurement time of 16 seconds, which is unacceptable.

Although the NBS sampling wattmeter has asynchronous sampling, it uses a trigger and cycle counting hardware to control the summation interval. By starting and stopping the summation interval with a trigger pulse synchronized with the signal being measured, the truncation angle δ is kept to less than the sample interval γ. Equation (6) shows that in this case the maximum error occurs when $\sin \delta = \sin \gamma$ and the cosine term is ± 1. Then, the maximum error is given by

$$E_{max} = \pm \frac{VI}{2n} , \tag{9}$$

which is independent of the input frequency. Thus, the truncation error can be made small by taking a large number of samples n independent of the input frequency. With the NBS sampling wattmeter, power measurements are typically made with an n of greater than 150,000 samples so that the truncation error is negligible. With a sampling rate of 300 kHz, this number of samples requires a summation interval of about one half second.

2.2. Theory of Operation

This section briefly describes how the wideband sampling wattmeter performs the operations described in the previous section (2.1). Figure 2.2.1 is a simplified block diagram of the wattmeter. The current and voltage signals are sampled and converted to digital quantities in the two Amplifier/Data Converter modules. This circuitry scales the input signal to a $\pm$ 5 V ac amplitude and then digitizes this signal. Scaled copies of the input signals called Trigger 1 and 2 are sent to the Summation Interval Control circuitry. In turn, the Summation Control circuitry provides the Delay 1 and Delay 2 signals to the Amplifier/Data Converter Modules, which control the sample time for the data conversion circuitry.

The trigger signals are used by the Summation Interval Control circuitry to synchronize the wattmeter data processing with the period of the input signals. As mentioned in section 2.1, this synchronization is required to get accurate measurement results. The Summation Interval Control circuitry performs the synchronization by counting a number of the periods of the input signals using one of the trigger signals. When the proper number of periods has been counted, a synchronization signal Sync is sent to the data processing circuits to indicate the termination of one summation interval and the start of the next one. The Summation Interval Control circuitry also counts the number of samples taken during each such interval and passes this number on to the microcomputer.

The two 12-bit binary numbers from the converter modules are sent to the Multiplier-Accumulator circuitry and to the Direct-Memory-Access (DMA) memory circuitry. The Multiplier-Accumulator multiplies the two numbers and sums the resulting products over the intervals controlled by the Sync signal from the Summation Interval Control circuitry. At the end of each interval the accumulated product results are sent to the microcomputer. The DMA circuitry

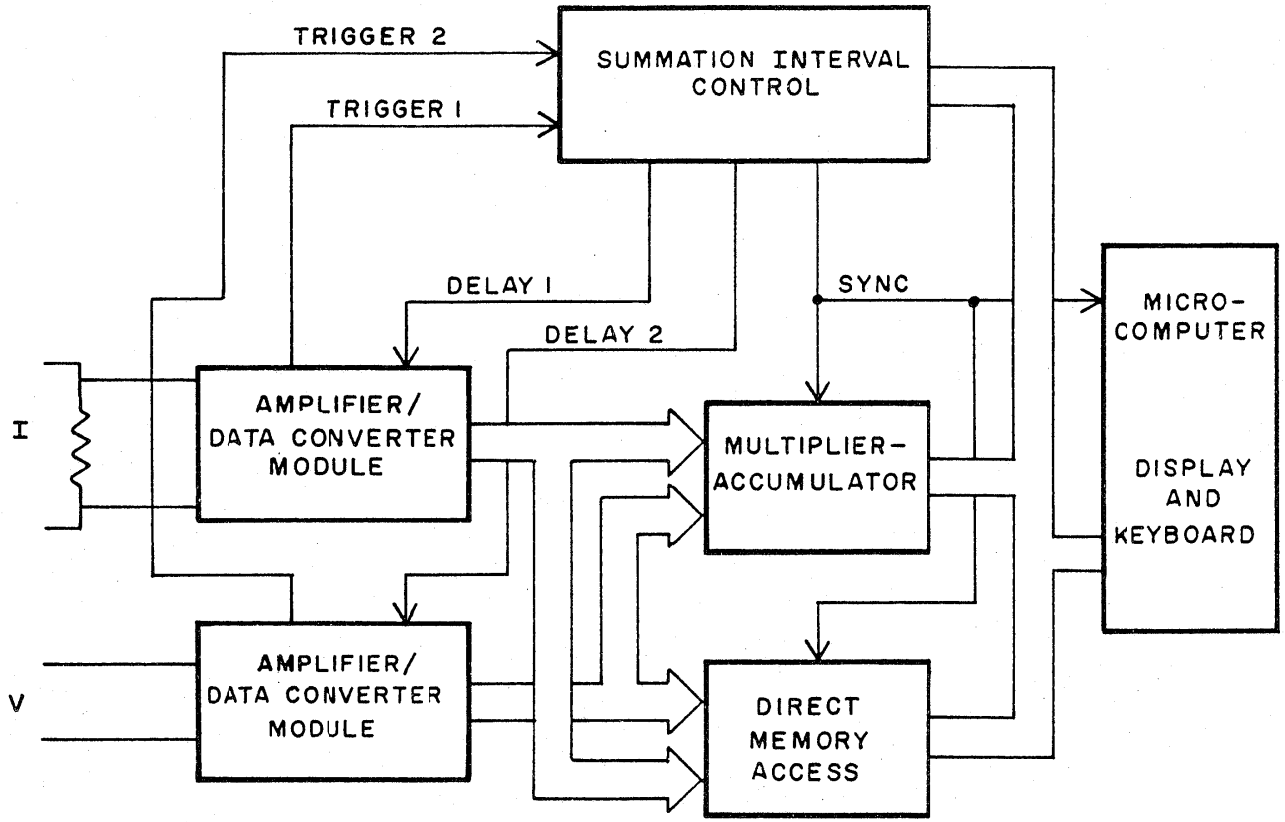


Figure 2.2.1 Simplified Block Diagram of Wideband Sampling Wattmeter

stores up to 4096 consecutive pairs of numbers from the two input channels.

The microcomputer calculates the average power for each of the summation intervals from the accumulated products and number of samples. It also uses the data stored in the DMA memory to calculate the average and root-mean-square (rms) values for the current and voltage input signals.

2.3. Hardware Overview[1]

This section describes the hardware used in the NBS wideband sampling wattmeter and the associated circuits to the block diagram level. A more detailed discussion of the various circuits is given in chapter 5, and a complete set of schematics is provided in appendix A.

A more detailed block diagram of the NBS sampling wattmeter is shown in figure 2.3.1. The input signal conditioning units consist of a variable gain amplifier, track-and-hold (T/H) unit, and a 12-bit analog-to-digital (A/D) converter. These units have six ranges which accept input signals with a peak amplitude of from ± 125 mV to ± 5 V. This input circuitry is optically isolated from the remainder of the system as indicated by the dashed lines in figure 2.3.1. This optical isolation ensures that the sampling wattmeter will not introduce ground loops into the measuring circuit.

The convert command signals, which initiate the data conversion process for the two input modules, are derived from a crystal controlled clock. The convert command signal can be set to rates of from 2.344 kHz to 300 kHz. The convert command signal passes though two programmable delay units which allows for a differential delay between the two signal channels of from -60 ns to +195 ns in one ns steps. The delay is used to compensate for the differential time delay between the two input amplifiers and can be used also, to compensate for any external differential time delays. The significance of this compensation can be illustrated by considering the potential errors from differential time delays. If the system is measuring a 10 kHz power signal at half-power-factor, i.e., a phase angle of 60°, then a differential time delay of only 18 ns will cause an error of ± 0.2 percent of the true power value. The differential time delay of the input amplifiers can be as large as 44 ns when set on different gains. Thus, without the ability to set differential time delays, the resulting error could be almost 0.5 percent for a half-power-factor signal.

The summation interval control is accomplished by the four blocks labeled trigger, cycle counter, delay, and sample counter in figure 2.3.1. An analog multiplexer in the trigger circuit allows selection of one of four fixed

---

[1] This technical note reports on equipment developed at the National Bureau of Standards. Certain commercial components are identified in this paper in order to adequately describe the design and operation of the equipment. Such identification does not imply recommendation or endorsement by the National Bureau of Standards, nor does it imply that the components are necessarily the best available for the purpose.

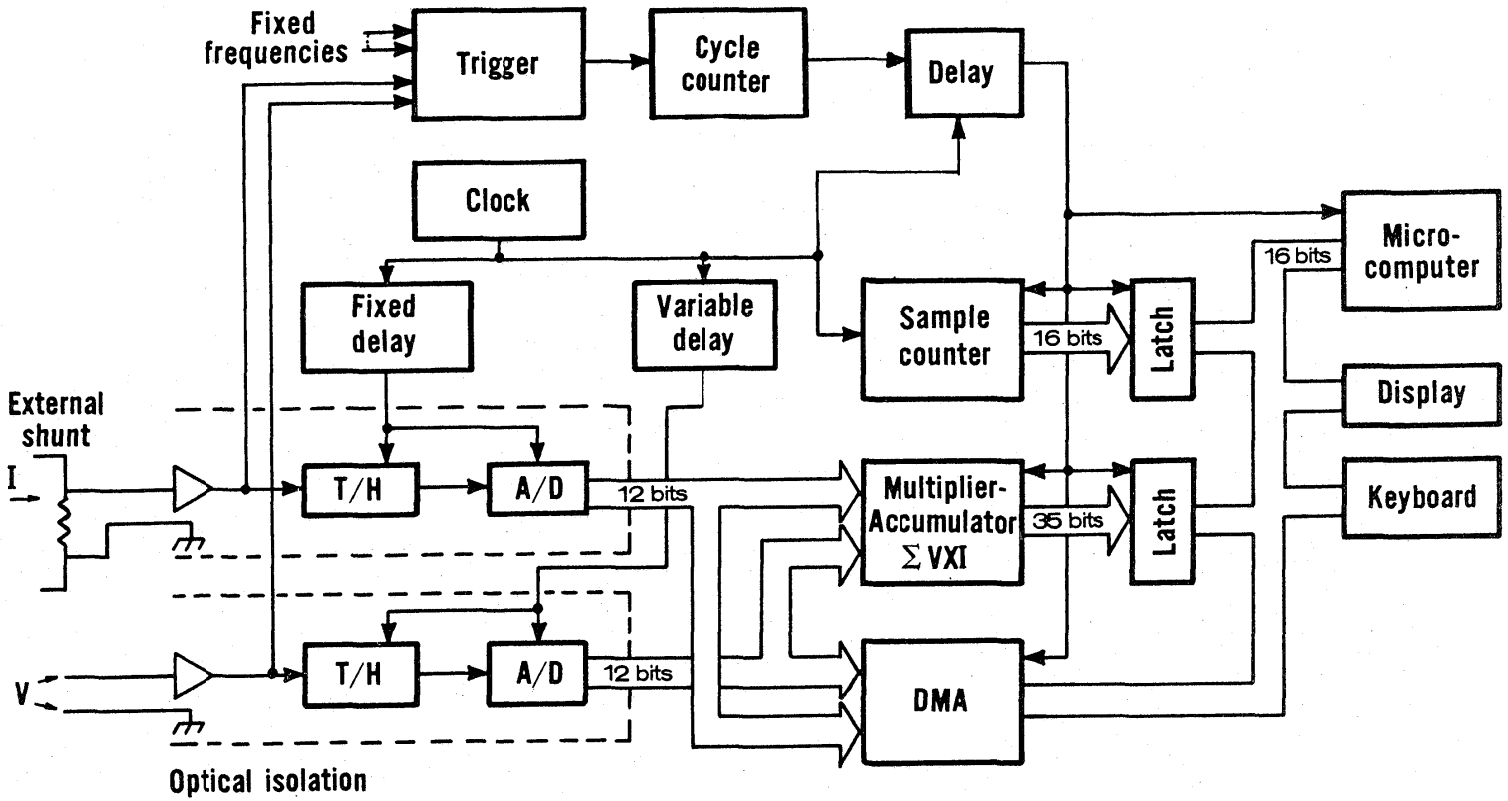


Figure 2.3.1 Block Diagram of the NBS Wideband Sampling Wattmeter

frequencies or a signal from either of the input channels. The choice of fixed frequencies allows the wattmeter to measure dc inputs. This ability is useful in calibrating the gain and offsets of the input modules. A filter option on the signals from the input modules (not shown in figure 2.3.1) also allows for clean triggering on a variety of input waveforms. The trigger level is programmable so that reliable triggering can be obtained on complex waveforms. The delay circuit synchronizes the trigger event with sample times and allows for a programmable delay in units of sample periods. This delay allows control of the phase angle at the start of the summation interval α which was discussed in section 2.1.1 above. The cycle counter determines the summation interval in units of input signal cycles while the sample counter determines the number of samples taken during each summation interval. The start and stop time for each summation interval is determined by the signal out of the cycle counter and delay circuits which indicates that the programmed number of cycles has been registered. This signal is used to strobe the data out of the sample counter, to strobe the data out of the multiplier-accumulator, to trigger the DMA, and to interrupt the microcomputer.

The digitized voltage and current values are multiplied and summed by the high speed multiplier-accumulator circuit. Although this circuit is presently multiplying two 12-bit numbers at up to 300 thousand products per second, the circuit is capable of multiplying two 16-bit numbers at a rate of one million products per second. This added capability is provided to accommodate improved amplifier/data converter modules in the future. The accumulator has a capacity of 35-bits which limits the number of products that can be summed without overflow. For full scale dc signals the limit is 4096 samples, whereas for a full scale sine wave the limit is 8192 samples. This summation limit can be software extended to allow for a total accumulation of over a million samples. The sample count and product accumulation data are strobed into latches, and the multiplier-accumulator circuits are cleared in time to respond to the next data sample. Thus, the software-extended summation interval is provided without any loss of data.

The digitized voltage and current values are also directed to the direct-memory-access (DMA) unit. This unit stores up to 4096 samples of both channels. Since the DMA unit is started with the same signal that strobes the sample counter and multiplier-accumulator, the data that is stored is synchronized with the same starting phase angle as the product data.

The microcomputer is a commercial (Intel SDK-86) single-board, 16-bit computer with 4k bytes of random-access-memory (RAM). The SDK-86 board utilizes the Intel 8086 microprocessor central processing unit (CPU) chip. A priority interrupt controller and a Multibus[2] interface have been added to this basic microcomputer. The software operating system is stored on a commercial programmable read only memory (PROM) board plugged into the Multibus. Presently only 24k of the available 64k bytes of memory space is being used to hold the system software (i.e., firmware).

---

[2] Multibus is a trademark of Intel Corporation.

The operator interface is via a 32-character alphanumeric display and two 12-key keyboards located on the front of the sampling wattmeter enclosure. The display is interfaced directly with the Multibus as two read-and-write addresses in the I/O space of the microcomputer. The keyboards are connected to the Multibus via a keyboard interface chip which performs the keyboard scanning function and buffers up to eight keystrokes. This chip provides an interrupt to the microcomputer when a key action is sensed. Twenty-three of the keys are connected to this interface chip while the last key (Reset) is directly wired to the reset line on the microcomputer.

The sampling wattmeter is entirely housed in a rack 27 inches high by 19.5 inches wide by 25 inches deep. The major parts of the system electronics are built on four circuit cards which conform to the Multibus I interconnection specification. These cards are mounted in a commercial Multibus cage. One of the cards is the commercial PROM board and the other three are cards made at NBS using wire wrap connections.

The SDK-86 microcomputer board is mounted above the card cage. It is connected to the other boards via a specially designed SDK-86 to Multibus I interface. The reason for using this approach instead of a commercial microcomputer card mounted in the Multibus cage is because the SDK-86 board provided for convenient prototype development. A current design would make use of an integrated Multibus microcomputer card.

The input signal conditioning is accomplished by two identical amplifier/data converter modules mounted below the card cage. These units plug into the system via a 44-pin connector. The display and keyboards are mounted on panels in the front of the enclosure and are connected via cables to one of the NBS-developed Multibus cards. The power supplies are mounted behind the card cage.

2.4. Software Overview

The operating system software for the sampling wattmeter is contained in the PROMs within the instrument. Most of this software is compiled from Pascal into native microprocessor code. These routines are interfaced with the hardware of the wattmeter via assembly language driver routines. Hardware interrupts synchronize the hardware functions with the software operation. The Pascal and assembly language routines are summarized below and described in more detail in section 6.

2.4.1. Pascal Routines

The main control program first initializes the hardware and software to the startup default values, enables the keyboard interrupt, then goes into an infinite loop. Program execution in the loop first waits for the specified display delay, starts the measurements sequence, and then calls the appropriate subroutine which calculates and displays the measured values. At the end of each loop the number of sample products accumulated in hardware is checked and adjusted, if necessary. Since this number is controlled by a counter which counts input signal cycles, the number of samples accumulated gives a way to determine the frequency of the input signal. The number of

hardware accumulations must be kept within certain limits. If the number is too small, then the hardware results will come faster than the microcomputer can handle, resulting in lost data. Also, if the number of hardware accumulations is too high, then the hardware accumulator will overflow, again causing a loss of data. This automatic updating at the end of each pass through the loop allows the wattmeter to track changes in signal frequency.

The keyboard monitor routine uses the operator key entries to change the state of the wattmeter; then, depending on the current state, sends the key entries to the appropriate subroutines to change the wattmeter operating parameters. The remaining Pascal routines include the display and parameter changing subroutines mentioned above and a display routine to output Pascal floating point numbers in engineering notation.

2.5. Specifications

The following specifications describe the input characteristics of the two channels of the wideband sampling wattmeter and the accuracy of the power measurements.

Input ranges in peak voltage: ± 0.125, ± 0.25, ± 0.5,
± 1.25, ± 2.5, ± 5.

Input impedance: 100 kΩ and 47 pF.

Common mode isolation: 300 pF to ground.

Accuracy of ranges 0.5 to 5,

dc, 10 Hz to 10 kHz: ± 0.1 %,
10 kHz to 100 kHz: ± 0.5 %.

Accuracy of ranges 0.1 and 0.2,

dc, 10 Hz to 10 kHz: ± 0.2 %,
10 kHz to 100 kHz: ± 1 %.

3. OPERATING THE SAMPLING WATTMETER

3.1. Keypad Control and Display

The operator interface of the NBS wideband sampling wattmeter consists of a 32-character display and two 12 button keypads each arranged as a three by four matrix. Figures 3.1.1 and 3.1.2 show the keypads and display, respectively.

7	8	9
4	5	6
1	2	3
Reset	0	.

(blank)	↑	Home
(blank)	↓	Run
(blank)	←	F↑
ER	→	F↓

Figure 3.1.1 Control Keypads

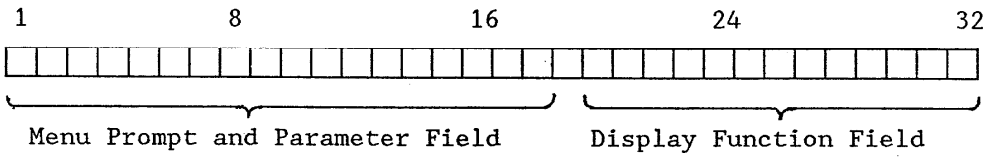


Figure 3.1.2 32-Character Alphanumeric Display

Rather than having an individual key select each function that the operator wants to examine or change, the sampling wattmeter uses a menu approach with direction or arrow keys to step through the menu. The operator interface software controls the sequence of menu items displayed, based on the operator inputs. This software has two modes of operation called "Home" and "Run". The Home mode is used to enter operating parameters and to view a single function. The Run mode is used to display up to four measured functions.

When the system is first powered on, the Reset key (located on the left keypad) must be activated to start the microcomputer. This causes the system to initialize all the various interfaces using the default parameters and to set the operator interface to the Home mode. These default parameters will be described below as each parameter is described. While in the Home mode, the first 18 characters of the display are used to display the menu prompt and the parameter value. Simultaneously, the display field of characters 20 through 32 is used to display one of the measured functions. The first field is called the parameter setting field and the second is called the display function field.

Table 3.1.1 shows the sequence of prompts that appear on the display in response to the arrow keys. The up and down arrow keys (↑, ↓) move vertically through the prompts and the left and right arrow keys (←, →) move horizontally between the three columns. The arrow keys are located in the center column of the right hand keypad. Within the software is a menu pointer called "State" that moves through the menu in response to key inputs. The value of this variable "State" determines what is shown on the display and the response to the next key entry. When the Home mode is first entered, either by pressing the Reset key or the Home key, the menu pointer is set to the "Top of List" location, so that the parameter-setting field will display "Top of List". From here successive activations of the down arrow will move the menu pointer to Sample Pulse; Synchronization, Integration Period, Scale Factors, Display, and finally DMA Parameters, with the corresponding message displayed in the function display field. Another activation of the down arrow brings the pointer back up to Sample Pulse; thus, the list is actually a circular list. The up arrow can be used to move along the loop in the reverse direction. From location Sample Pulse, pressing the right arrow moves the pointer to Sample Freq, and the display shows "Smp Frq = 300 kHz", where 300 kHz is the default value for the sample rate. Activation of the down arrow gives displays of "Pls Wdt = 1000 ns", "Conv Dly = 1 ns", and back to "Smp Frq = 300 kHz", so this is also a circular list. To move to the Multiplexer, which selects the signal which controls the summation intervals requires pressing a left arrow, down arrow, and right arrow with successive displays being "Sample Pulse", "Synchronization", and finally, "Mplx = 4.688 kHz". Movement around the rest of the menu is accomplished in a similar manner. From each of the subloops a right arrow moves the pointer into the data field which enables changing of the parameters. However, before these operations are described, it will be useful to examine the operation of the display function field, since observing the values of functions in this field provides useful feedback while setting many of the system parameters.

Table 3.1.1 Parameter Selection Menu and Default Values

Top of List		
Sample Pulse	Sample Freq.	300 kHz
	Pulse Width	1.00 μs
	Conv Cmd Delay	0 ns
Synchronization	Multiplexer	4.68 kHz
	Trigger Level	2.0 V
	Trigger Delay	1 sample
Integration	Lock Mode	
	Integration Per.	16 cycle
Scale Factor	Shunt Res.	1.0
	Ch 1 Gain	1.0
	Ch 2 Gain	1.0
	Approx Freq.	1000
Display	Display Delay	1000 ms
	Display 1	Power
	Display 2	Number of Samples
	Display 3	Frequency
	Display 4	Period
DMA Parameter	DMA Trigger	TC
	DMA Trancation	None
	DMA Ave	(Blank)

The two keys labeled F↑ and F↓ are used to control the display function field. This field displays one of the 12 measurement values, or a blank display, see table 3.1.2. Using the F↑ and F↓ keys moves the function pointer in software much the same as the menu pointer described above. Table 3.1.2 represents the sequence of functions that will be calculated and the code letters which will appear on the display to identify the numeric data. Table 3.1.3 shows the codes as they appear on the display and the default values assigned at reset.

An appropriate measurement function can be displayed while the parameter values are changed. For example, while setting the summation interval of the sampling wattmeter, which is done in terms of cycles of the incoming signal, the number of samples that are taken in this period can be displayed. To reduce the effects of quantization and other types of noise, the number of samples should be high without introducing excessive measurement delay. Also, to take full advantage of the DMA memory for measuring the average and rms value of each channel, the number of samples should be set to greater than 4096 samples.

Now back to how the parameter values are changed. The parameter values are set using one of three methods. Some parameters have a small fixed set of parameter values and they are set by stepping through the fixed sequence of values. For example, the synchronization source can be set to one of four fixed frequencies or to either input channel. With each input channel signal a filter can be included or not. Thus, this parameter is set by stepping through the eight possible settings. Secondly, some parameters which require a numeric entry, are set by incrementing or decrementing digits of these numbers. The summation interval is set using this method. The number can be incremented or decremented in the units, tens, hundreds, thousands, or ten thousands place. The final method used to set parameters is to enter the various scale factors. In this case, each entry requires the number to be entered using the left numeric keypad. The exponent of the entered number can be changed using the up and down arrow keys. Thus, for example, to enter a value of one milliohm for the shunt resistance, the operator selects the shunt scale factor display location using the arrow keys. Pressing the right arrow key blanks the numeric field and allows numeric entry. Enter 1.0 on the numeric keys, then press the down arrow once. This last step reduces the exponent by a factor of 1000 so the display shows 1.0 m, which is the desired value of one milliohm.

3.2. Parameter Ranges

The setting and significance of each of the sampling wattmeter operating parameters is described in this section:

Sample Rate - A frequency which is adjustable by powers of 2 from 2.34375 kHz to 300.0 kHz. The value of this parameter determines the sampling rate for the wattmeter. Note that when measuring input signals below 80 Hz the sampling rate must be less than 300 kHz to prevent overflow of the hardware accumulator.

Pulse Width - A time interval which is adjustable from 0.333 μs to 2.666 μs by

Table 3.1.2 Function Displays

Function	Display Code
Nothing	(Blank Display)
Power	P=
Frequency	F=
Number of Samples	SN=
Period	Pe=
Average of Channel 1	Av1=
RMS of Channel 1	Rm1=
Average of Channel 2	Av2=
RMS of Channel 2	Rm2=
Decimal Count	Dec2=
Trigger Count	TrC=
Channel 1 Correlation Count	C1C=
Channel 2 Correlation Count	C2C=

Table 3.1.3 Display Codes

Top of List

SAMPLE PULSE	Smp Frq = 300.0 kHz Pls Wdt = 1000.0 ns Conv Dly = 0 ns
SYNCHRONIZATION	Mplx = 4.688 kHz Trg L = 0.0 mV Trg Dly = 1 sm
INTEGRATION PERIOD	LOCK MODE - NIY (Not Implemented Yet) Int Per = 32 cy
SCALE FACTORS	ShuntR = 1.000000 Ch1 SF = 2.441406 m Ch2 SF = 2.441406 m Approx F = 1.000000k
DISPLAY	Disp D = 1000 ms Disp1 = Power Disp2 = Smp Num Disp3 = Freq Disp4 = Period
DMA PARAMETERS	DMA Trg S = TC DMA Trunc = none DMA Ave

steps of 0.333 μs. This parameter sets the hold time for the track-and-hold unit. Note that for the present input amplifier/data converter modules this value must be set at 1.0 μs or greater.

Converter Delay - A time interval which is adjustable from -60 to 195 ns by steps of 1 ns. This parameter adjusts the differential time delay between the two input channels and is used to compensate for both internal and external differential time delays. Table 6.1.1 gives the recommended settings for the present input amplifier/data converter modules.

Multiplex Source - This parameter selects one of eight synchronization signals. Four are internally generated fixed frequencies of 2.344 kHz, 4.688 kHz, 18.75 kHz, or 75.00 kHz. The other four are one of the input channel signals with a low pass filter of either 1.5 kHz or 15 kHz.

Trigger Level - A voltage level which is adjustable from -5.000 to 4.961 V by steps of either 625 mV or 39.06 mV. This parameter sets the reference level which the comparator uses to trigger on the selected synchronization signal. For internal fixed frequencies this level is automatically set to about 2.0 V to be compatible with the internal TTL levels.

Trigger Delay - An adjustable delay from 1 to 65,535 samples in steps of 1 sample. This parameter sets a delay between the trigger event and the start of the summation interval. This feature allows the trigger event to be placed at a reliable, rapidly-rising part of the input signal and, yet, to start and stop the summation interval at another phase of the input signal.

Lock Mode - A planned feature that has not been implemented. The hardware was developed but the software was not. The presently implemented approach for locking to the input signal using the trigger circuit has worked so well with all signals that no need exists for implementing an alternate approach.

Intergration Period - An adjustable interval from 2 to about 99,000 cycles of the incoming signal. This parameter sets the summation interval for calculating the power in terms of the number of cycles of the incoming signal. The values are selected by incrementing or decrementing the units, tens, hundreds, thousands, or ten thousands place. The actual increment will depend on the resolution allowed by the hardware. Since the number of samples which can be accumulated in hardware is limited, as described in section 2.2, long sumation intervals require accumulations in software of hardware results. Thus, the summation interval will be incremented in multiples of the number of cycles over which the hardware accumulator is integrating.

Shunt Resistance - A seven-digit floating-point constant for the shunt resistance, which is used to convert the accumulated product value to true power.

Channel 1 Scale Factor - A seven-digit floating-point constant for the gain of channel one, which is used to change the results from the A/D converter to physical units such as volts or amperes. The default value represents an input range of $\pm$ 5 V full scale.

Channel 2 Scale Factor - Same as above for channel two.

Approximate Frequency - Unused parameter.

Display Delay - An adjustable delay from 1 to 32,767 ms in steps of 1 ms, which is used to specify a delay in the update rate for the display. This allows easy reading of the function values while using short summation intervals.

Display 1 to 4 - These four parameters specify the four functions that will be displayed in the Run mode. The display will show two quantities at a time, either the functions specified by Display 1 and 2 or those specified by Display 3 and 4. The F↑ or F↓ keys will toggle between these two choices.

DMA Trigger Source - The selection of this parameter value determines which of two signals is used to start and stop the summation interval and the DMA module. TC coincides with the sample following the trigger event and FC is delayed from TC by the number of samples specified by the trigger delay parameter.

DMA Truncation Correction Source - The selection of a value for this parameter determines the method used to measure the fraction of a sample needed to make the truncation correction calculation. The choices for this parameter are "no correction", calculate correction from the trigger count, or do a correlation-like fit to data in the DMA memory using either channel one data or channel two data.

DMA Ave - Unused parameter.

4. HARDWARE

4.1. Amplifier/Data Converter Module

The amplifier/data converter unit provides the necessary voltage gain and impedance transformation of input signals for conversion within the unit into a 12-bit, parallel binary code. A front-panel range switch permits selection of full-scale input range levels from 125 mV to 5 V. The input impedance is 100 kΩ in parallel with 47 pF. The two channels of the wattmeter employ identical amplifier/data converter units which are each housed in a plug-in module assembly, thereby permitting easy replacement with updated versions. Generally in power measurement applications, the signal levels in each channel may differ significantly in amplitude. In other words, the "potential" channel may be several hundred volts and the "current" channel may be in the millivolt range from the voltage drop across a resistive current shunt. These wide differences in amplitude for each channel generally require attenuation for the "potential" channel and amplification for the "current" channel. However, the decision was made to design the instrument as a general purpose, dual channel signal analyzer with two identical signal conditioning channels that can be used for other applications as well as power measurement. A further advantage of employing identical signal conditioning units for each channel is the inherent minimization of any differential phase or time delay

between channels. For small differential delays, the instrument contains a programmable differential delay compensation system. This compensation system is discussed in section 4.3. For voltage levels above 5 V, the range can be extended up to 500 V for either channel by means of an external attenuator specially designed for use with the amplifier/data converter module.

The amplifier/data converter unit features a floating input common which permits measurement of signals riding on common mode signals referenced to ground as high as 500 V. The floating input circuit is achieved by optical isolators and dc to dc power converters. This floating feature reduces ground loop errors and permits measurement of small voltages from a resistive 4-terminal shunt not referenced to a common ground potential. Outputs are an isolated analog signal used to trigger the summation interval control circuitry and a 12-bit binary number used to compute the power and other measured quantities.

Figure 4.1.1 shows a simplified schematic of the amplifier/data converter unit. For discussion purposes, the unit can be broken into three major parts: the input circuit and amplifier, the analog isolation circuit, and the digital conversion and output isolators. Following the description of these three parts is a section describing common mode voltage considerations.

4.1.1. Input Circuit and Amplifier

The input signal is coupled to the unit through a BNC connector that has its outer shell insulated from the system ground but connected to the floating common of the circuitry. Range selection is executed by switches S1, S2, and S3 each of which is one deck of a ganged wafer switch. When switches S1 and S2 are in the A position the signal is connected directly to the noninverting input of amplifier U1. The amplifier provides three selectable voltage gain levels of 40, 20, or 10 by respective connections of R5, R6, or R7 to the signal common by switch S3. When switches S1 and S2 are in the B position the input signal is attenuated by a factor of 10 by means of a compensated attenuator formed by R3, C5 and R4, C7. With the attenuator switched in, three additional selectable voltage gains of 4, 2, or 1 are available, thus providing a total of six gains ranging from 1 to 40 in a 1, 2, 4 sequence. Since the track-and-hold amplifier has a gain of one, and the analog-to-digital converter requires a 5-V level, the full scale input voltage for each range is as follows: 0.125, 0.25, 0.5, 1.25, 2.5, 5 V.

The attenuator is compensated by capacitors C5 and C7 and the frequency response is adjusted by trimmer capacitor C5. However, the practicality of obtaining a frequency response for the attenuator that is flat to better than 0.025 percent out to 100 kHz with a resistance of 1 MΩ requires high quality resistors and capacitors with dissipation factors in the order of 10^{-4}. The greatest errors or aberrations in the frequency response for this type of attenuator occur around the "crossover" frequency (i.e., the frequency where the capacitative reactance equals the parallel resistance of each arm of the attenuator). To obtain the desired flatness in frequency response around 10 kHz, the "crossover" frequency was designed slightly above 100 kHz. A "crossover" frequency in this region dictates an attenuator resistance of about 100 kΩ (i.e., R3 = 90 kΩ and R4 = 10 kΩ) because of restrictions imposed

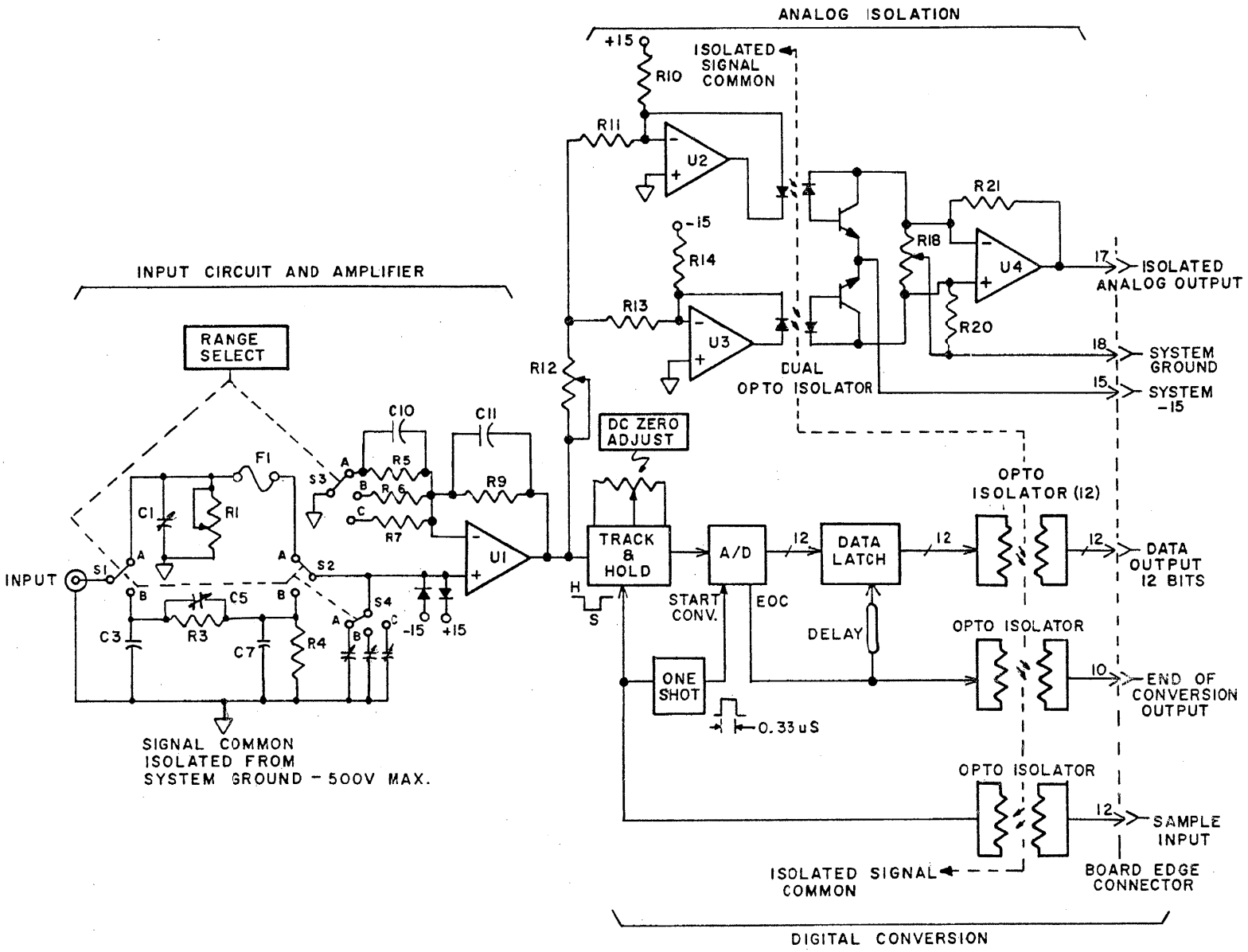


Figure 4.1.1 Simplified Schematic of Amplifier/Data Converter

by practical values of capacitance for C5. Because the input impedance of amplifier U1 contributes primarily a capacitive load, which changes slightly for different gain settings of the attenuator, additional compensation is provided by three trimmer capacitors which are switched in synchronism with S3 by S4. Three capacitors are adjusted to obtain the best overall flat frequency response for all gain settings of the amplifier. When the range switch is set so that the input is connected directly to the amplifier, a fuse in conjunction with two clamping diodes protect the amplifier against an inadvertent overload voltage. Capacitor C1 and R1 are adjusted to obtain the same input impedance when switches S1 and S2 are in either the A or B position.

4.1.2. Analog Isolation

The purpose of the analog isolation circuit is to provide a trigger signal from the floating output of U1 that is referenced to the system ground. Since very high linearity are not required for the trigger signal, optically coupled isolators provide a satisfactory means of direct coupling the floating analog input signal to an output port referenced to the system ground. The analog isolation amplifier uses a matched pair of optically coupled isolators in a differential connection. This system operates on the principle that at a given operating point a gain increment in one optically coupled isolator is approximately balanced by a gain decrement in the second opto-isolator. This technique tends to reduce the total linearity error to a few percent. Amplifiers U2 and U3 are used in a voltage-to-current configuration to forward bias each light-emitting diode at an operating point of 3 mA by means of the 15-V supply and resistors R10 and R14. The signal current into each diode is controlled by resistors R11 and R13, and since the diodes are connected in opposition, the signal causes current to increase in one diode and decrease in the other. The photodetectors of each isolator are connected to U4 which combines the difference in detected currents to produce a ground referenced output signal that is a replica of the input signal (from the output of U1). Resistor R12 is used to trim the overall voltage gain to one. Resistor R18 is adjusted to produce a zero output when the output of U1 is zero.

There are two uses for this analog signal: first, a system-ground-referenced trigger signal is required at the frequency counter board to control the summation interval. A second use is for monitoring signals into the track-and-hold amplifier. Monitoring signals at this point provide the convenience of using a single-ended ground-referenced instrument to observe a signal riding on a common mode voltage and enables a quick means of checking whether a particular signal is within the dynamic range of the A/D converter.

4.1.3. Digital Converter and Output Isolators

The digital conversion circuitry converts the analog signal at the output of U1 into a 12-bit, two's complement, binary code. The A/D converter is preceded by a track-and-hold module which upon command makes a fast "acquisition" of the varying analog signal and "holds" this signal level for the duration of the conversion process. The conversion process is started by a sample input pulse which is coupled through an opto-isolator from the

frequency counter board. The sample input pulse has a minimum duration of 1 μs which is the maximum acquisition time of the track-and-hold module. At the end of the sample input pulse the track-and-hold module "holds" the analog signal. The conversion process in the A/D converter is delayed by approximately 0.3 μs to ensure that switching transients from the track-and-hold have settled. The A/D converter requires 2 μs to convert the "held" analog signal to a 12-bit word. When the conversion is completed, the A/D converter sends an End of Conversion (EOC) signal which after a short delay causes the data from the A/D converter to be fed to the data latch. The EOC signal is also coupled through an opto-isolator to the multiplier-accumulator to indicate that the data is ready. The total delay required for the 1 μs acquisition time, the 0.3 μs settling time, and the 2 μs conversion time limit the sampling frequency to 300 kHz.

4.1.4. Common Mode Voltage Considerations

Significant measurement errors can arise from undesired conversion of common mode signals to normal mode signals. Common mode voltages often appear in noisy environments as potentials between two reference ground terminals because of ground loops. Also, common mode voltages will be present at the potential terminals of a 4-terminal resistance shunt that is used in a power measurement setup.

An ideal, floating input, signal conditioning circuit that is perfectly isolated from earth ground (i.e., infinite resistance and zero capacitance) is devoid of any measurement errors from common mode signals. However, a truly infinite isolation impedance is impossible to achieve, and therefore, depending on the applications, the finite isolation impedance, which results from a practical system, must be accommodated. Figure 4.1.2 shows the floating measurement system of the Amplifier/Data Converter Unit. The circuit shows an isolation capacitance (C_i) from the signal common to ground of about 300 pF. Although not infinite, the isolation resistance in this system can be neglected, and, as will be shown, it is the isolation capacitance which causes the greatest error from high frequency common mode voltages. The bulk of the isolation capacitance is contributed by the dc-to-dc power converters and less than 20 pF by the 16 opto-isolators. As shown in figure 4.1.2 the presence of a common mode voltage V_{cm} causes a common mode current I_{cm} to flow through the low-side or signal common lead to ground through the isolation capacitance C_i. This current will produce a voltage drop across the impedance of the signal lead R_{sl} which causes an error in the measurement of the normal mode voltage V_{nm}. Although there is a component of common mode current that flows through the high-side of the signal lead, it can be neglected because the normal mode input impedance is high compared to R_{sl}. Decreasing the normal mode input impedance tends to improve the common mode rejection by equalizing the currents in both signal leads, but the penalty is an increased normal mode loading error. The error voltage developed across R_{sl} from figure 4.1.2 is

$$E_{sl} = \frac{(V_{cm})\ (R_{sl})}{R_{cm} + R_{sl} + \frac{1}{jwC_i}} \qquad (10)$$

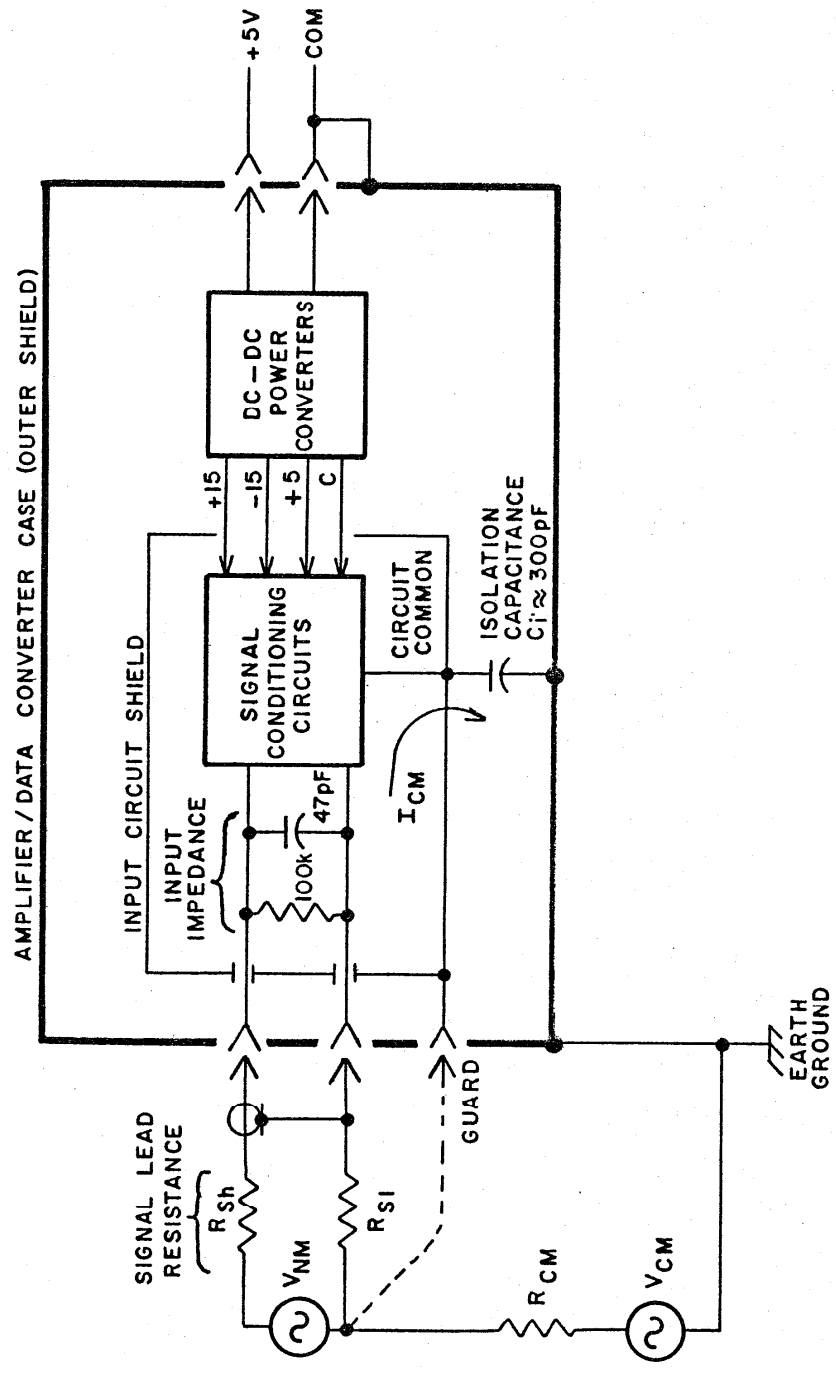


Figure 4.1.2 Floating Measurement System Showing Effect of Common-Mode Signal

and for the case where $(R_{cm} + R_{sl}) \ll \frac{1}{jwC_i}$, equation 10 becomes

$$E_{sl} = V_{cm} R_{sl} jwC_i \quad . \tag{11}$$

To gain an appreciation of the error that can be caused by the frequency dependent common mode voltage, consider the worst case where the amplifier is set for maximum gain (125 mV range) which will result in an input quantization resolution of 60 μV. This means that when a common mode error voltage of at least 60 μV is developed across R_{sl} a common mode error of one least significant bit (1 LSB) will result.

For example, a 500-V peak, 60-Hz common mode voltage with a signal lead resistance of 0.1 Ω (including connector contact resistance) will result in an error voltage of about 5 μV which is well below the quantization resolution of 60 μV; hence, the error can be completely neglected. On the other hand, it is the high frequency common mode signals and noise which become the most troublesome. At higher frequencies the signal lead impedance becomes complex and no longer can be considered all resistive. Thus, for example a 1-V, 1-MHz common mode signal with a 0.1-Ω signal lead resistance will produce an error voltage of 188 μV which is a common mode voltage error equivalent to 3 LSB's.

The errors caused by high frequency common mode voltages can be considerably reduced by connecting a low impedance conductor (i.e., a flat braided wire) from the low-side of the signal source to the guard terminal as shown in figure 4.1.2. If the impedance of this additional circuit is sufficiently low, the common mode current will favor this path, thus, reducing the common mode error voltage drop across R_{sl} . Measurements on the effectiveness of this technique have shown that the error from a 400-V, 200-kHz common mode signal applied at the low-side of the far end of a 1-m 50-Ω coax cable terminated in 50 Ω can be reduced by a factor of 50 with the use of a bypass 2-cm wide braided cable connected from the end of the signal cable to the guard terminal. It is recommended that, when measurements are being made in the presence of high frequency common mode signals, a low impedance conductor be connected to the common mode source and the guard terminal provided on the front of the amplifier/data converter modules.

4.2. Summation Interval Control

4.2.1. Overview

In order to accurately measure the power of an ac signal, the time interval over which the sample data is accumulated must be matched with the period of the input signal. This matching or synchronization function is performed by the summation control circuitry described in this section. The circuitry allows selecting one of eight signals to be used as the synchronizing signal. Through the setting of programmable counters, this circuitry also allows for summation over a variable number of periods, and allows the summation interval to start at a variable phase of the input signal. The time resolution of this matching or synchronization process is the sample interval (reciprocal of the sampling rate). This circuit also counts the number of samples which are taken during each summation interval.

A block diagram of this circuitry is shown on figure 4.2.1, and a timing diagram for the signals is given in figure 4.2.2.

4.2.2. Circuit Description

The synchronization source can be set to one of the internal (crystal-based) fixed frequencies (75 kHz, 18.8 kHz, 4.7 kHz, 2.3 kHz) when pure dc signals are being measured (and for self-test), or set to either channel for ac input signals. In the latter case, the synchronization signal is derived from the optically isolated signal from either channel one or two. The signals from each channel are conditioned with a low-pass filter with a cutoff frequency of either 1.5 kHz or 15 kHz. The selection of the synchronization signal is accomplished by the microcomputer setting the binary output levels for signals F, C1, and T as shown in table 4.2.1. These control signals are applied to a one-of-eight analog multiplexer. The analog output of this device is then converted to a digital signal via a threshold comparator. The other input to this comparator is connected to an 8-bit D/A converter (DAC) which provides programmable reference levels for the triggering of the comparator. The microcomputer programs the output of the 8-bit DAC to set the trigger level to anywhere in the - 5 V to + 5 V range in 39 mV steps.

The output of the threshold comparator goes to the cycle counter. This counter is programmed to function as a rate generator, providing an output pulse (designated FC for frequency count) after a preset number of input pulses. This preset number is set by the microcomputer and determines the number of cycles of the input signal over which summation takes place. The counter is sixteen bits wide and counts from 2 to 65,535 cycles. This counter is one of three in a programmable counter/timer chip. Table 4.2.2 describes the function of the three counters. The output of the cycle counter, normally high, goes low for one period of the input once the preset count has been reached. The high to low transition of this output latches flip-flop A. The output of the flip-flop then serves as a gate to the delay counter. This counter also has a preset count, set by the microcomputer, which signifies the delay in sample command pulses. The delay counter is also sixteen bits wide and counts from one to 65,535 sample pulse commands. This counter works as a retriggerable one-shot down counter. The output of the delay counter TC (Terminal Count), normally high, goes low for one incoming clock period once the preset count is reached. The Terminal Count pulse resets flip-flop A and starts the timing control circuitry which generates two control signals. The first signal (strobe) latches the parallel output of the sample counter into latches accessible to the microcomputer. The second signal (clear) resets the contents of the sample counter for the next summation interval. This latching and clearing operation takes place within one sample interval, so no sample counts are missed.

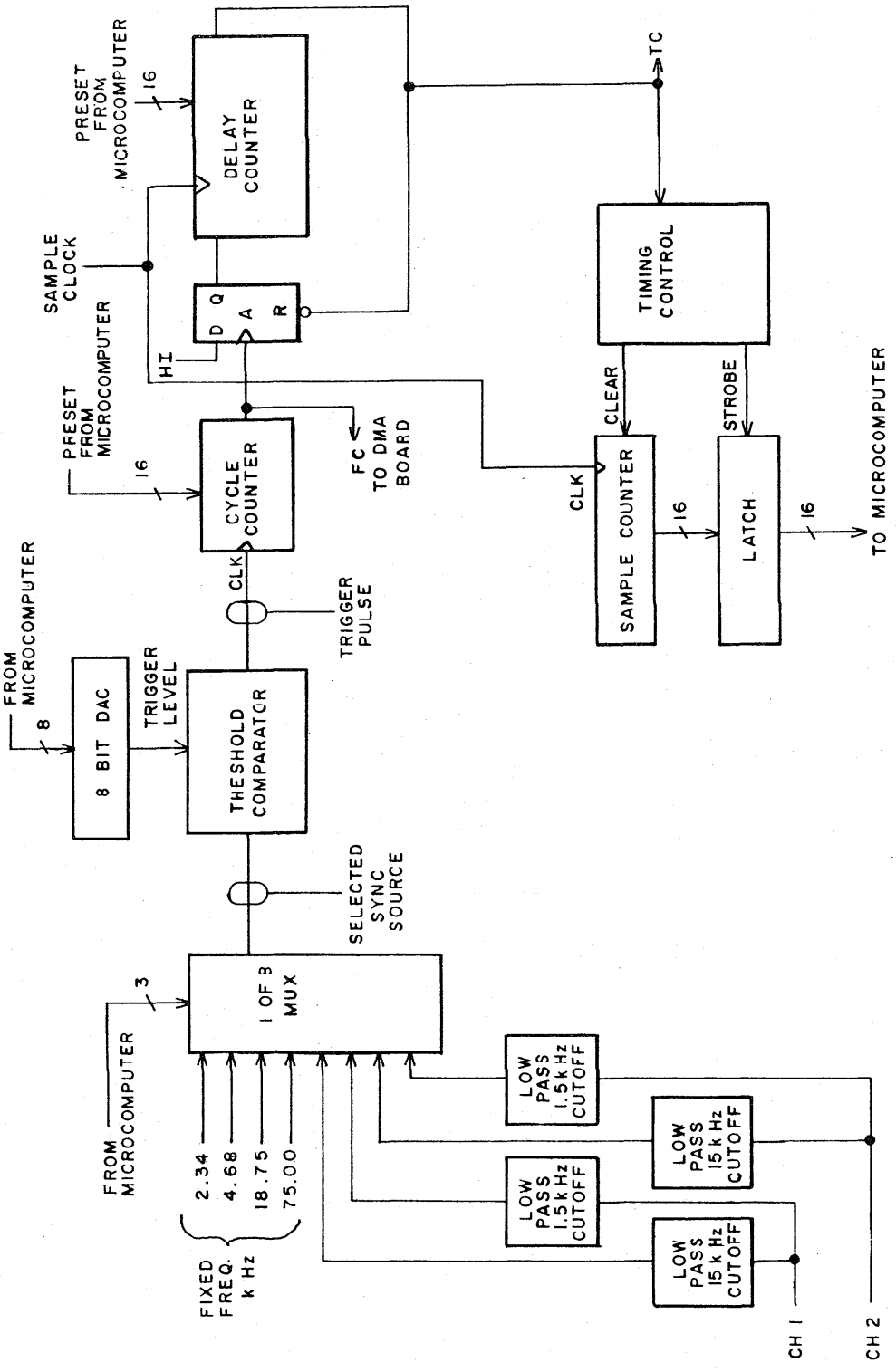


Figure 4.2.1 Summation Interval Control Circuitry

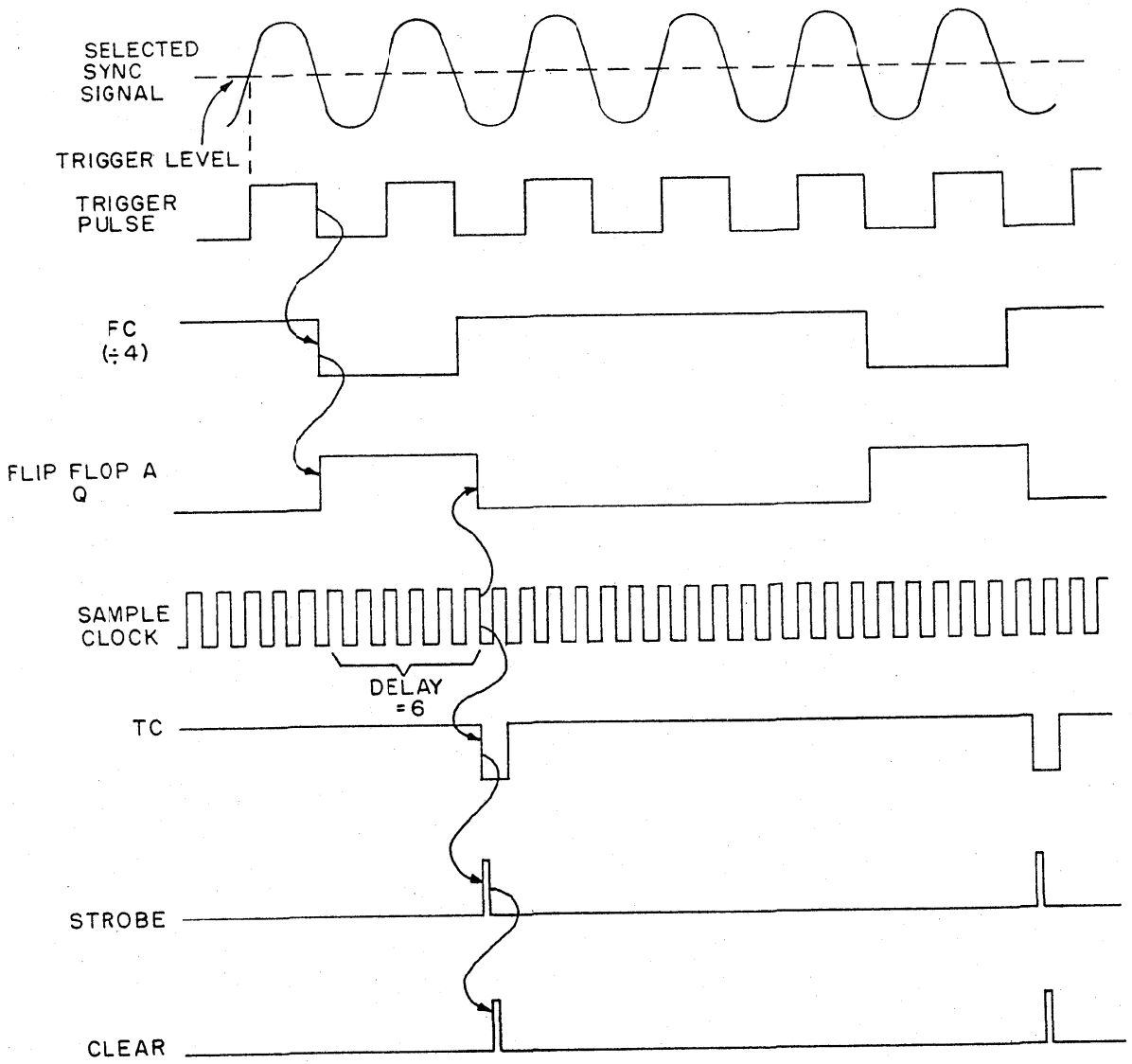


Figure 4.2.2 Timing Diagram for Summation Interval Control

Table 4.2.1 Control Signals of Interval Control Source

T = Test Frequency
C1 = Channel Control
F = Filter Control

T	C1	F	Selected Signal
0	0	0	Channel 2 Low Cutoff
0	0	1	Channel 2 High Cutoff
0	1	0	Channel 1 Low Cutoff
0	1	1	Channel 1 High Cutoff
1	0	0	2.3 kHz
1	0	1	4.7 kHz
1	1	0	18.8 kHz
1	1	1	75 kHz

Table 4.2.2 Programmable Counter Timer Function/Description

Counter	Function	Description
Counter 0	Unused	
Counter 1	Delay Counter	- Retriggerable one shot down count - Counter starts counting on positive going edge of data input - Output (Terminal Count) goes low for one incoming clock period when count is reached - Counts between 1 to 65,535 sample intervals. Sets start of summation interval relative to trigger event
Counter 2	Cycle Counter	- Continuous rate generator - Output low for one period of incoming clock - Counts between 2 to 65,535 cycles of the selection interval control signal.

4.3. Differential Time Delay

4.3.1. Overview

This circuits provides the convert command pules for the A/D converters on the data acquisition boards. The circuit uses programmable time delays to compensate for the differential time delays between the voltage and current channels. In addition, the circuit controls the width of the convert command pulses, thereby controlling the track time (sample input) for the track-and-hold modules. Table 4.3.1 shows the range, steps, and controls for these parameters. The controls are binary level outputs issued by the microcomputer. A complete listing of all controls is given in tables 4.3.2 to 4.3.5. The outputs originate in a 24 bit programmable I/O device, an 8255 chip. This chip is directly accessible to the microcomputer through the Multibus. The chip is configured as three output ports A, B, and C and the individual bit assignments of these outputs are given in table 4.3.6.

Table 4.3.1 Control Signals for Selection of Sample Frequency, Track Time, and Differential Time Delay

Output	Range	Steps	Controls
Sample freq	2.34 k - 300 kHz	powers of 2	F1, F2, F4
Track time	0.33 s - 2.67 s	0.33 s	W1, W2, W4
Diff delay	-60 ns to 195 ns	1 ns (chan 1) 16 ns (chan 2)	L1, L2, L3, L4 L5, L6, L7, L8

Table 4.3.2 Sample Frequency Control

F1	F2	F4	Sample Freq.
0	0	0	300 kHz
1	0	0	150 kHz
0	1	0	75 kHz
1	1	0	3.75 kHz
0	0	1	18.75 kHz
1	0	1	9.38 kHz
0	1	1	4.69 kHz
1	1	1	2.34 kHz

Table 4.3.3 Pulse Width Control

W1	W2	W4	Pulsewidth
0	0	0	0.33 μs
1	0	0	0.67 μs
0	1	0	1.00 μs
1	1	0	1.33 μs
0	0	1	1.67 μs
1	0	1	2.00 μs
0	1	1	2.33 μs
1	1	1	2.67 μs

Table 4.3.4 Channel 1 Delay Timer

Controls (L4, L3, L2, L1)	Delay ns
0000	60
0001	61
0010	62
0011	63
0100	64
0101	65
0110	66
0111	67
1000	68
1001	69
1010	70
1011	71
1100	72
1101	73
1110	74
1111	75

Table 4.3.5 Channel 2 Delay Timer

Controls (L8, L7, L6, L5)	Delay ns
0000	15
0001	31
0010	47
0011	63
0100	79
0101	95
0110	111
0111	127
1000	143
1001	159
1010	175
1011	191
1100	207
1101	223
1110	239
1111	255

Table 4.3.6 Function Assignment for Programmable I/O Chip

bit	Port A	Port B	Port C
7	L8	W4	unused
6	L7	W2	"
5	L6	W1	"
4	L5	unused	"
3	L4	M (lock mode)	"
2	L3	F4	T
1	L2	F2	C1
0	L1	F1	F

4.3.2. Circuit Description

A block diagram of the circuit is shown in figure 4.3.1. The 6-MHz source is composed of an 18-MHz crystal oscillator and a dual J-K flip-flop. The dual flip-flop performs a divide-by-three function reducing the 18-MHz source to 6 MHz. The 6-MHz is then further divided by a decade counter to create a 3-MHz and a 600-kHz signal. The 600-kHz signal is then divided by dual four stage binary counters. The outputs of these binary counters are eight distinct frequencies between 300 kHz and 2.34 kHz. One of the eight outputs is selected by an eight-to-one multiplier as the sampling frequency. The selection is controlled by the microcomputer by setting the binary levels for F4, F2, F1, as shown in table 4.3.2. To control the pulse width of the sampling frequency pulses, the original sampling frequency is "ANDed" with the complement of a delayed version of the same sampling frequency. The result creates a pulse width equal to the delay time. This pulse width is used to control the track time of the track-and-hold circuit in the input data converter circuitry. The delayed version of the sampling frequency is developed using an eight stage shift register and an eight-to-one multiplexer. The shift register samples the original sampling frequency at a 3-MHz rate so that the delay between the outputs of successive stages of the shift register is 0.33 μs. Thus, the shift register outputs are essentially eight increasingly delayed versions of the original sampling frequency with the delays in increments of 0.33 μs. Selection of one of the eight delayed versions is done with the eight-to-one multiplexer under microcomputer control. The microcomputer sets the binary levels for the W4, W2, W1 controls, as shown in table 4.3.3, for the desired width. The output of the AND gate then contains the desired sampling rate and pulse width. This signal is synchronized with the 6-MHz oscillator output via a flip-flop to reduce any time jitter introduced by the counters.

The pulse output of the flip-flop then becomes the convert command signal for the A/D converters. This signal is routed to two programmable time delay lines to create the individual convert commands for channel one and channel two. The delay lines are controlled by the microcomputer by setting the binary levels of L1, L2, L3, L4 and L5, L6, L7, L8 for channels one and two, respectively. The time delay for channel one is variable from 60 to 75 ns in

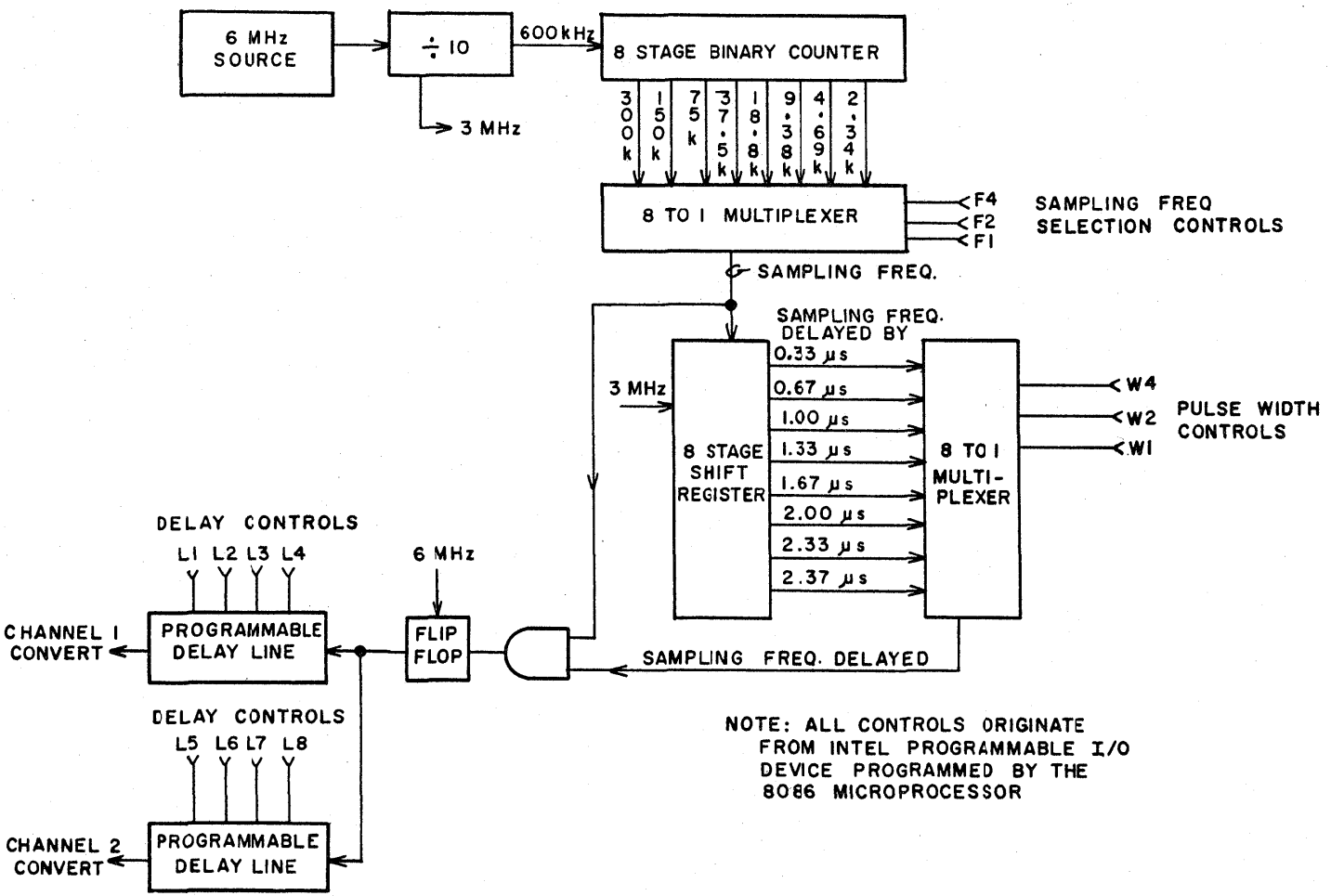


Figure 4.3.1 Sample Convert Command Pulse Generator Circuit with Programmable Differential Time Delay

1 ns steps. The time delay for channel two is variable from 15 to 255 ns in 16 ns steps. Thus, the relative time delay between the two channels is variable from -60 ns to 195 ns in 1 ns steps. This variable time delay is used to compensate for the difference in time delays introduced by probes, cables, current shunts, and amplifiers in the two channels.

4.4. High Speed Numerical Integrator

4.4.1. Overview

The high speed numerical integrator is used to make real time power measurements. This circuit multiplies the output of the voltage A/D converter and current A/D converter and adds the product to the contents of an accumulator. This process continues for a number of accumulations N. In this manner, the circuit performs the fundamental computational chore for a power measurement

$$P_{avg} = \frac{1}{N} \sum_{i=1}^{N} V_i \, I_i \, , \tag{12}$$

where the stored result in the accumulator is the sum of N V_i-I_i-product terms.

From a hardware point of view, the numerical integrator can be separated into two parts: the multiplier-accumulator section and the high speed sequencer section (see block diagram of figure 4.4.1). Each section is covered in detail in the material that follows. The operation of the numerical integrator is controlled by the terminal count (TC) signal from the frequency counter circuit board. The sequences of operation necessary to perform a numerical integration are as follows. The numerical integrator performs successive multiplications and additions of each set of A/D conversion results. Shortly after the TC pulse is applied, the product of the last A/D output pair is added. Then the accumulated result is placed into latches accessible to the microcomputer. Coincident with this action an interrupt signal is sent to the microcomputer. In addition, the multiplier-accumulator is cleared in preparation for the next series of numerical integrations.

4.4.2. Multiplier-Accumulator

A block diagram of the multiplier-accumulator is shown in figure 4.4.2. The key component is the high speed LSI device TDC1010J shown bordered by the dashed lines in the block diagram. This monolithic device contains a 16 by 16-bit multiplier, a 35-bit accumulator, and an internal storage latch. The 16-bit X_{in} and Y_{in} inputs are latched into registers R1 and R2 on the rising edge of clock X+Y. The values in R1 and R2 are multiplied, and their product is added to the contents of R3 with the result stored back into R3 on the rising edge of clock P.

The function of R4 is two fold. During initialization, R3 must be preset to all zeros. The PREL control signal latches all zeros into R4 and enables bus transceivers in the multiplier-accumulator so that the output of R4 is

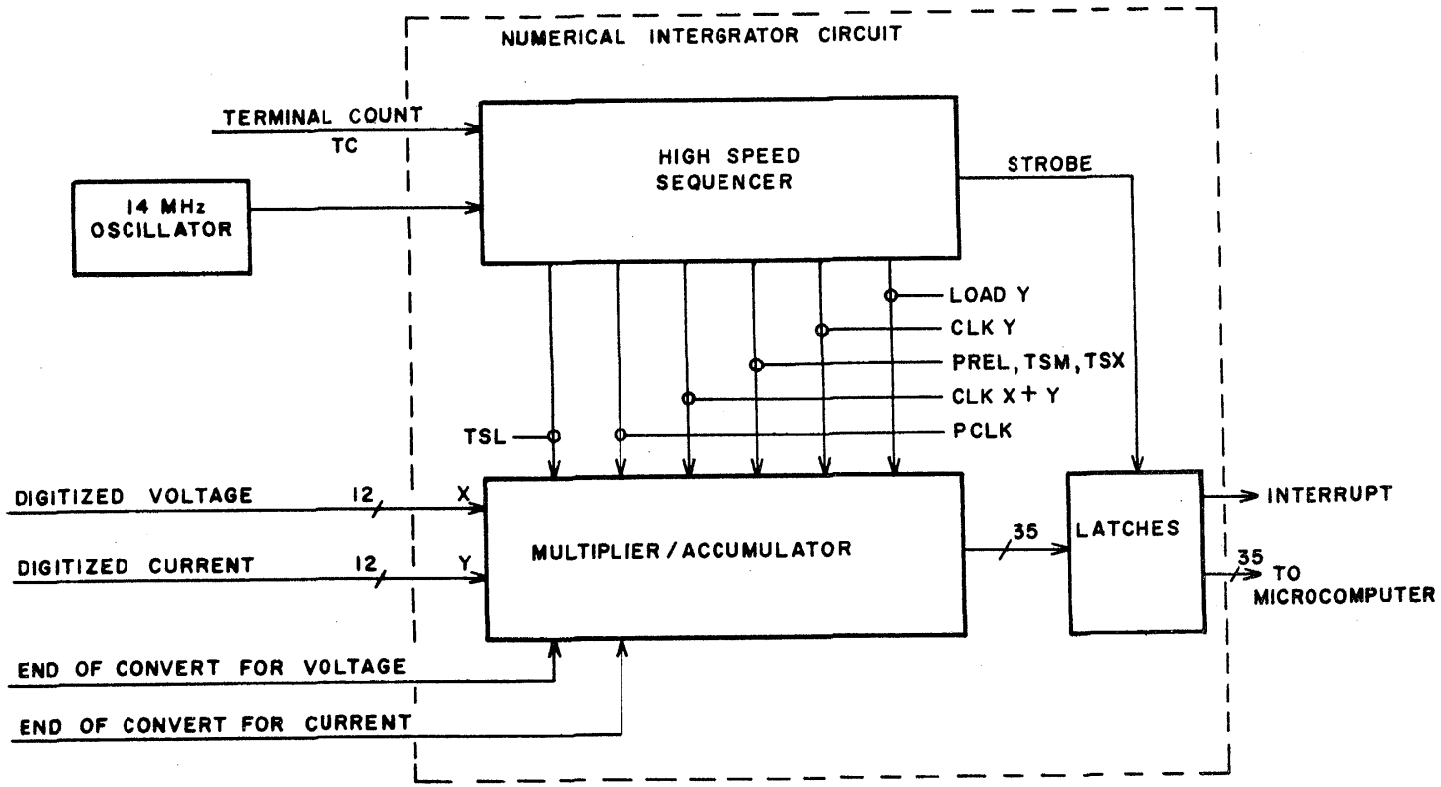


Figure 4.4.1 High Speed Numerical Integrator

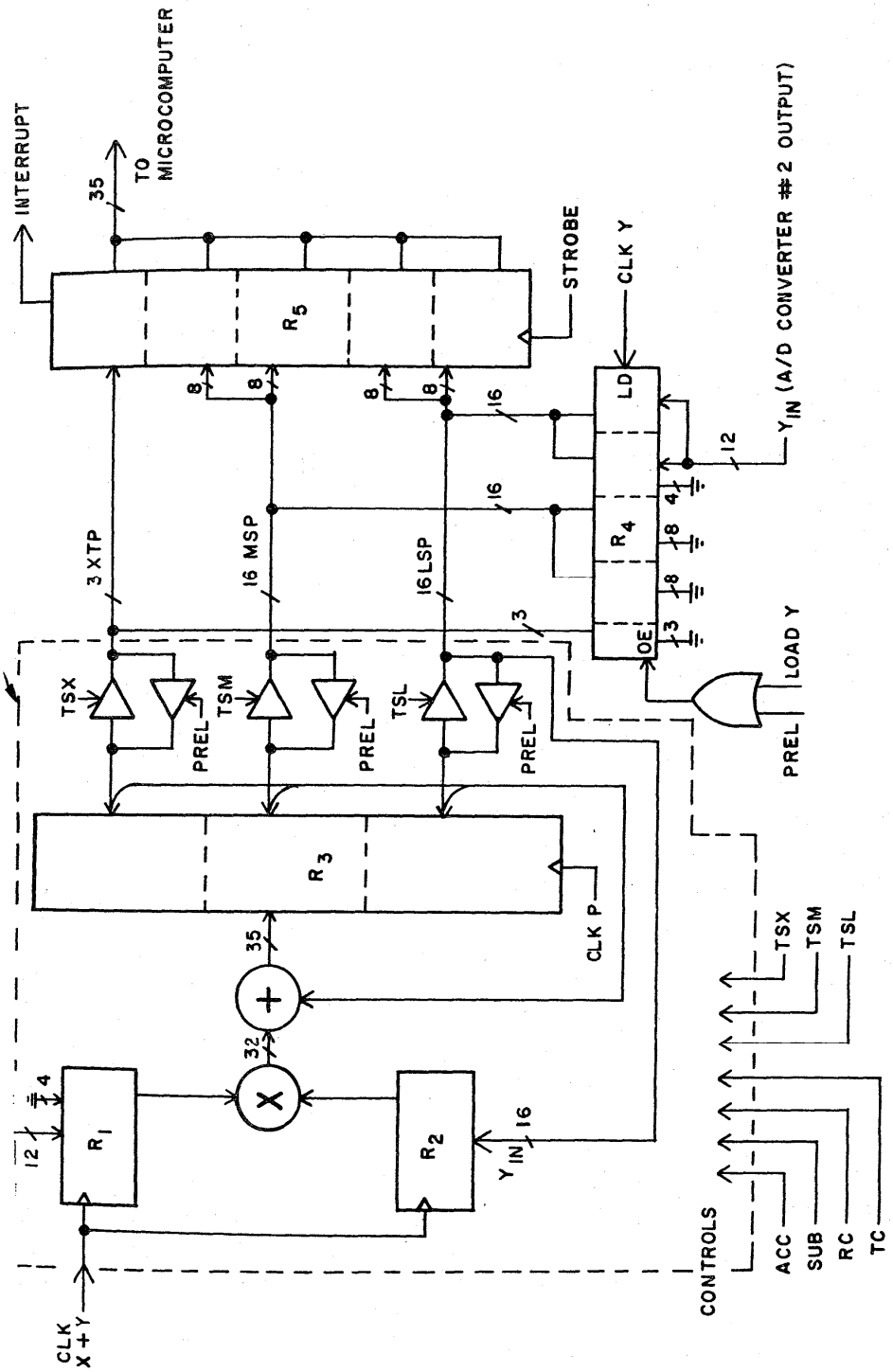


Figure 4.4.2 Multiplier-Accumulator Circuit

entered into R3. With TSX, TSM, and TSL controls set high disabling the output buffer, the contents of R4 are latched into R3 on the rising edge of clock P. The second function of R4 is to hold the value of Y_{in} temporarily for transfer over the multiplexed LSP bus for final latching into R2. This is accomplished by a LOAD Y clock which latches the value of Y_{in} into R4 and disables the output LSP buffer by setting TSL high. The contents of R4 are then latched into R2 with the rising edge of clock X+Y. In addition to latching Y_{in} into R2, X_{in} is also latched into R1 on the rising edge of clock X+Y.

Register R5 provides an interface to the microcomputer. Upon completion of a multiply-accumulation sequence, the results of R3 are latched into R5 when a STROBE signal occurs. Coincident with the completion of STROBE, an interrupt is generated from R5 to the microcomputer signifying that an accumulation product is ready for transfer. Other controls to the TDC1010J are fixed as follows:

ACC	+5 V dc	Enables running sum
SUB	gnd	" " "
RC	gnd	Enables truncation rather than roundoff
TC2	+5 V dc	Enables two's complement number system

4.4.3. High Speed Sequencer

The high speed sequencer operates in two modes, controlled by the level of the TC signal. The function of the two modes is given in table 4.4.1. The high speed sequencer issues seven control signals to the multiplier-accumulator which are listed in table 4.4.2. As shown in figure 4.4.3, the high speed sequencer consists of a 256 by 8-bit ROM, address counter, and timing circuitry. Each of the instructions (LOAD Y, CLK P, etc.) occupies a bit location out of each byte in ROM. A sequence of instructions is performed by reading successive byte patterns out of the ROM. The particular set of instructions executed depends upon the address initially stored into the address counter. The starting address is controlled by the terminal count (TC) signal.

Each instruction takes about 70 ns. The longer program, performed when TC is high, takes nine steps or 630 ns. At the end of each program, a pulse is generated to reset the timing circuitry, thereby disabling the address counter. The steps for the two types of program operations of the high speed numerical integrator are shown in table 4.4.3.

The timing circuitry of the high speed sequencer consists of one-shots and latches. The EOC 1 (end of conversion 1) signal is passed through two one-shots such that the trailing edge preloads the address counter and sets a latch, and the EOC 2 signal sets another latch. The outputs of both latches are "ANDed" to generate a gating control pulse to the address counter clock. This scheme ensures that the sequencer does not begin a program sequence until the outputs of both A/D converters are available.

Table 4.4.1 High Speed Numerical Integrator Sequences

Normal Operation

The data from the A/D converters are multiplied and the product is added to the existing accumulator contents. A total of 2048 full-scale, 12-bit inputs may be multiplied and accumulated before the accumulator can reach an overflow condition.

Terminal Count Operation

The terminal count operation is initiated by the TC pulse from the summation interval control board. After the pulse is received, one more multiplication-accumulation is performed, and the accumulator data is strobed into the latches. An interrupt request to the microcomputer is activated as part of this sequence. After strobing the data latches, the accumulator is zeroed so it is ready for subsequent accumulations.

Table 4.4.2 High Speed Sequencer Signals

Signal	Function
CLK X + Y	Rising edge causes X_{in} and Y_{in}, both 12 bit digitized values, to be latched into registers R1 and R2 respectively.
P CLK	Rising edge causes the accumulator R3 to be loaded with either the contents of the internal adder or the value on the XTP, MSP, LSP buses depending on other controls.
PREL*TSX*TSM	Controls buffer direction on XTP, MSP buses internal to the TDC1010J. Also controls the three state output of register R4. Used during initialization.
TSL	Controls input buffer direction of LSP buses. Used during initialization and during Y_{in} multiplex on LSP bus.
LOAD Y	Controls three-state output of R4 for multiplex of Y_{in} on the LSP bus.
CLK Y	Rising edge causes Y_{in} to be latched into R4.
STROBE	Latches the accumulator results into a buffer register and sends an interrupt to the microcomputer indicating that the accumulator data is available.

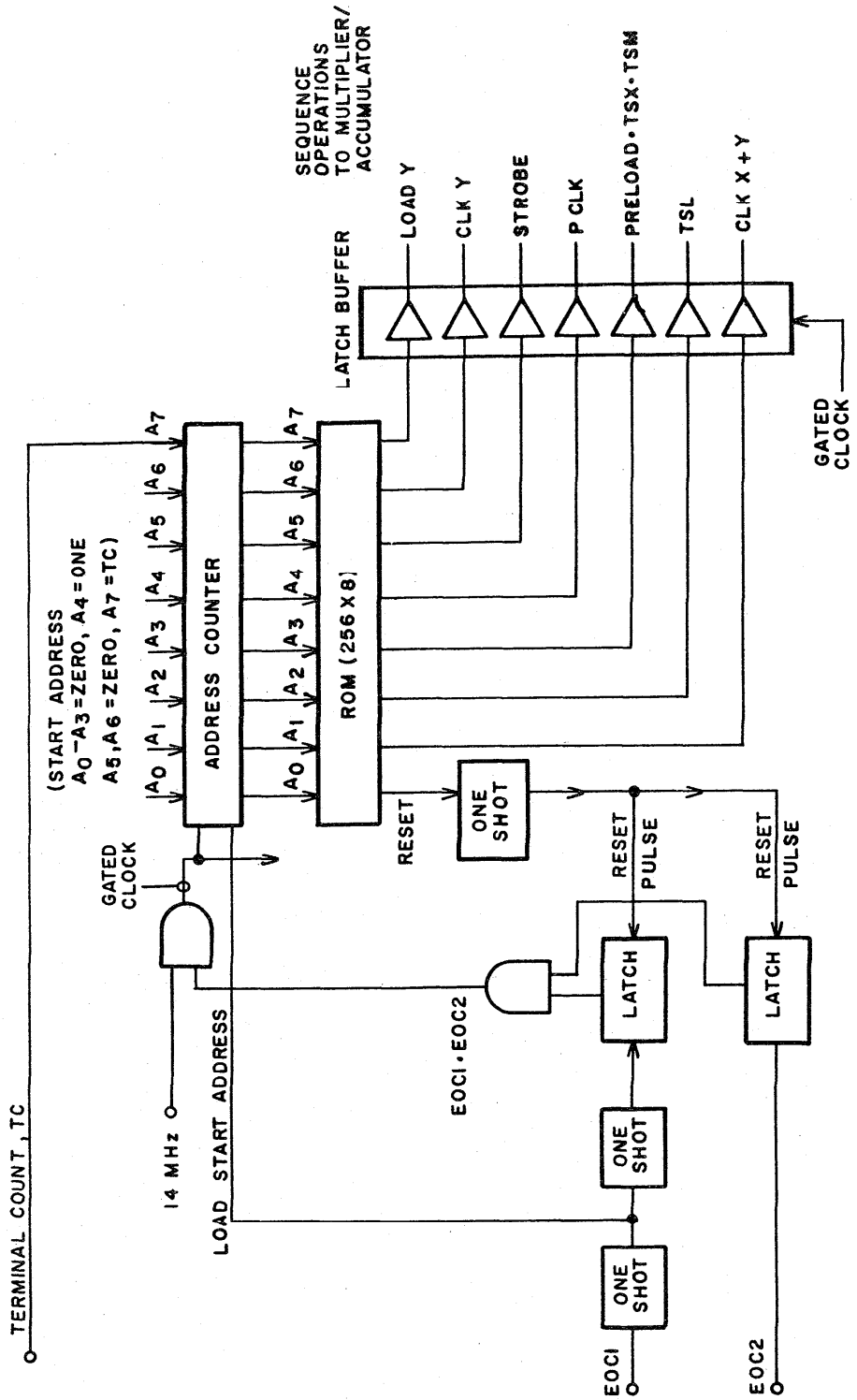


Figure 4.4.3 High Speed Sequencer for Multiplier-Accumulator

Table 4.4.3 High Speed Sequencer Programs

Address	TSL	PREL	P CLK	CLK X+Y	STROBE	CLK Y	LOAD Y	RESET	Action
Normal									
90	0	0	0	0	0	0	0	0	NOP
91	1	0	0	0	0	1	1	0	Load Y, TSL, CLK Y
92	1	0	0	1	0	0	1	0	Load Y, TSL, CLK X+Y
93	0	0	0	0	0	0	0	0	NOP
94	0	0	1	0	0	0	0	0	P CLK
95	0	0	0	0	0	0	0	1	Reset
Terminal Count									
10	0	0	0	0	0	0	0	0	NOP
11	1	0	0	0	0	1	1	0	Load Y, TSL, CLK Y
12	1	0	0	1	0	0	1	0	Load Y, TSL, CLK X+Y
13	0	0	0	0	0	0	0	0	NOP
14	0	0	1	0	0	0	0	0	P CLK
15	0	0	0	0	1	0	0	0	Strobe
16	1	1	0	0	0	0	0	0	PREL, TSL
17	1	1	1	0	0	0	0	0	PREL, TSL, P CLK
18	0	0	0	0	0	0	0	1	Reset

4.5. Direct Memory Access Circuitry

4.5.1. Overview

The purpose of the Direct Memory Access (DMA) board is to provide the wattmeter with a buffer memory capable of storing high-speed data samples. With A/D conversion rates of up to 300 kHz, collection of this data by the microcomputer through normal I/O channels is impossible, and the DMA approach is the only practical solution. Data obtained through the DMA is used in the following ways:

- calculation of truncation error
- selection of proper trigger level
- signal channel offset alignment
- signal channel gain calibration

4.5.2. Circuit Description

A block diagram of the DMA circuit is shown in figure 4.5.1. The circuit has a total capability of 16k bytes. This memory is organized into two 4k by 16-bit memory banks. One bank is dedicated to storage of digitized voltages and the other to storage of digitized currents. Three-way bus transceivers

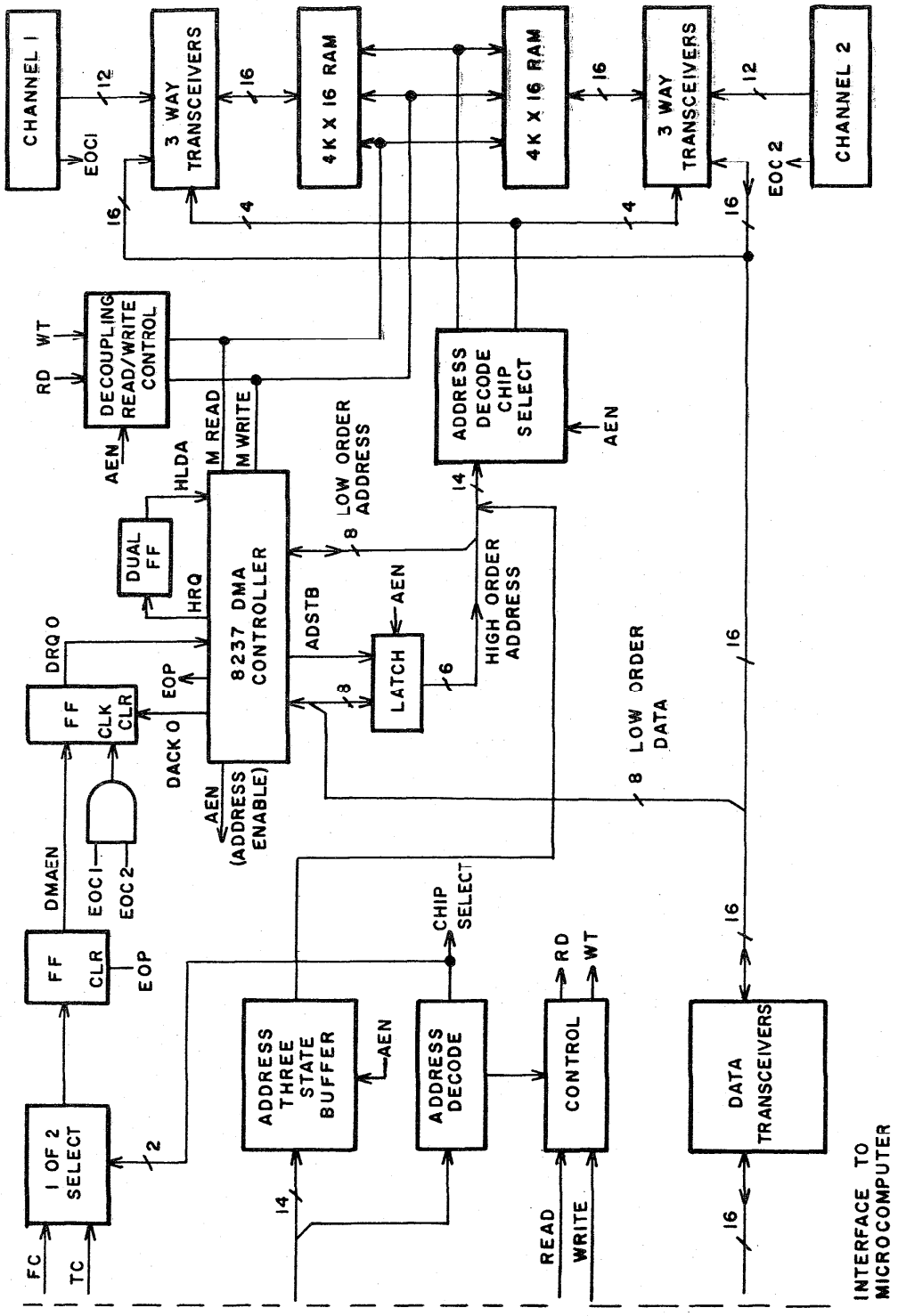


Figure 4.5.1 Direct-Memory-Access Control Circuitry

are used to connect the data bus of a memory bank to either the output of the channel A/D converter or to the system data bus (Multibus). This connection is controlled by the address enable AEN signal from the DMA controller. When AEN is high the DMA memory is connected to the A/D channels, and when it is low the memory is connected to the Multibus. The direction of data flow is determined by the states of the read/write controls. Although the A/D outputs are 12 bits wide, the DMA electronics accommodate 16-bit transfers and storage to allow for future expansion.

The heart of the DMA board is the programmable DMA controller 8237 chip. This chip provides four independently controlled DMA channels, each of which is programmable. Presently, only channel 0 is being used. Channel 0 is programmed by the microcomputer through the Multibus address and data lines. These lines are used to access and write into the programmable, base-address and terminal-count registers inside the controller. The DMA operation is synchronized with the operations of the multiplier-accumulator circuits by using the final count FC or terminal count TC signals (see section 4.2 for an explanation of these signals), originating from the summation interval control circuit. Selection of either FC or TC as a trigger source is under software control. Arrival of the selected signal will set a flip-flop whose output is the DMA enable control DMAEN. This control signal is then clocked into another flip-flop whose output is the DMA request DRQO. The clocking signal is derived by the "ANDing" of the end of conversion signals from the two A/D converters. The presence of both of these signals assures that data is ready from both channels before a DMA request is issued. Activating the DRQO initiates a DMA operation. The controller, programmed to operate in the demand transfer mode, outputs a hold request HRQ which, in typical applications of this controller, is sent to the system microcomputer. This hold request is to ensure that there is no contention for the system address and data lines once DMA has begun. However, in the sampling wattmeter, DMA transfers do not occur over the system bus, but rather locally on the DMA board. Hence, operation of the DMA controller does not interfere with the system microcomputer.

The DMA controller HRQ signal is clocked through a dual flip-flop circuit to delay it, and then routed back to the controller as a hold acknowledge HLDA. The controller then activates the address enable AEN signal which isolates the DMA board from the system bus and configures the memory transceivers for A/D converter to DMA memory transfers. The DMA controller also takes control of the address, read RD, and write WT lines. The lower eight bits of the address are output on the eight-bit address lines by the controller, and the upper eights bit of address are output on the data lines of the controller. The controller activates an address strobe ADSTB to load these upper address bits into a latch. When the AEN signal goes high, it enables the outputs of the latch to be placed onto the upper eight address lines of the DMA board. This operation ensures that the complete address is available to select the desired DMA memory location. During the A/D to memory write sequence, the memory chip select signals for the individual RAMs are decoded such that 32 bits of data (two 16-bit pairs) are stored concurrently. Thus, during DMA operation, the memory effectively looks like 4k by 32 bits of memory, whereas, when the same memory is accessed via the system bus it appears as an 8k by 16-bit memory. The DMA request acknowledge DACKO signal

issued by the controller, in response to the DRQO signal, resets the DMA request flip-flop to get ready for the next DMA request.

When the DMA circuitry has stored the programmed number of samples, the DMA controller issues the end-of-process EOP signal which resets the DMA enable flip-flop and sends an interrupt to the microcomputer. This interrupt signifies to the microcomputer that DMA data is accessible, and the DMA controller must be programmed for the next DMA cycle.

4.6. Microcomputer and Memory

4.6.1. Microcomputer

The microcomputer used in the NBS sampling wattmeter uses a 16-bit microprocessor the 8086. This microprocessor resides on an Intel SDK-86 microcomputer single board computer kit [7]. This kit was interfaced to a Multibus card cage through a Multibus interface constructed on the prototype circuit area. All of the other wattmeter electronics, excluding the amplifier/data converter modules, reside in the Multibus card cage. To increase EPROM memory capacity of the wattmeter beyond the 8k available on the SDK-86, an Intel SBC-464 PROM board [8] was added to the Multibus card cage.

4.6.2. Memory

Memory for the sampling wattmeter is located on the microcomputer board, the DMA electronics board, and the PROM board (SBC-464). Table 4.6.1 shows the memory map for the microcomputer board. Note that the I/O interface chips on the microcomputer are memory mapped. Although only the lower 2k RAM and the parallel port are used for the wattmeter, the other devices are listed to show the memory space that they occupy. Note that the lower 8k of EPROM on the PROM board can not be used since it overlaps the RAM on the microcomputer board. Table 4.6.2 shows the memory map for the PROM board and the DMA memory board.

Table 4.6.1 Microcomputer Memory Map

Location (HEX)	Description
OFFE8-OFFEF	8279 keyboard/display (unused)
OFFFO-OFFF7	8251 serial port (unused)
OFFF7-OFFFF	8255 parallel port (Init Signal)
00000-007FF	RAM 2k
00800-00FFF	RAM 2k (expansion capability)
FEOOO-FFFFF	EPROM 8k

Table 4.6.2 PROM and DMA Memory Map

Location (HEX)	Description	Circuit Board
00000-01FFF	EPROM 8k (not used)	PROM
02000-03FFF	EPROM 8k	PROM
04000-05FFF	EPROM 8k	PROM
06000-07FFF	EPROM 8k	PROM
08000-08077	CONTROL	DMA
0C000-0DFFF	RAM 8k	DMA
0E000-0FFFF	RAM 8k	DMA

4.7. Keyboard and Display

4.7.1. Overview

The keyboard and display are the operator interface for control of the sampling wattmeter. The display is an intelligent, 32-character, alphanumeric fluorescent display. The keyboard is comprised of two three-by-four matrix keypads. All but one key are decoded and accessed by the software. The one nondecoded key is used as an external hard reset to the wattmeter microcomputer. The operation of the keys and the display formats are described in chapter 3. The electrical connections to these devices are described in the next two sections.

4.7.2. Keyboard Electronics

A diagram of the keyboard electronics is shown in figure 4.7.1. It shows a front view of the physical layout of the keypads and their respective codes overlayed.

The keyboard is wired as a single three by eight matrix. Scanning of the keyboard, key debounce, decode, and store are performed by a single device, the programmable keyboard/display interface 8279. This device is programmed for keyboard use by the microcomputer when the wattmeter is reset. For each key entry the device generates an interrupt to the microcomputer to signify to the microcomputer that a key has been depressed and the key code is available. The device has a built-in, eight-character first-in first-out (FIFO) register to prevent loss of data.

4.7.3. Display

The display is an intelligent, 32-character, alphanumeric fluorescent display subsystem (model DE/432 display, manufactured by Digital Electronics). The display electronics has a full ASCII set character generator, horizontal scrolling, and cursor control. The microcomputer, using a parallel interface, is able to read from and write to the device (see figure 4.7.2). To write to the device, the microcomputer begins with a two character sequence; the first character is an escape character, and the following character designates the

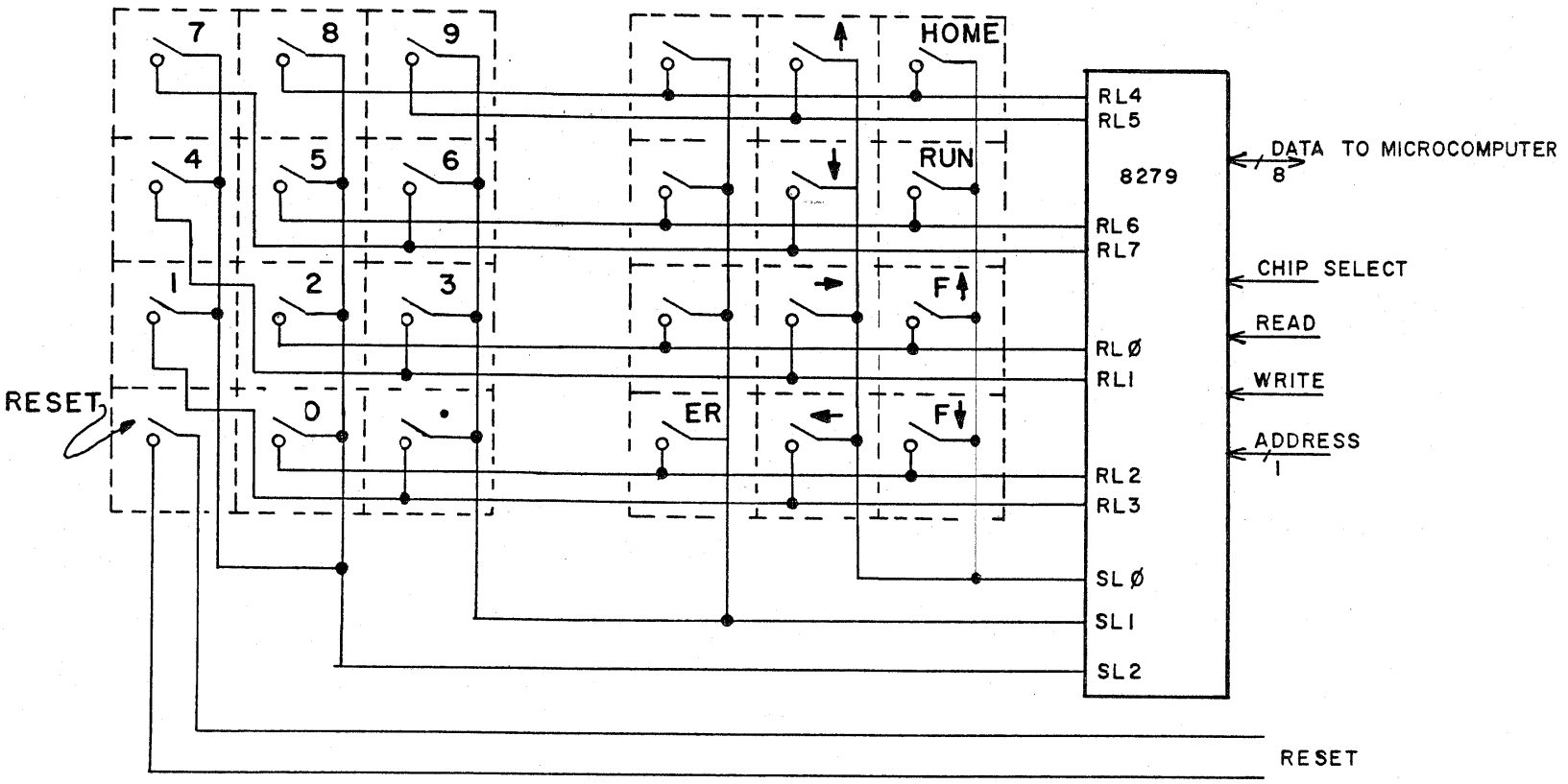


Figure 4.7.1 Keyboard Circuitry

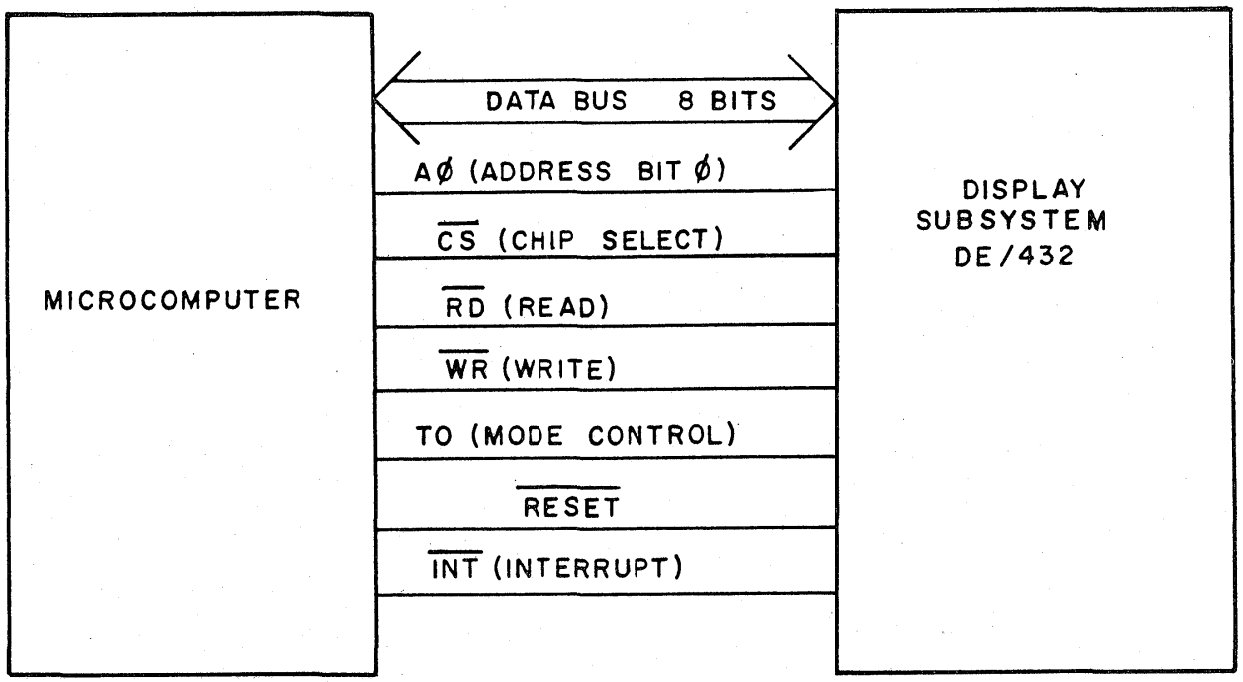


Figure 4.7.2 Display Circuitry

desired position for the cursor. The next data character will overlay the cursor position, and subsequent data characters will fill the display to the right of that position. Reading of data from the display follows a similar procedure.

Before sending any command or data, the microcomputer must check the busy bit (bit 0 of device status register), and only after it is low can the command or data be sent. After a command to read has been sent to the display, the data available flag must be checked (bit 1 of status register), and only after it is high can the data be read.

5. SOFTWARE

The sampling wattmeter software was written in both assembly language and Pascal. The assembly language routines act as an interface between the Pascal software and the wattmeter hardware and are used in two ways. First, the assembly language routines handle the interrupts which occur asynchronously with the execution of the Pascal program. The three principle interrupts result from keyboard activity, from completion of data acquisition by the multiplier-accumulator, and from the completion of a DMA transfer. The other use of assembly language is to interface the Pascal software with hardware registers. Such routines are used to write to the display, and to set the wattmeter operating parameters such as sampling rate and summation interval control.

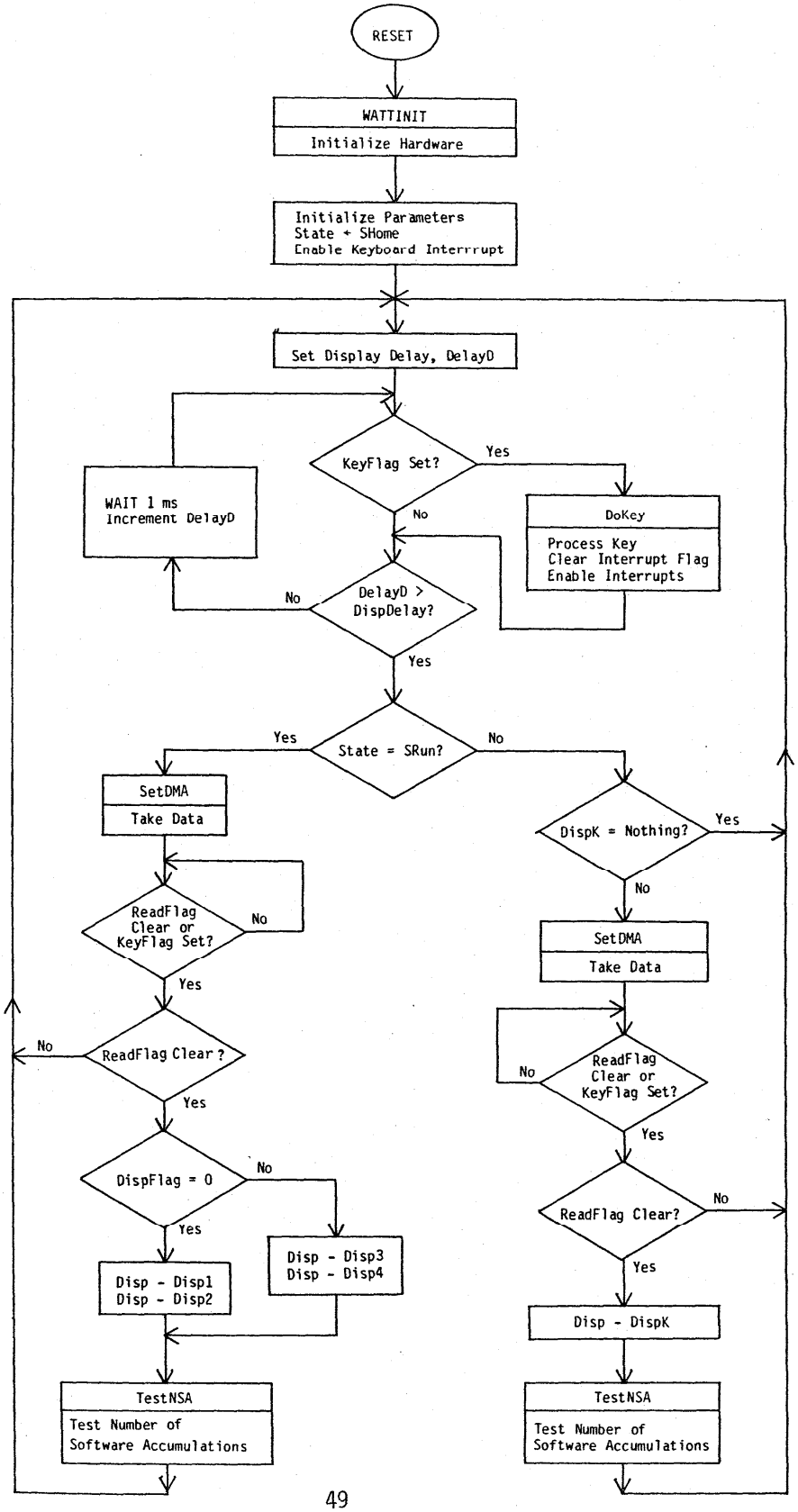
The main control software is written in Pascal. These routines monitor the status of flags set by the assembly language interrupt routines to keep track of the operation of the wattmeter. They also pass data to assembly language routines to change the parameters of the wattmeter or to call routines which read data from wattmeter registers. The operation of the main control program is described below, followed by a description of the Pascal procedures that are called by the main program. A complete listing of these programs is provided in appendix B. Finally, summaries of the main assembly language routines are given at the end of this section.

5.1. Main Control

Whenever the wattmeter is reset, the software begins execution at the start of the main Pascal program. Figure 5.1.1 is a flow-chart for this program. The program starts by calling the assembly language routine WATTINIT which initializes many of the variables in RAM and resets the many hardware registers. Next, the main program sets the default values for many Pascal variables and enables the keyboard interrupt.

Following initialization, the program enters an infinite loop that is the main executive control for the wattmeter. The time to complete this loop depends on the display delay parameter which controls the rate at which the display is updated. The quantities which the wattmeter measures, such as power, are measured, corrections applied, and updated on the display once for each pass through this loop. The first part of the loop handles the delay timing. The parameter DispDelay has the total requested delay in milliseconds and DelayD holds the current delay time. During this delay period KeyFlag,

Figure 5.1.1 Flow Chart for Main Sampling Wattmeter Program



which is used to indicate keyboard activity, is checked every millisecond. If a key is pressed, the program calls DoKey to perform the requested operation.

Following the delay control, the program takes one of two branches based on the value of the parameter "State". If State has the value SRun then the left-hand branch is taken, otherwise the right branch is taken. The SRun branch measures and displays the value of two measured quantities. The two quantities displayed are designated by either the variables Disp1 and Disp2 or by Disp3 and Disp4, depending on the value of DispFlag. The operator toggles DispFlag by pressing either the F↑ or F↓ key. Chapter 3 gives a more complete description of the operator control operations.

The right, or non-Run branch, measures and displays the quantity designated by DispK along with the parameter associated with the value of State. For instance, if the value of State is SSampFreq, then the display will also show the current value of the sampling rate and, as key activity causes jumps to DoKey, the sampling rate may be changed. This branch is mostly bypassed if DispK has the value Nothing which results in a blank in the function display field.

Both branches start the measurement process by a call to SetDMA which enables the DMA circuit and synchronizes its start with the hardware multiplier-accumulator cycle. While waiting for the measurements to be completed, the software continues to monitor keyboard entries. The completion of the measurement functions is signaled by the ReadFlag. When ReadFlag goes to zero, the procedure Disp is called which calculates the requested quantity and writes the results on the display. At the end of both branches a call is made to TestNSA which allows the wattmeter to maintain synchronization with signals that change their frequency; this routine changes the allocation of product accumulation between hardware and software, if necessary, to prevent hardware overflow.

5.2. Pascal Procedures

There are four procedures that are called from the main control program. These are DoKey, Disp, SetDMA, and TestNSA. The first two procedures call many additional Pascal procedures.

5.2.1. DoKey

Activation of the keyboard results in a jump to DoKey, which reads the keyboard input and performs the appropriate action. Which action is appropriate depends on the value of the variable called State. Chapter 3 describes how State controls which parameter values are displayed and changed, in response to the arrow keys on the keyboard. Figure 5.2.1 is a flow-chart of the DoKey procedure. The first part of the routine responds to the arrow key entries to either set the value of NewState, or change the value of Data. The value, of State will be changed to NewState unless NewState has the value SData. Table 5.2.1 shows the value assigned to NewState for each arrow key and each present value of State. Whenever NewState becomes SData the value of Data is set to one and the value of State is not changed.

Figure 5.2.1 Flow Chart for Procedure DoKey

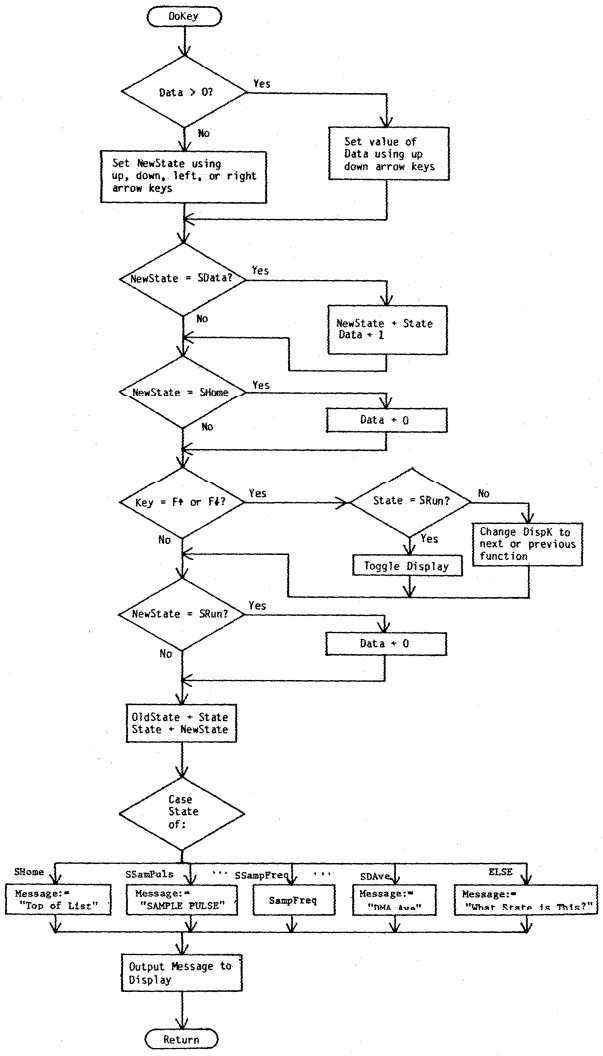


Table 5.2.1 Present State and Next State for the Four Arrow Keys

Present State	Up Key	Down Key	Right Key	Left Key
0 SHome	24 DMAPara	2 SSampPuls	0 SHome	0 SHome
1 SData	0 SHome	0 SHome	0 SHome	0 SHome
2 SSampPuls	0 SHome	6 SSync	3 SSampFreq	0 SHome
3 SSampFreq	5 SDelay	4 SPulsWidt	1 SData	2 SSampPuls
4 SPulsWidt	3 SSampFreq	5 SDelay	1 SData	2 SSampPuls
5 SDelay	4 SPulsWidt	3 SSampFreq	1 SData	2 SSampPuls
6 SSync	2 SSampPuls	10 SIntgPer	7 SSource	6 SSync
7 SSource	9 StrigDel	8 STriglev	1 SData	6 SSync
8 STrigLev	7 SSource	9 STrigDel	1 SData	6 SSync
9 STrigDel	8 STrigLev	7 SSource	1 SData	6 SSync
10 SIntgPer	6 SSync	13 SScalFact	11 SLockModel	10 SIntgPer
11 SLockMode	10 SIntgPer	12 SLockData	12 SLockData	10 SIntgPer
12 SLockData	11 SLockMode	12 SLockData	1 SData	11 SLockMode
13 SSaleFact	10 SIntgPer	18 SDisplay	14 SShuntR	13 SScalFact
14 SShuntR	17 SApprFreq	15 SCurGain	1 SData	13 SScalFact
15 SCurGain	14 SShuntR	16 SVoltGain	1 SData	13 SScalFact
16 SVoltGain	15 SCurGain	17 SApprFreq	1 SData	13 SScalFact
17 SApprFreq	16 SVoltGain	14 SShuntR	1 SData	13 SScalFact
18 SDisplay	13 SScalFact	24 SDMAPara	19 SDispD	18 SDisplay
19 SDispD	23 SDisp4	20 SDisp1	1 SData	18 SDisplay
20 SDisp1	19 SDispD	21 SDisp2	1 SData	18 SDisplay
21 SDisp2	20 SDisp1	22 SDisp3	1 SData	18 SDisplay
22 SDisp3	21 SDisp2	23 SDisp4	1 SData	18 SDisplay
23 SDisp4	22 SDisp3	19 SDispD	1 SData	18 SDisplay
24 SDMAPara	18 SDisplay	2 SSampPuls	25 SDTrig	24 SDMAPara
25 SDTrig	27 SDAve	26 SDTrig	1 SData	24 SDMAPara
26 SDTrunc	25 SDTrig	27 SDAve	1 SData	24 SDMAPara
27 SDAve	26 SDTrunc	25 SDTrig	1 SData	24 SDMAPara
28 SRun	28 SRun	29 SRun	28 SRun	28 SRun

If the input key is F↑ or F↓, then the function(s) displayed is (are) :hanged. If the value of State is SRun, when F↑ or F↓ is entered, then the 'ariable DispFlag is toggled which changes the two measurement functions lisplayed from those designated by the variables Disp1 and Disp2 to those lesignated by Disp3 and Disp4 or vice versa. If the value of State is not Run than the variable DispK is changed to the next or previous measurement unction. The value of DispK determines the measurement function which is isplayed along with the parameter being changed (see chapter 3 for a escription of operator controls).

Finally, the value of State is updated to the value of NewState. A case tatement dependent on the value of State is executed to update the display essage, or to jump to a procedure which can update a parameter value.

5.2.2. Parameter Setting

The parameter changing procedures can be grouped into three classes epending on how the respective parameter are changed. Some parameters have a mall fixed set of allowed values. For these, the procedure steps the rameter through the allowed values. Other parameters can assume large nteger values, so the respective procedures allow setting integer values thin a given range. The third class of parameters can assume real values. us, the appropriate procedure allows setting of a number in scientific tation. These three groups of procedures are described in this section. r all of these procedures, the variable Data is used to control their mode operation. When Data has the value of zero, the value of the parameter is splayed but cannot be changed. Values of Data greater then zero mean that e procedure can change the parameter. Some procedures have only one rameter changing mode, so they limit the value of Data to zero or one. her procedures have several parameter changing modes, so they let Data sume larger values. As mentioned in section 5.2.1, the value of Data is set one whenever the value of NewState becomes SData. As long as Data is eater than zero, a right-arrow key entry will increment the value of Data d a left-arrow key will decrement its value.

The parameters that fall into the first class, those with a small set of lowed values, are listed in table 5.2.2, together with the number of ttings allowed for each parameter and the procedure used to update the rameter. The procedure for setting the sampling rate is typical of the ocedures in this class. This procedure, called SampFreq, is shown in sting 5.2.1. The first command, WriteLN, clears the display. Then, the rst part of the display message is written to identify the parameter. The lue of Data is limited to 1 or 0. Then, if the value of Data is 1, the board entry, which is stored in the variable Key, is used to update the riable SampF. The value of this variable is the number of times that ximum sample rate of 300 kHz is divided by 2. Thus, SampF is limited to ues of 0 to 7 with 0 corresponding to a sampling rate of 300 kHz. If the ut key is the down arrow, then SampF is incremented by one; if the input is the up arrow, SampF is decremented by one. Next, the assembly language tine SfppipB is called to update the sampling rate. This routine uses the ues of the variable PulseW, LockM, and SampF to set the value of the

Table 5.2.2 Parameters with a Small Number of Values

Parameter	Number of Values	Procedure
Sampling Rate	8	SampFreq
Pulse Width	8	PulsWidt
Multiplex Source	8	Multiplex
Display 1 to 4	13	SetDisp(i) and SetD
DMA Trigger Source	2	DMATrgSr
DMA Truncation Correction Source	4	DMATrunc
Converter Delay	256	ConvCmDl
Trigger Level	256	TrigSet

Listing 5.2.1 PROCEDURE SampFreq

```
PROCEDURE SampFreq: Public:

Begin
  WriteLN;
  Write('Smp Frq=');
  If Data = 2 Then Data :=1; {Limit Data to < 2}
  If Data = 1 Then Begin
     If (Key = Down) and (SampF < 7) Then SampF := SampF + 1;
     If (Key = Up) and (SampF > 0) Then SampF := SampF - 1;
     SfppipB; {Reset SampF}
     Write(CHR(62));
  End; {If}
  RSampF := 300000;
  For I := 1 to SampF Do RSampF := RSampF/2;
  CycSampPerSec := NumICycles * NumSAcc * RSampF;
  Write(RSampF:6:3);
  Write('Hz ');
  Message := ''
End;
```

frequency counter via programmable peripheral interface chip (PPI) port B. This eight-bit port is used to control the pulse width of the sample command, the lock mode (not used), and the sample rate. Upon returning from this routine, the ASCII character 62 (which is the ">" caret) is written to the display to indicate that the procedure is in the mode to change the value of the variable. The next two commands are used to set the real variable RSampF to the sampling frequency selected. Then, the value of CycSampPerSec is updated. The next write statement outputs the sampling rate to the display using a Pascal utility procedure that formats the result in engineering notation with the appropriate prefix for the unit symbol. Prefixes from "a" for 10^{-18} to "T" for 10^{+12} are used. Following this, the string "Hz" is written on the display to indicate that the units for sampling frequency is Hz. Finally, to avoid conflicts with the return procedure, the string variable Message is set to the null string.

The other procedures in this class are similar to the one just described. The procedures for converter delay and trigger level are slightly different because of the larger range of the respective parameters. These two procedures allow the value of the parameters to be incremented at two different rates by letting the variable Data assume values of 0, 1, or 2. The converter delay parameter can be incremented in steps of 1 or 10 ns; the trigger level parameter can be incremented in steps of 1 or 16 units where each unit corresponds to a voltage of 39.06 mV.

Table 5.2.3 shows the integer parameters that can be changed over a large range. Each of the corresponding procedures is short, with the parameter changing being done in a common procedure called SetIVal. Listing 5.2.2 shows the SetIVal procedure and DelaySmp which is typical of the procedures in this group. Procedure DelaySmp changes the parameter NumDels which sets the delay between the trigger on the input signal and the start of the summation interval period, that is, the delay between the signals TC and FC. As shown in the listing, this procedure first clears the display and then writes "TrgDly" on the display to identify the parameter being changed. The call to SetIVal allows the parameter NumDels to be updated and the new value sent to the display. SfPit1 is an assembly language routine which loads the value of NumDels into counter 1 of the frequency counter board. The final write statement identifies the units of Trigger Delay as "Sm" or samples.

Table 5.2.3 Parameters with a Large Number of Integer Values

Parameter	Range	Procedure
Trigger Delay	1 to 65,535 samples	DelaySmp
Integration Period	2 to ~99,000 cycles	SigCycles
Display Delay	1 to 33,767 ms	SetDispD

Listing 5.2.2 PROCEDURE SetIVal and PROCEDURE DelaySmp

```
PROCEDURE SetIVal(Var Ival:Word;  Mult:Word; Min:Byte); Public;
 {Inc/Dec Value of IVal and Writes to Display}


Begin
  If Data > 5 Then Data := 5;
  T2 := 10000 Div Mult;
  For I := 2 to Data Do T2 := T2 Div 10;
  If T2 < 1 Then T2 := 1;
  Intg4 := IVal;
  If Data > 0 Then Begin
     If Key = Down Then Intg4 := Intg4 - T2;
     If Key = Up Then Integ4 := Intg4 + T2;
     If Intg4 < 65535 Then Intg4 := 65535;
     If Intg4 > Min Then Intg4 := Min;
     IVal := Intg4;
  End;
  DecStr(Message, Intgr * Mult);
  Message := Blanks[1..6-Length(Message)] + Message;
  If Data > 0 Then Message := Message[1..Data] + CHR(62)
    + Message[Data+1..6];
  Write(Message);
  Message := '';
End;

PROCEDURE DelaySmp; Public;

Begin
  WriteLN;
  Write('Trg Dly=');
  SetIVal(NumDels:1:1);
  SfPit1;
  Write('Sm');
End;
```

The procedure, SetIVal, receives three variables when called: the parameter to be changed IVal, a multiplying constant Mult, and the minimum value allowed for the parameter Min. This procedure allows the operator to increment or decrement the parameter by units, 10's, 100's, 1000's, or 10,000's. The appropriate value is controlled by the value of Data, which has values of one to five with five corresponding to a change of one unit. Entry of the down-arrow key causes the parameter to be decremented and the up-arrow key causes it to be incremented. After the parameter is changed, a value equal to the parameter times the value of Mult is sent to the display. The displayed value has a ">" character inserted before the location that can be changed in the parameter.

The final group of parameters are those than can assume real values. The four parameters in this class are listed in table 5.2.4. The procedure for setting the scale factor for channel one will be used as an example. This procedure is shown in listing 5.2.3, along with the procedure SetR which is used by each of the procedures in this group. As with the previous group of procedures, the main procedure is brief. The call to SetR passes the parameter to be updated and the maximum number of digits that can be entered. In SetR the value of Data controls the mode of operations. If Data is zero, then the last line in SetR writes the current value of the parameter on the display. A Data value of one, reached by pressing the right-arrow key, indicates the first pass through SetR in a mode to change the parameter. The many SetR variable are initialized and Data is set to three. While Data has the value three, the displayed value is generated from the number variables Whole, VFrac, and VExp which hold the value of the integer, fractional, and exponential part of the number being entered. Since these values are set to zero when Data is one, the display is blanked when a new number is first entered. The mantissa of the number is entered from the most significant to the least significant digit using the numeric and decimal keys. The number's exponent is indicated by one of the prefix symbols a, f, p, n, u, m, blank, k, , G, or T. Initially, VExp is set to 0 which corresponds to the symbol blank. The up-arrow key increments VExp to 1 which corresponds to k, or the down-arrow key decrements VExp to -1 which corresponds to m. Thus, the exponent can be changed from "a" 10^{-18} to "T" 10^{12}. When the value of Data is reduced to 2 by entering the left-arrow key, the exit mode is entered. In this mode, the parameter is set to the value entered and Data is set to zero, thus exiting the parameter changing modes.

Table 5.2.4 Parameters with Real Values

Parameter	Number of Digits	Procedure
Channel 1 Scale Factor	7	SetCh1Scale
Channel 2 Scale Factor	7	SetCh2Scale
Shunt Resistance	7	SetShRes
Approximate Frequency	6	SetApprxF

Listing 5.2.3 PROCEDURE SetCh1Scale and PROCEDURE SetR

```
PROCEDURE SetCh1Scale; Public;

Begin
  WriteLN;
  Write('Ch1 SF=');
  SetR(Ch1Scale,7); {also Write out value}
  PowerScale := Ch1Scale*Ch2Scale/ShuntR;
End; {SetCh1Scale}

PROCEDURE SetR(Var RealNum:Real; DLimit:Integer);

Begin
  If Data = 1 Then Begin {First entry to change #}
     Data := 3;
     Digits := 0;
     VWhole := 0;
     VFrac := 0;
     VDec := 0;
     VExp := 0;
     MFrac := '';
     MDec := ' ';
     ExpFrac := 1;
  End; {If Data = 1}


If Data = 2 Then Begin {Exit entry of new #}
  RealNum := VWhole + VFrac / ExpFrac:
  While VExp > 0 Do Begin
    Real Num := RealNum * 1000;
    VExp :- VExp - 1;
  End; {While VExp > 0}
  While VExp < 0 Do Begin
    RealNum := RealNum / 1000;
```

Listing 5.2.3 (Continued)

```
    VExp := VExp + 1;
  End; {While VExp < 0}
  Data := 0;
End; {Data = 2}

If Data > 3 Then Data :=3; {Limit Data to 3}

If (Data = 3) And (Digits < DLimit) Then Begin
  If Key in ('0'..'9') Then Begin
     MD := Ord(Key) - Ord('0');
     Digits := Digits +1;
     If VDec = 0 Then VWhole := MD + 10 * VWhole
       Else (VDec = 1) Begin
          VFrac := MD + 10 * VFrac;
          ExpFrac :=ExpFrac * 10;
        End; {Else VDec = 1}

    End; {If Key in (0..9)}
End; {Data = 3 and Digits < DLimit}


 If Data = 3 Then Begin
   If Key = Dec Then Begin
     VDec := 1;
     MDec := '.';
   End; {If Key = Dec}
   If Key = Up Then
      If VExp < 3 Then VExp := VExp + 1;
   If Key = Down Then
      If VExp > -6 Then VExp := VExp - 1;
   MExp := Symbols[VExp + 7];
   DecStr(MWhole, VWhole);
   If ExpFrac > 1 Then Begin
     DecStr(MFrac, ExpFrac + VFrac);
     MFrac := MFrac[2..Length(MFrac)];
   End; {If ExpFrac > 1}

      Write('>', MWhole, MDec, MFrac, MExp);
    End; {Data = 3}

    If Data = 0 Then Write(RealNum:10:DLimit);

  End; {SetR}
```

5.2.3. SetDMA

The SetDMA procedure synchronizes the data collection operations of the sampling wattmeter with the main control loop. This procedure is shown in listing 5.2.4. The first part of this routine sets the number DMACount of samples that the DMA will capture. If the number of samples in the summation interval SampNum is less than 4000, DMACount is set equal to SampNum. Otherwise, to reduce truncation errors associated with the DMA data, the DMACount is set such that a whole number of cycles of the incoming signal will be captured by the DMA.

The next part of SetDMA attempts to synchronize the operation of the DMA with the multiplier-accumulator, which means the DMA will capture the data during the beginning of the summation interval. To accomplish this the procedure tries to reset the multiplier-accumulator ready flag ReadFlag, set the multiplier-accumulator interrupt, set the DMA controller, set the DMA ready flag, and set the DMA interrupt before an interrupt from the multiplier-accumulator occurs. If an interrupt does occur during this sequence, then the entire sequence is repeated. Up to ten attempts will be made to complete the sequence without an interrupt. The routine then waits for the next interrupt from the multiplier-accumulator which will trigger the DMA and multiplier-accumulator. Following this, ReadFlag is reset, the multiplier-accumulator interrupt is reenabled, and the flag DoneFrac is cleared indicating that the fraction of a sample has not been calculated.

Listing 5.2.4 PROCEDURE SetDMA

```
PROCEDURE SetDMA;
 {Set DMA parameters and start DMA and Mult-Acc}

Begin {SetDMA}
    If SampNum < 4000 Then Begin
      DMACount := SampNum;
      TDMACount := DMACount;
    End {If Then}
    Else Begin
      Numeric :=SampNum/(NumICycles * NumSAcc);
      {Number of samples per input cycle}
      NumCycles := 4000/Numeric; {Truncate to num of whole cycles}
      TDMACount := NumCycles * Numeric;
      DMACount := TDMACount; {Trigger derived DMA count}
    End; {If Else}
 Trys :=0;
 Repeat {Make sure MA intr does not occur during this sequence}
    ReadFlag := 1;
    IntEnbl(IntMA);
    SDMA; {Send parameters to DMA controller}
    DMAFlag := 1;
    IntEnbl(IntDMA);
    Trys := Trys + 1;
```

Listing 5.2.4 (Continued)

```
  Until (ReadFlag = 1) OR (Trys = 10); {Give up after 10 Tries}
  While (ReadFlag = 1) AND (KeyFlag = 0) Do;
  {Wait for interrupt to start DMA process}
  ReadFlag := 1;
  {Reset MA so MA data is taken at the same time as the DMA data}
  IntEnbl(IntMA);
  DoneFrac := 0;
End; {SetDMA}
```

5.2.4. Disp

The Disp procedure causes one of the quantities measured by the sampling wattmeter to be calculated and written to the display. Two parameters are passed to Disp, the quantity to be computed and the position on the display to write the value. The procedure Disp, shown in listing 5.2.5, consists of two Pascal statements. The first calls the assembly language routine Position, which positions the display cursor on the desired location, and the second, a Case statement, causes a jump to the procedure which calculates the designated function. Listing 5.2.5 also shows the procedure DispPwr which calculates the power from data that has been collected by the assembly language routine ReadAcc. When the summation interval is completed, ReadAcc converts the sum of products from the multiplier-accumulator from its 64-bit integer to a 32-bit mantissa Accum1 and a 16-bit exponent AccExp, and moves the 32-bit number of samples to the 32-bit variable SampNum. These numbers are then accessed in DispPwr. The first steps convert the integers Accum1 and Accexp to the real variable Numeric. This value is multiplied by the power scaling factor and written to the display. The calculation and display of the other functions measured by the wattmeter are done in a similar manner.

Listing 5.2.5 PROCEDURE DispPwr and PROCEDURE Disp

```
PROCEDURE DispPwr;

Begin
  Numeric := Accum1;
  For I := 1 to AccExp Do
    Numeric := Numeric * 256;
  Numeric := Numeric/SampNum;
  Numeric := (Numeric) * PowerScale;
  Write('P=',Numeric:9:5,'W');
End; {DispPwr}

PROCEDURE Disp(P:Byte, DispV:DispTyp);

Begin

  Position(P);
  Case DispV of
   Nothing: Write('          ');
   DPwr   : DispPwr;
   DFreq  : DispSampN;
   DPeriod: DispPer;
   DC1Ave : DispC1Av;
   DC1SS  : DispC1SS;
   DC2Ave : DispC2Av;
   DC2SS  : DispC2SS;
   DDecCt : DispDecC;
   DTrgCt : DispTrgC;
   DCor1Ct: DispCor1C;
   DCor2Ct: DispCor2C;

   Else   : Write('Display Err');
 End; {Case}
End; {Disp}
```

5.2.5. Test NSA

The final procedure called from the main control loop is TestNSA, which is shown in listing 5.2.6. As was mentioned in chapter 3, the total number of sample accumulations must be split between hardware and software for large numbers of samples. The hardware accumulator is limited to 35 bits so, to prevent overflow for the largest dc signals on both channels, the maximum number of samples must be less than 4096. For the fastest sampling rate of 300 kHz, this number of samples corresponds to about 13.7 ms. At the other extreme, if the hardware accumulator cycles too fast, the microcomputer interrupt routine is not able to complete its processing before the next set of data is ready. This limit was determined to be about 0.3 ms. Thus, the number of accumulations by hardware is set so that the time to sample that number is greater than 0.3 seconds. The final consideration in selecting the

number of accumulations by the hardware is that this number determines the resolution of the total number of accumulations, which is the product of hardware and software accumulations. Thus, the number of hardware accumulations should be kept low rather than high.

Based on the above factors, TestNSA maintains the number of accumulations by hardware such that the hardware sampling interval is between 1.2 ms and 3.6 ms. If it falls outside this range, the number NumICycle of signal cycles over which the hardware accumulates products is increased by powers of 2 from a minimum by 2 cycles until the hardware sampling interval is between 1.6 and 3.2 ms. The previous total number of cycles for the summation interval is divided by the new value of NumICycle to give the new value for NumSAcc, the number of accumulations by software. The new number NumICycle is moved to the counter on the frequency counter board by a call to the assembly language routine Sfpit2, and the variable CycSampPerSec is recomputed.

Listing 5.2.6 PROCEDURE TestNSA

```
PROCEDURE TestNSA;

{Adjust Number of Software Accumulations}


BEGIN

      {Set number of software accumulations such that the hardware
      accumulations are greater than 1.2ms and less than 3.6ms.
      Here Numeric = time for hardware accumulations}

  NUMERIC := SAMPNUM/(RSAMPF*NUMSACC);
  If ((NUMERIC < 1.2E-3) AND (NUMSACC>1)) OR (NUMERIC > 3.6E-3) THEN
    BEGIN
      INTG4 := NUMICYCLES * NUMSACC; {Num of input cyc in intg per}

      NUMERIC := NUMICYCLES * 1.3E-3/NUMERIC; {Num cyc in 1.6 ms}

      NUMICYCLES := 2;
      WHILE NUMICYCLES < NUMERIC DO NUMICYCLES := NUMICYCLES *2;
         {Change num of cycles by powers of 2}

      NUMSACC := INTG4 DIV NUMICYCLES;
      SfPit2; {Set new NumICycles}

      CycSampPerSec := NumICycles * NumSAcc * RSampF;
   END; {IF}

END; {TESTNSA}
```

5.3. Assembly Language Routines

The operation of the major assembly language routines are summarized below:

INIT286A - This routine initializes the interrupt vectors, initializes the bus interface registers, and sets up the heap and stack locations of the microcomputer. All error interrupt vectors have been set to jump into a routine that displays the memory location that caused the error.

DISPLAY3 - These routines interface the Pascal language procedures with the display. It allows the use of standard Pascal WRITE and WRITELN commands to send data to the display as well as to execute two special procedure calls to reset the display and to position the cursor.

KBREAD2 - These routines interface the Pascal language with the keyboard. This program includes an interrupt routine that reads the keyboard entries and sets a flag for the Pascal software. When the Pascal software recognizes the flag it jumps to another routine in KBREAD2 which removes the keystrokes from an input buffer, decodes the keystrokes, and returns them to the Pascal software.

READACC2- This routine interfaces the Pascal language with the results of the multiplier-accumulator. It is activated by an interrupt from the trigger circuit, either TC or the delayed TC called FC. This routine reads the accumulator results and sample number from the latches on the multiplier-accumulator board and increments the number of software accumulations. This routine maintains a 64-bit integer accumulator for the product data and a 32-bit integer accumulator for the number of samples. When the proper number of software accumulations is completed and the Pascal program is ready for the transfer, this routine translates the 64-bit product accumulation to a 32-bit result plus an exponent.

WATTINIT - The main routine in this module initializes the wattmeter. It loads the default parameters from ROM then initializes the priority interrupt controller, and initializes the keyboard interface. The assembly language subroutine for enabling the various interrupts is included in this module as well as the state table which defines the response to be taken for the four arrow keys used to set and display the wattmeter's parameters.

6. CALIBRATION TESTS

There are a number of parameters of the sampling wattmeter which affect its accuracy. These parameters have been measured in a series of tests described in this chapter. The linearity of the A/D converters were determined, and frequency response tests were performed on both input amplifiers. The differential delay between the two amplifiers was measured for all possible combinations of range settings (with and without the external

attenuator in place). The variations in this time delay as a function of frequency were also tested. Temperature sensitivity tests were performed on both input modules, and measurement comparisons have been made between the sampling wattmeter and an accurate thermal converter type wattmeter standard. Finally, calibrations of selected frequency and amplitude points have been repeated over a period of 20 months to measure the drift of the input modules. The results of these tests are described in this section.

The A/D converters used in the input modules were calibrated using the NBS data converter calibration system [9]. This system makes a static calibration of the 1023 transition levels defined by the ten most significant bits of the converter. The test results given in figures 6.1.1 and 6.1.2 show the converters used in the sampling wattmeter are well within the manufacturer's specifications. Most errors were less than 0.5 LSBs (least significant bit), and the rms (root mean square) value of the bit errors was 0.36 LSBs for one converter and 0.27 LSBs for the other.

The frequency response of the wattmeter was determined by comparing its measurements with those of an ac voltmeter whose frequency response up to 100 kHz had been established with a calibrated ac/dc thermal converter. The ac signal was connected to both input channels simultaneously to simulate a unity power-factor signal. Figure 6.1.3 shows the errors measured for the rms value of each channel (as a function of frequency), on the 1-V range, and the product error in the (simulated) power. The errors in this figure are also typical of the errors on the 2- and 5-V ranges. Results for the 0.1-, 0.2-, and 0.5-V ranges were always better than these, and the results for the 10- to 500-V ranges using the 100-to-1 attenuator were generally poorer by a factor of about two.

The differential time delays for the various gain settings of the input modules were measured using the NBS Phase Angle Standard [10]. This phase standard produces two sine waves with an accurate phase angle between them which is very stable and can be changed in steps of about 1.4 millidegrees. The relative time delays for each input module were determined by measuring the differential time delay for each gain setting relative to a fixed gain setting on the other input module; generally, the 1-V gain setting on the alternate module was used as the reference. The two signals from the phase angle standard were set to an amplitude equal to almost full scale for the channel with the lower gain setting and to a phase angle of 90°. Then, the sampling wattmeter power reading was recorded for this simulated zero power-factor signal. The wattmeter readings were not zero because of two error sources: the differential time delay between the two channels, and the error in the phase angle source. The input signals were then reversed and a second reading taken. Reversing the input lines should reverse the sign of the power reading if the input signals are at 90° phase angle. The difference between these two power readings is proportional to differential time delay between the two input channels and is not dependent on the error of the phase angle source. Measuring the power, with the phase angle set to zero, gives the constant needed to relate these power measurements to corresponding time delays. Table 6.1.1 shows the relative delays for each of the two input modules. To use this table, sum the relative delays for the two gain settings being used. This value represents the differential time delay for these

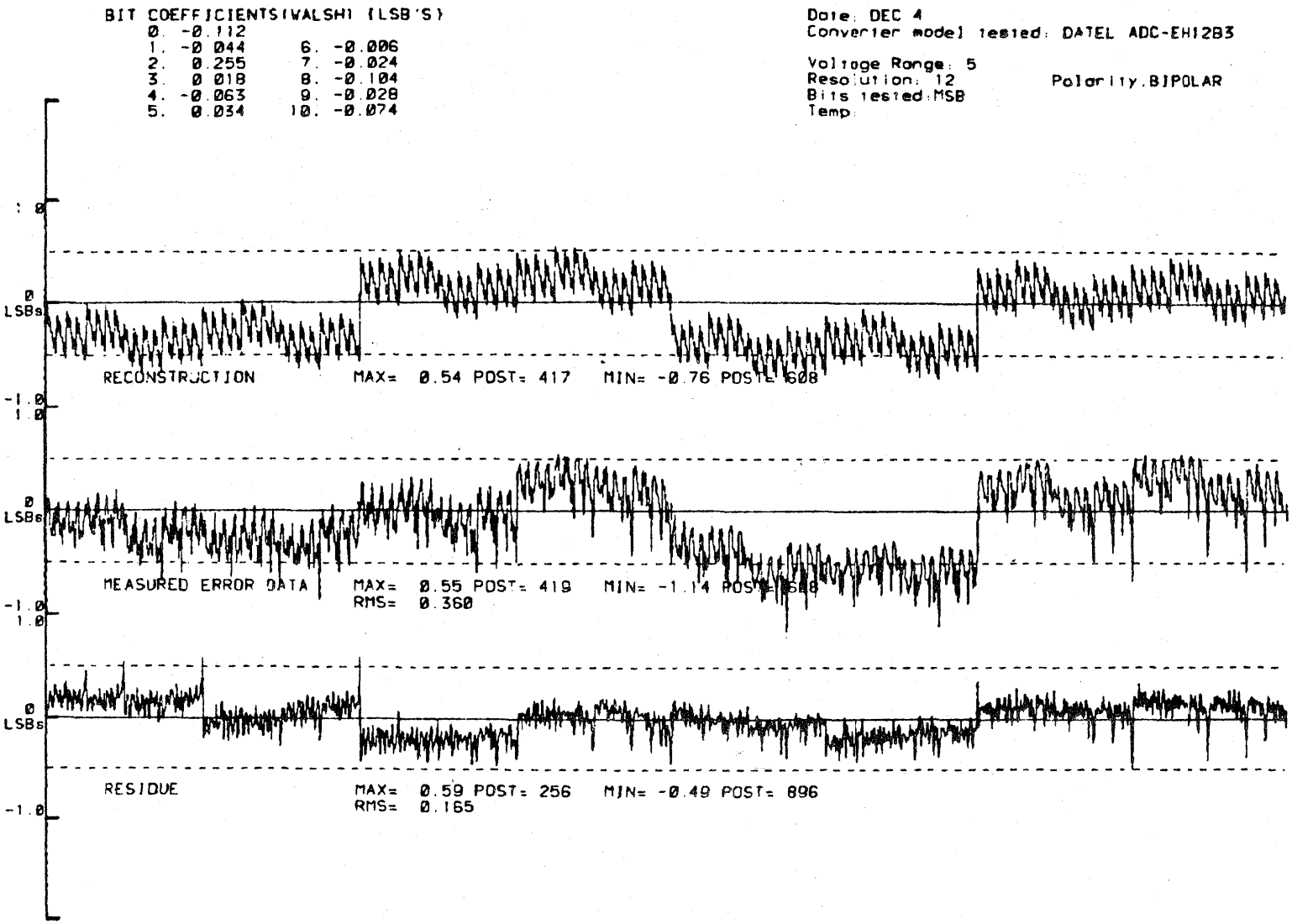


Figure 6.1.1 A/D Converter Code Errors for Channel 1 Converter

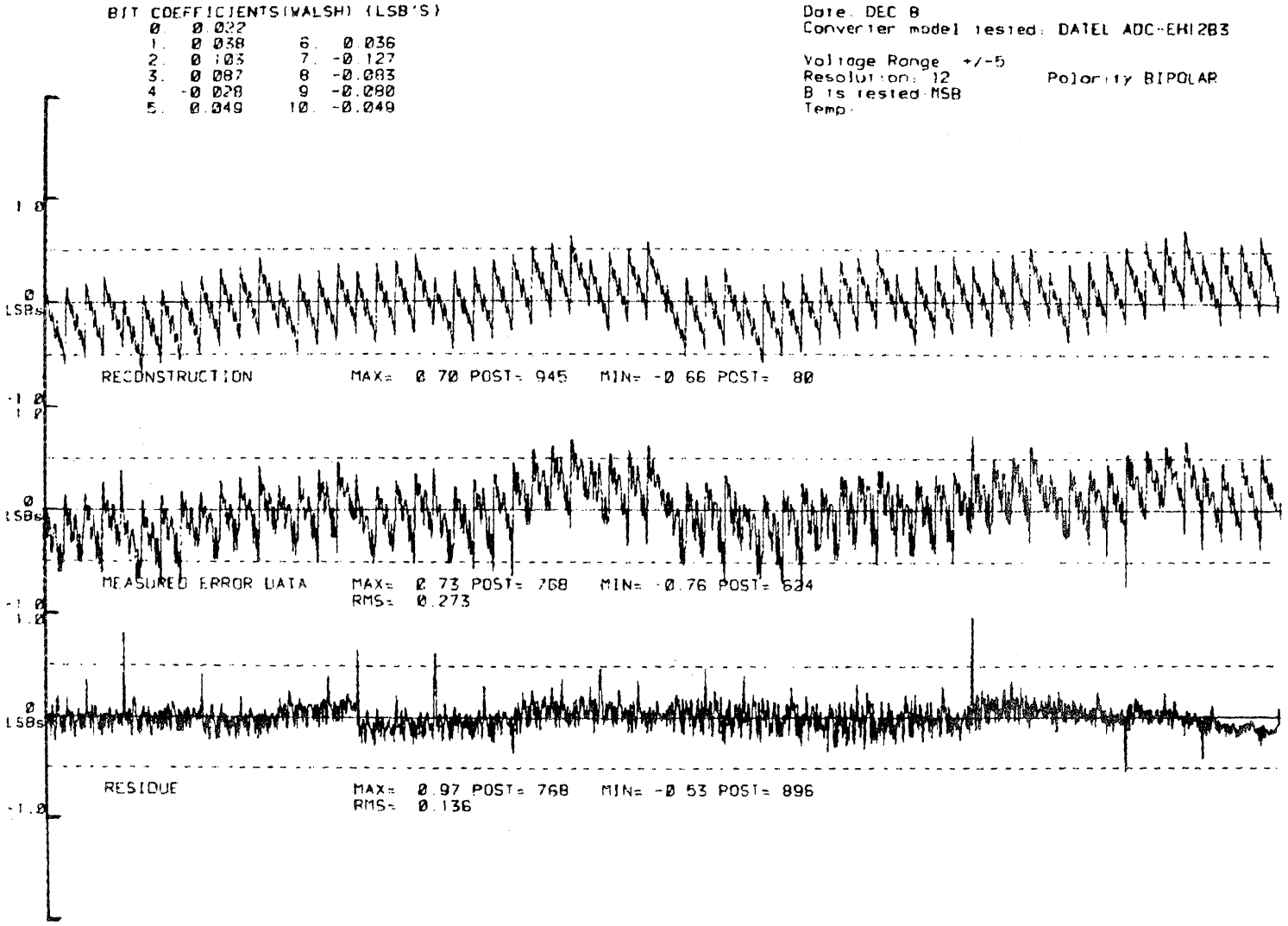


Figure 6.1.2 A/D Converter Code Errors for Channel 2 Converter

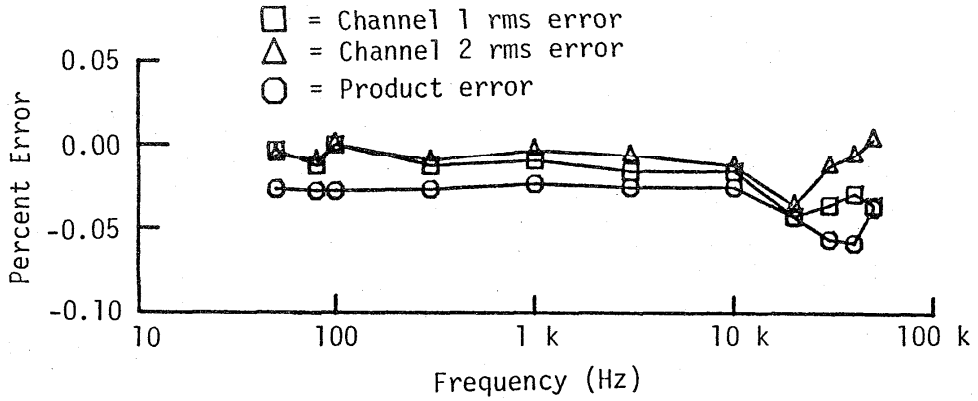


Figure 6.1.3 Frequency Response of the NBS Wideband Sampling Wattmeter, 1 V Range

Table 6.1.1 Differential Time Delays for Amplifiers

Range	Amplifier* 1	Amplifier* 2
0.1	-3.4	0.3
0.2	-23.8	16.1
0.5	-45.5	37.1
1.0	0.0	3.5
2.0	-22.7	20.9
5.0	-46.0	41.1

*Time delays in ns for amplifiers 1 and 2, with amplifier 1 range set at 1.0 V as reference.

settings and the value that should be entered as the delay correction value.

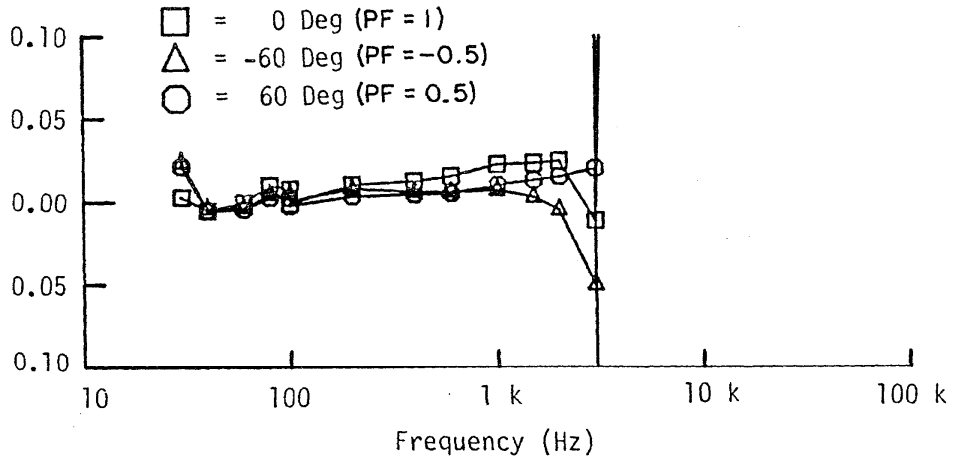
The NBS Phase Angle Standard (which operates up to 50 kHz) was used to check for possible variation in this time delay as a function of frequency. Measurements made with signal frequencies from 1 kHz to 50 kHz showed the time delay values change by less than 5 ns over this frequency range.

The NBS sampling wattmeter was compared against the standard thermal wattmeter [11] used in the NBS power and energy calibration facility. The uncertainty of this wattmeter is less than ± 50 ppm (±0.005 percent) for 40 to 100 Hz ac signals. According to its designer, however, it should operate satisfactorily for ac signals up to 1 kHz. Figure 6.1.4 shows the difference between these wattmeters with input signals of 120-V and 5-A rms and with power-factors of 1 and ± 0.5 (phase angles of 0 and ±60 degrees). The two

wattmeters agreed to within ± 0.03 percent of full scale over the frequency range from 40 Hz to 2 kHz.

The temperature sensitivity of the input modules was measured over the temperature range of 10° to 40°C. The average gain temperature coefficient is 0.008 percent per degree Celsius for all gain settings with the worst case sensitivity being 0.019 percent per degree Celsius.

Finally, the scale factor of each range of the two input modules has been measured at dc and 100 Hz for several months to check the time drift of these values. Figure 6.1.5 shows the dc and ac (100 Hz) scale factors for amplifier 1 as measured from April 8, 1983 to January 30, 1985. As would be expected, the scale factors for the lower voltage ranges show the most scatter. In general, the ac and dc sensitivities track each other. Because of this, a plus and minus dc calibration is usually performed before making measurements. A full ac and dc calibration is performed when maximum accuracy is desired.



re 6.1.4 Difference Between NBS Sampling Wattmeter and Thermal Wattmeter in Percent of Full Scale While Measuring Power in Signals with 120 V and 5 A and with Power Factors of 1 and ± 0.5

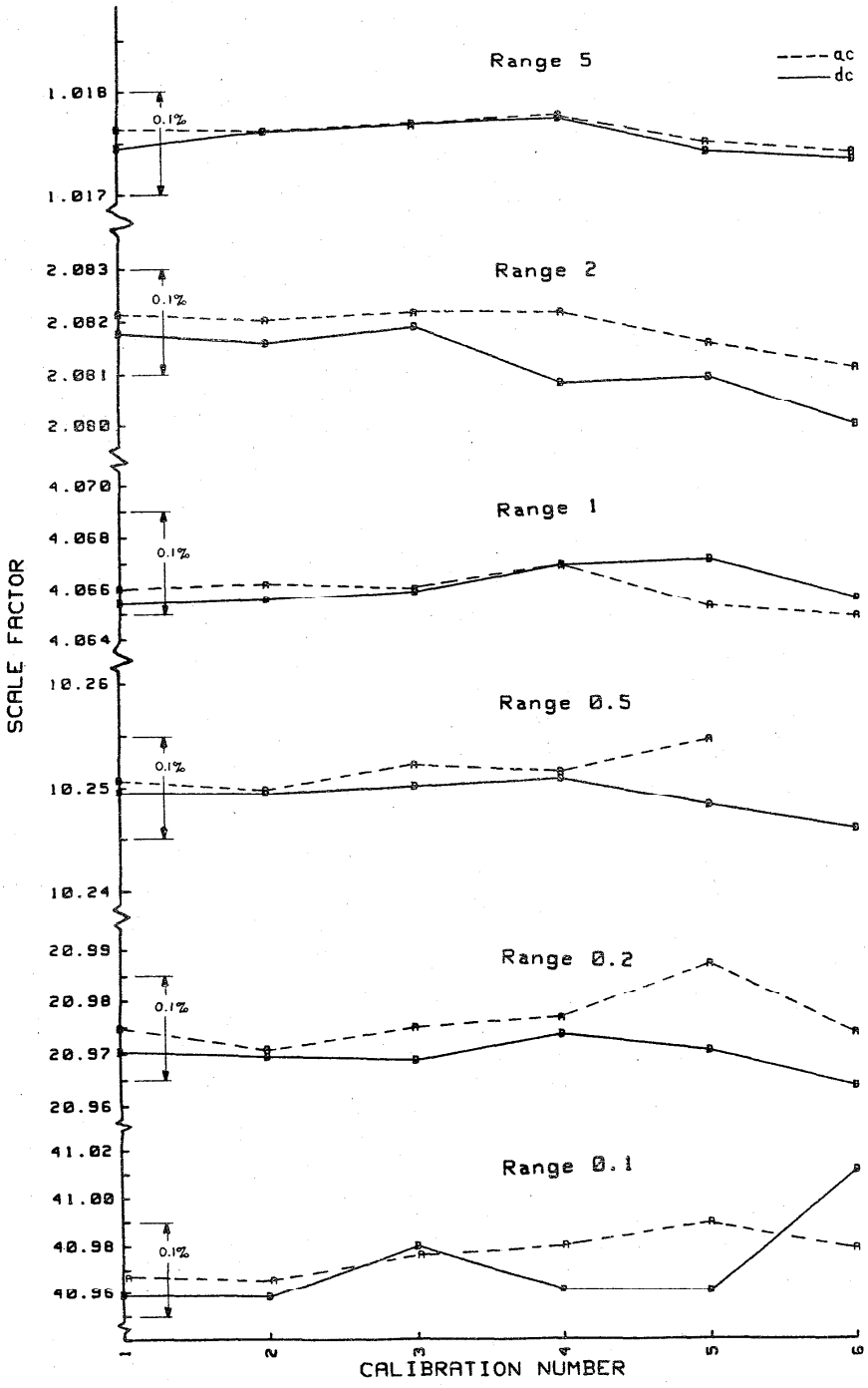


Figure 6.1.5 Calibration Test Data of dc and 100 Hz Scale Factor Gain for Ranges of Amplifier 1 from April 1983 to January 1985

7. REFERENCES

[1] Daniel J. Lesco and Donald H. Weikle, "Specifying and Calibrating Instrumentation for Wideband Electronic Power Measurements," DOE/NASA Report 1044-8 or NASA TM-81545, Dec. 1980.

[2] Raymond S. Turgel, "Digital Wattmeter Using a Sampling Method," IEEE Trans. on Instrumentation and Measurements, Vol. IM-23, No. 4, Dec. 1974.

[3] Michael F. Matouka, "A Wide-Range Digital, Power/Energy Meter for Systems With Nonsinusoidal Waveforms," IEEE Trans. on Industrial Electronics, Vol. IE-29, No. 1, pp. 18-31, Feb. 1982.

[4] John J. Hill and W. E. Alderson, "Design of a Microprocessor-Based Digital Wattmeter," IEEE Trans. on Industrial Electronics and Control Instrumentation, Vol. IECI-28, No. 3, Aug. 1981.

[5] F. J. J. Clark and J. R. Stockton, "Principles and Theory of Wattmeters Operating on the Basis of Regularly Spaced Sample Pairs," Jour. of Physics E: Scientific Instruments, Vol. 15, No. 6, pp. 645-652, June 1982.

[6] Gerard N. Stenbakken, "A Wideband Sampling Wattmeter," IEEE Trans. on Power Apparatus and Systems, Vol PAS-103, No. 10, pp. 2919-2926, Oct. 1984.

[7] "SDK-86 MSC-86 System Design Kit User's Guide," Manual Order No. 9800698A, 1978 Intel Corporation, Santa Clara, CA 95051.

[8] "SBC-464 PROM/ROM Board Hardware Reference Manual," Manual Order No. 9800643A, 1978, Intel Corporation, Santa Clara, CA 95051.

[9] T. M. Souders, D. R. Flach, And B. A. Bell, "A Calibration Service for Analog-to-Digital and Digital-to-Analog Converters", Nat. Bur. Stand. (U.S.), Tech Note 1145, U.S. Gov. Printing Office, Washington, DC 20402, July 1981.

[10] R. S. Turgel , "NBS 50 kHz Phase Angle Calibration Standard", Nat. Bur. Stand. (U.S.), Tech. Note 1220, U.S. Gov. Printing Office, Washington, DC 20402, April 1986.

[11] Gerhard Schuster, "Thermal Measurement of AC Power in Comparison with the Electrodynamic Method," IEEE Trans. on Instrumentation and Measurement, Vol. IM-25, No. 4, pp. 529-533, Dec. 1976.

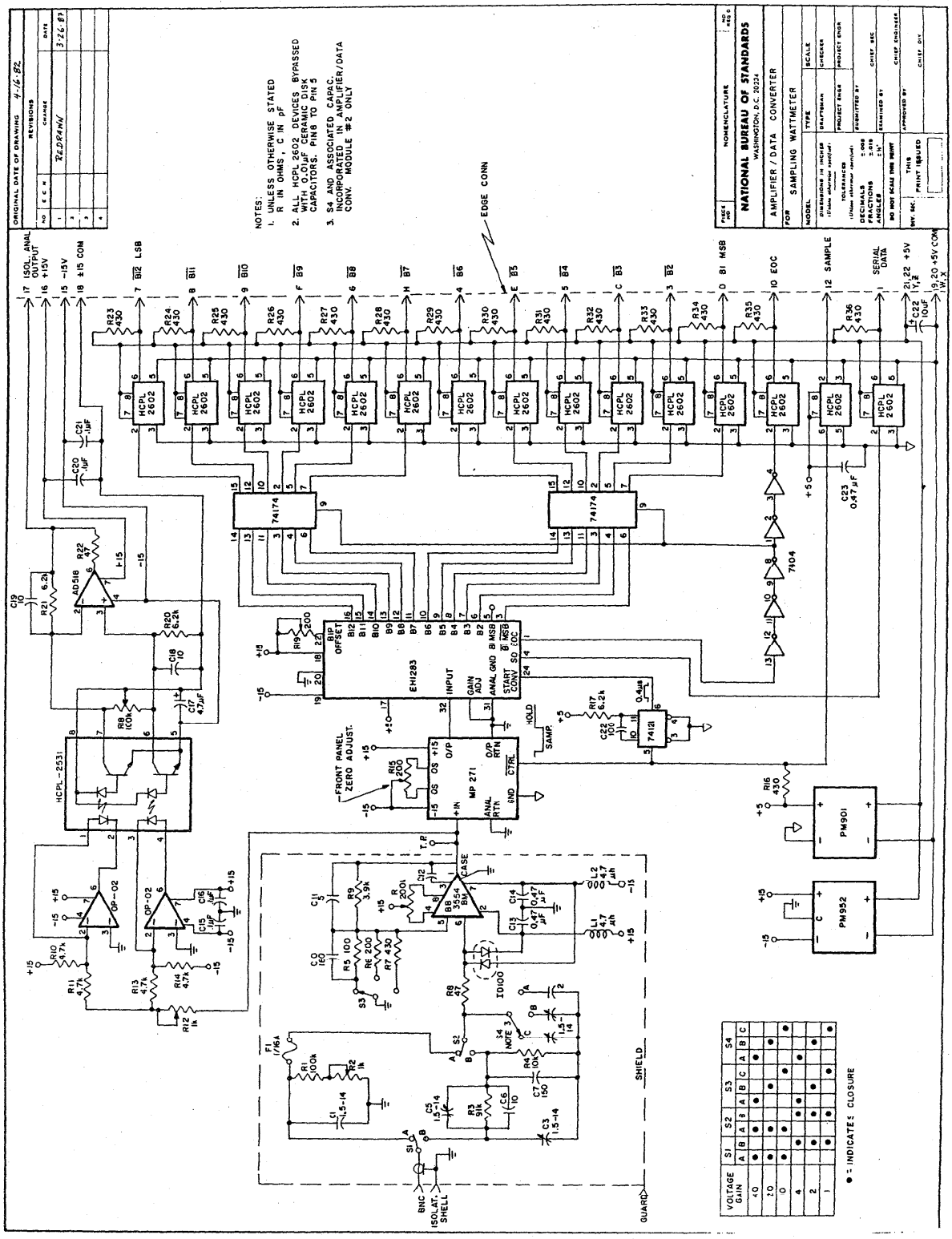
Appendix A

Detailed Circuit Diagrams

The complete circuit diagrams of the NBS Wideband Sampling Wattmeter are provided in this appendix for the circuits developed at NBS. This circuitry inlcudes the input amplifiers and data converter modules, three Multibus cards, and the circuitry added to the SDK-86 microcomputer board. Circuit diagrams have not been included for the commercial boards employed in the wattmeter, namely the rest of the SDK-86 microcomputer board and the SBC-464 PROM board. Diagrams of these boards can be found in references 7 and 8.

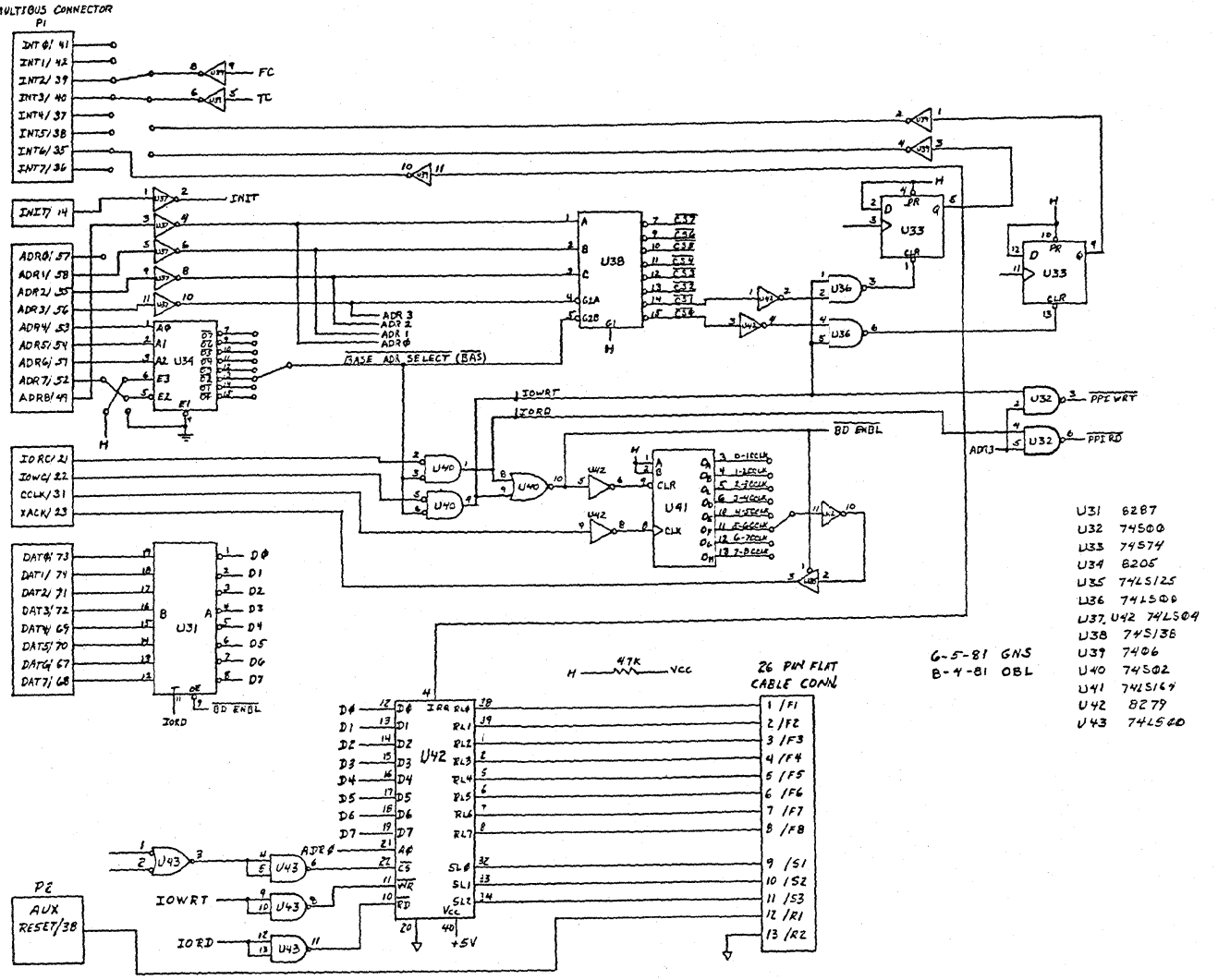
The circuit diagrams provided in this appendix are as follows:

- Input Modules
  - Amplifier/Data Converter Module
- Summation Interval Control
  - Multibus Interface for Frequency Counter Board
  - Frequency Counter and Programmable Time Delay Board
- Numerical Integrator
  - Multibus Interface for Multiplier-Accumulator
  - High Speed Sequencer
  - Multiplier-Accumulator Board
- Direct Memory Access
  - Multibus Interface for Direct Memory Access Board
  - Direct Memory Access Board
- Microprocessor
  - Multibus Interface for Microprocessor


NOTES:
1. UNLESS OTHERWISE STATED R IN OHMS, C IN pF
2. ALL HCPL 2602 DEVICES BYPASSED WITH 0.01μF CERAMIC DISK CAPACITORS. PIN 8 TO PIN 5
3. S4 AND ASSOCIATED CAPAC. INCORPORATED IN AMPLIFIER/DATA CONV. MODULE #2 ONLY
NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20234
AMPLIFIER / DATA CONVERTER
FOR SAMPLING WATTMETER
EDGE CONN
HCPL 2602
74174
74104
74123
EH1283
MP 271
HCPL-2531
OP-02
AD518
PM901
PM952
VOLTAGE GAIN
● - INDICATES CLOSURE
SHIELD
GUARD

for

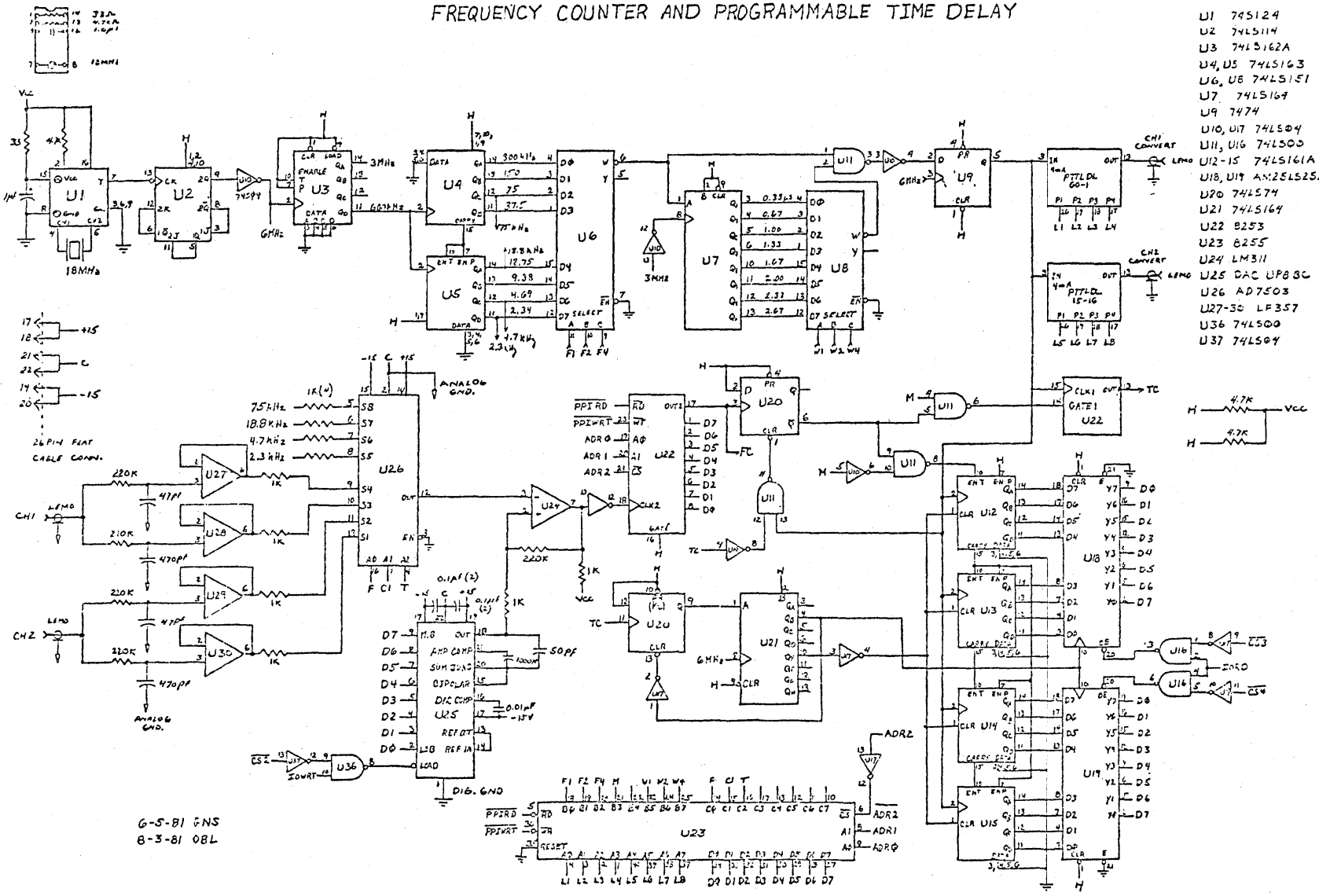
FREQUENCY COUNTER AND PROGRAMMABLE TIME DELAY



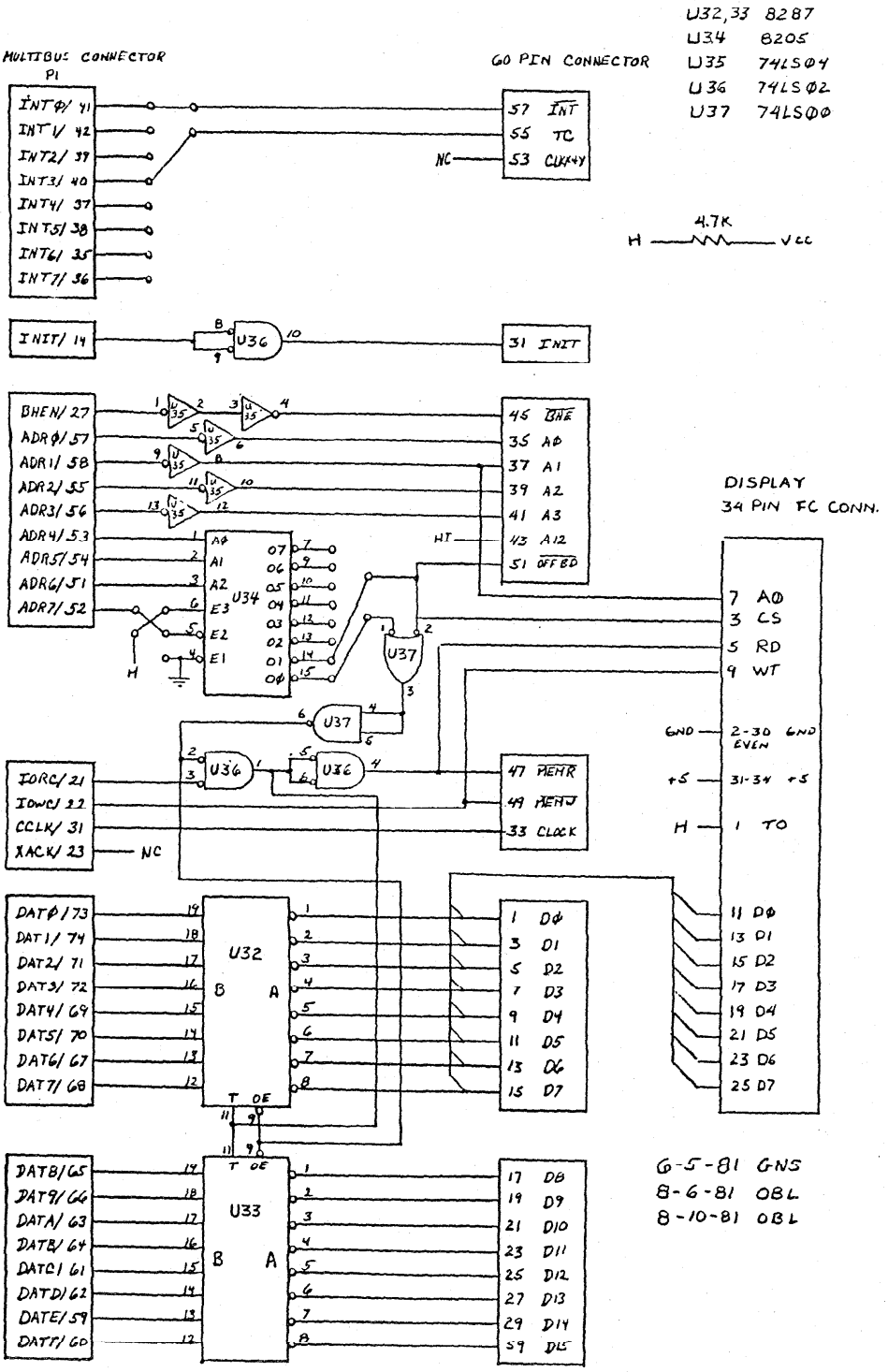
U31 8287
U32 74S00
U33 74S74
U34 8205
U35 74LS125
U36 74LS00
U37, U42 74LS04
U38 74S138
U39 7406
U40 74S02
U41 74LS169
U42 8279
U43 74LS00

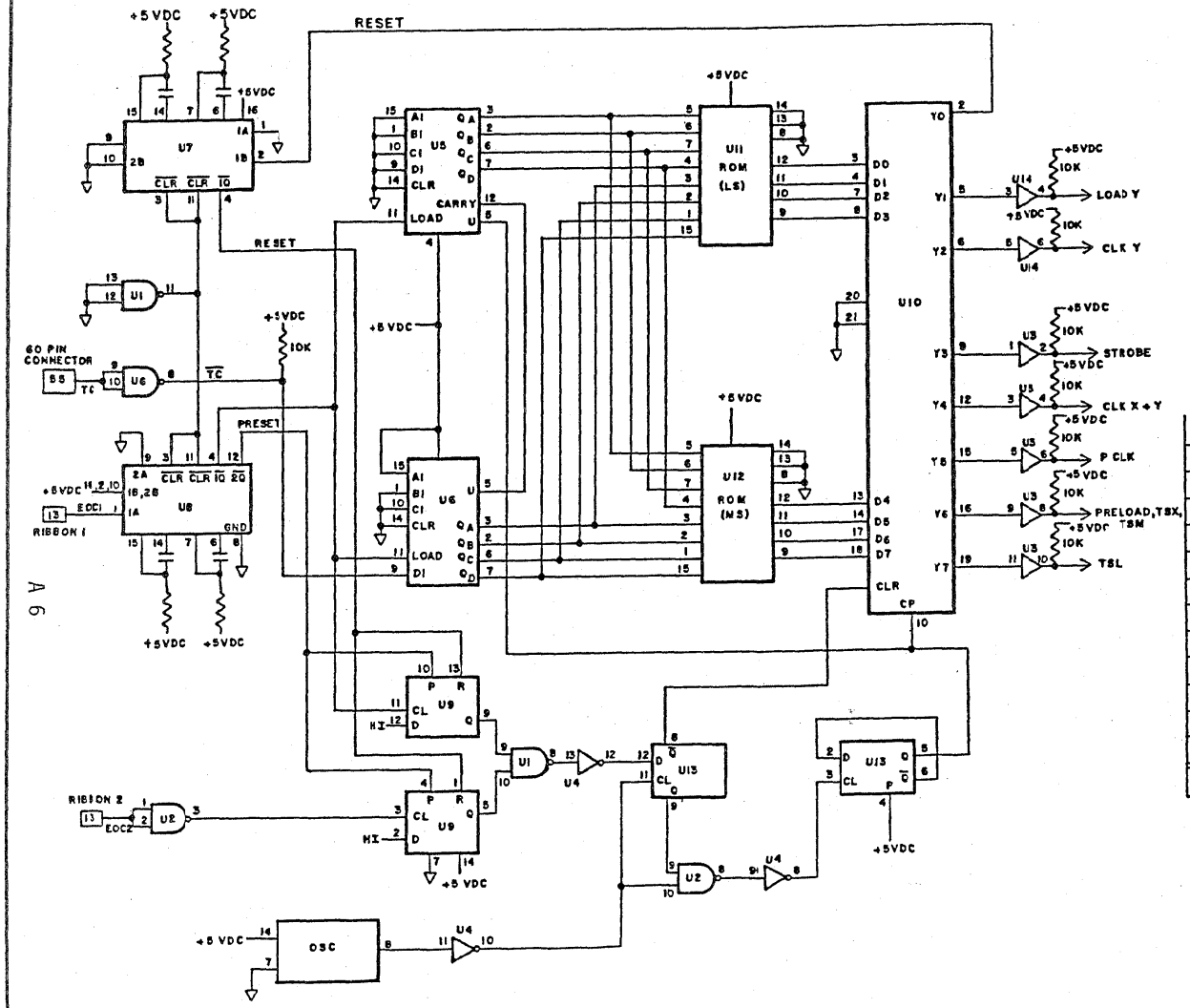
6-5-81 GNS
8-4-81 OBL

FREQUENCY COUNTER AND PROGRAMMABLE TIME DELAY



MULTIBUS I/O INTERFACE
for
MULTIPLIER / ACCUMULATOR





ORIGINAL DATE OF DRAWING 6 JAN 86			
REVISIONS			
NO	LTR	CHANGE	DATE
1			
2			
3			
4			

REF	PART
OSC	10 MHz OSCILLATOR
U11, U12	ROM, INTERSIL
U9, U13	SN7474
U7, U8	74LS123
U10	AM25LS2520
U5, U6	SN74193
U4, U17	SN7404
U3, U14	SN7407
U1, U2	SN7400

PIECE NO	NOMENCLATURE	NO REQ'D

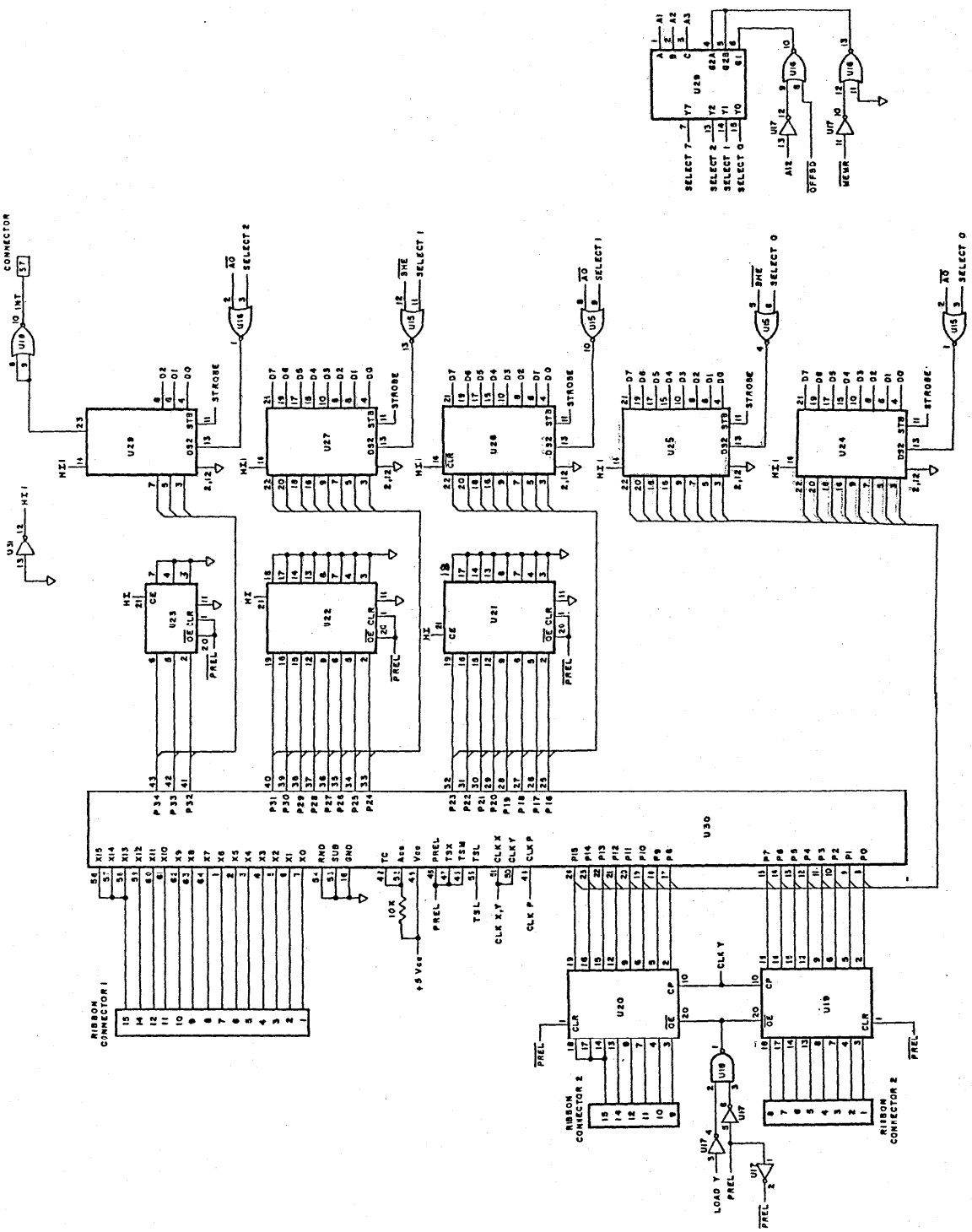
NATIONAL BUREAU OF STANDARDS
WASHINGTON, D.C. 20234

HIGH SPEED SEQUENCER

FOR NUMERICAL INTEGRATION CIRCUITRY

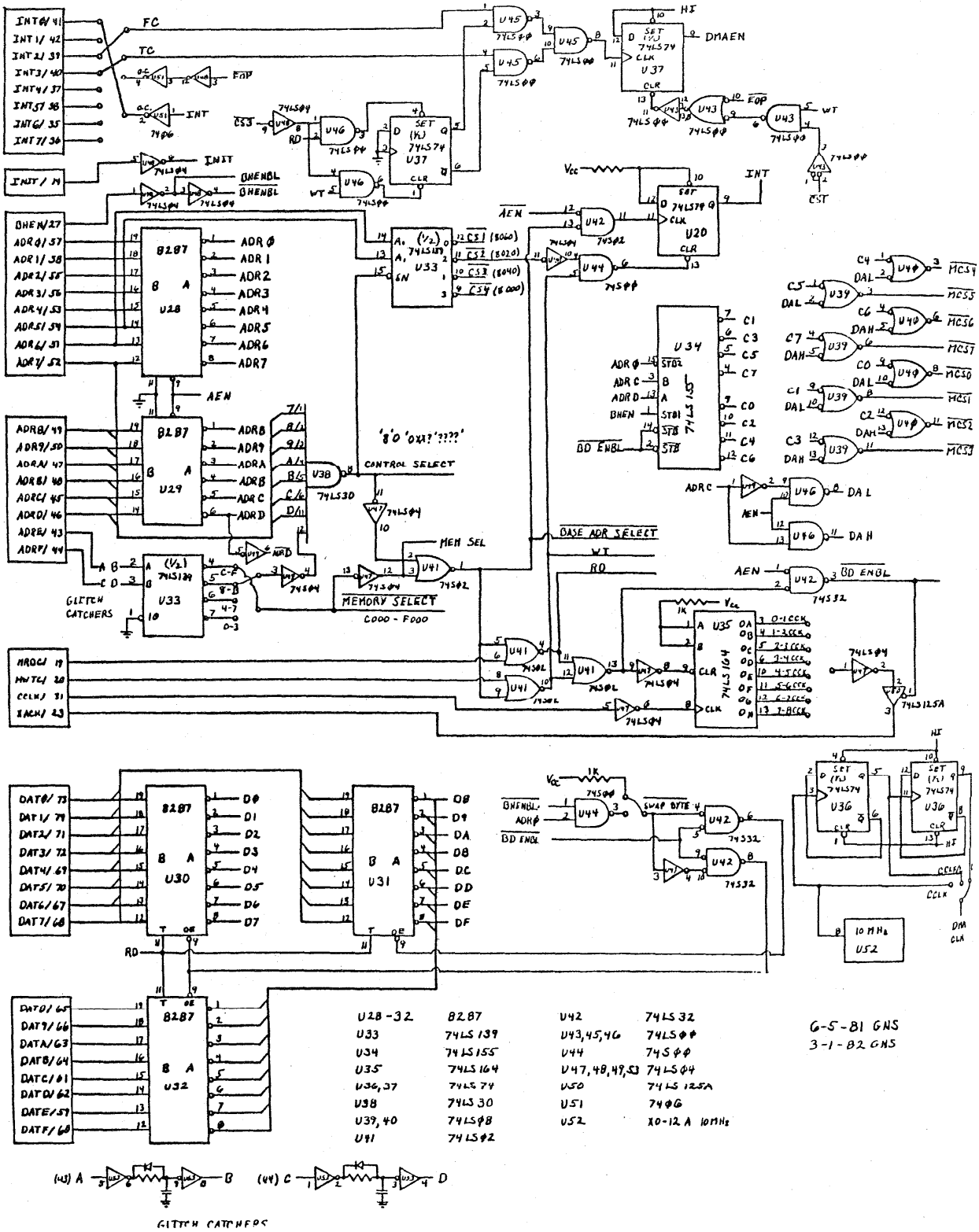
MODEL	TYPE	SCALE
DIMENSIONS IN INCHES / MILLIMETERS (UNLESS OTHERWISE SPECIFIED)	DRAFTSMAN	CHECKED
TOLERANCES (UNLESS OTHERWISE SPECIFIED)	PROJECT ENGR	PROJECT ENGR
DECIMALS ± .XXX / .XX FRACTIONS ± 1/64 ANGLES ± 1°	SUBMITTED BY	CHIEF ENG.
DO NOT SCALE THIS PRINT	EXAMINED BY	CHIEF ENGINEER
DIV. SEC.	THIS PRINT ISSUED	APPROVED BY CHIEF DIV.

A 6



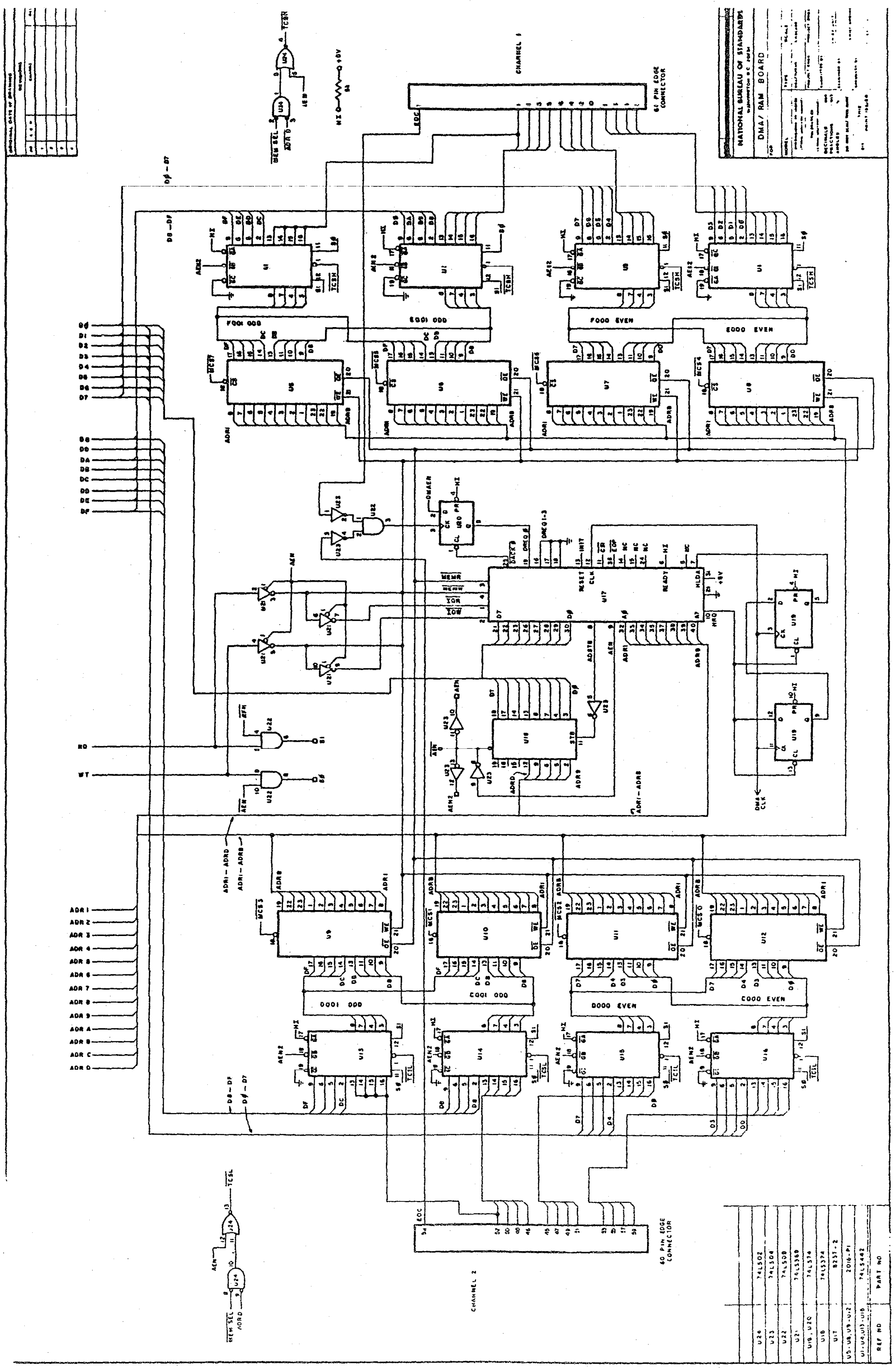
Multiplier-Accumulator Board

MULTIBUS DMA/RAM INTERFACE

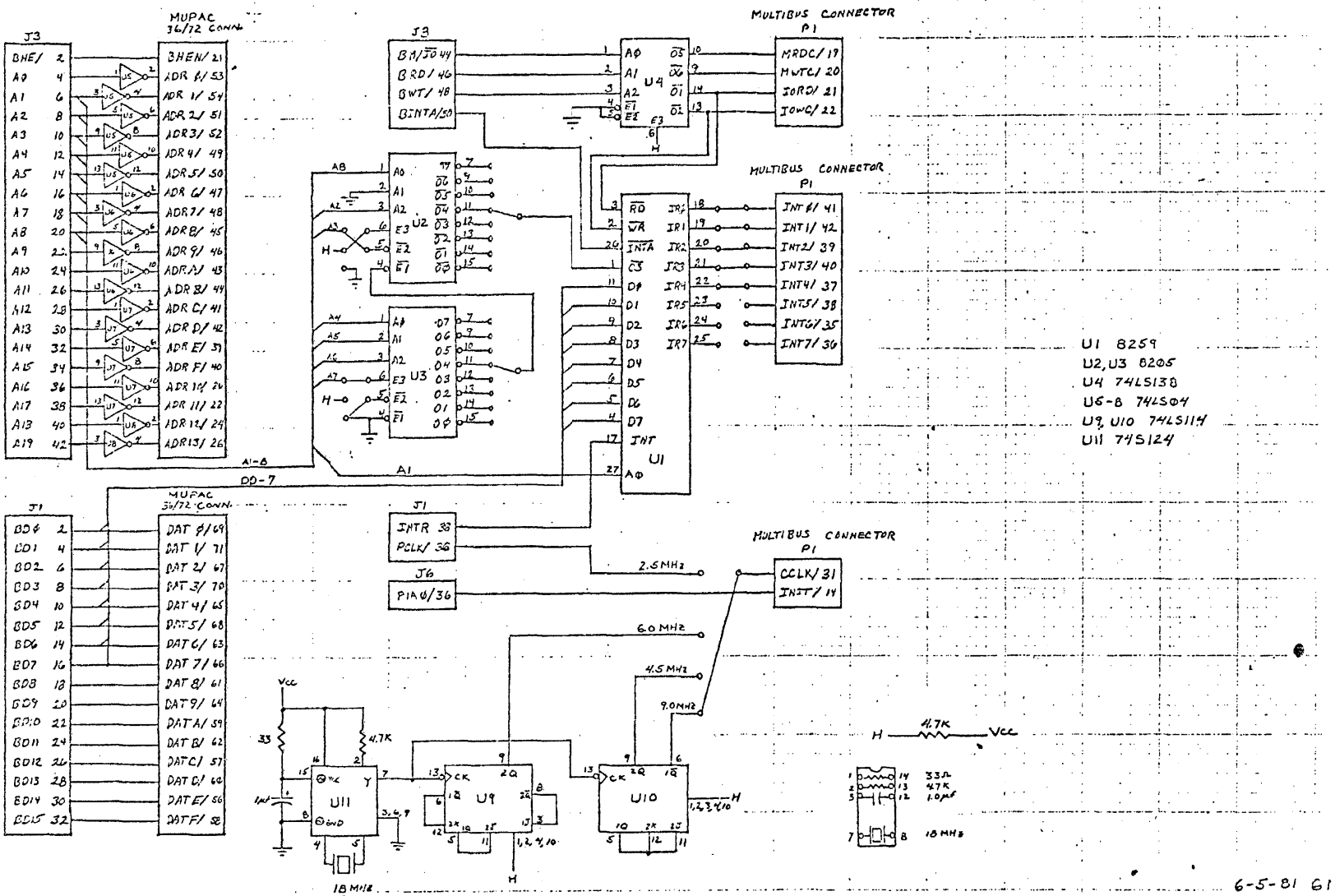


U28-32	8287	U42	74LS32
U33	74LS139	U43,45,46	74LS00
U34	74LS155	U44	74S00
U35	74LS164	U47,48,49,53	74LS04
U36,37	74LS74	U50	74LS12.5A
U38	74LS30	U51	7406
U39,40	74LS08	U52	X0-12 A 10MHz
U41	74LS02		

G-5-81 GNS
3-1-82 GNS



SDK MULTIBUS INTERFACE



6-5-81 GI
8-4-81 OB

Appendix B - Pascal Programs

The Pascal programs were written on a Future Data (TM) microprocessor development system using Pascal-G. This language is very much like standard Pascal with several extensions. Among the extensions used were string variables, string functions, and external variables.

To speed up the compilation process, the programs were separated into three modules which were later linked. In the listing in this appendix the modules are labeled Wattmeter, WattParameters, and Parameters 2. Table B.1. below lists the procedures that are in each module. The type and variable declarations that were common to the three modules were placed in separate files and linked in at compilation time by the "include" statements at the beginning of each module. These two files are listed here as Type P and PVARS.P.

Table B.1.

Procedures in Wattmeter

DoKey	DispC2SS
Disp Pwr	DispDecC
DispSampH	Disp Cor1C
DispFreq	DispCor2C
DispPer	DispTrgC
GetCount	Disp
DispC1AV	TestNSA
DispC1SS	SetDMA
DispC2AV	"Main"

Procedures in WattParameters

SampFreq	SetDispD
PulsWidt	SetR
Multiplx	SetCh1Scale
ConvCmDl	SetCh2Scale
SetIVal	SetShRes
DelaySmp	SetApprxF
SigCycles	

Procedures and Functions in Parameters 2

Invalid	SetDisp3
SQRT	SetDisp4
TrigSet	DisplayE
SetD	DMATrgSr
SetDisp1	DMATrunc

File TYPE.P

```
Type StateTyp = (SHome,SData,SSampPuls,SSampFreq,SPulsWidt,
          SDelay,SSync,SSource,STrigLev,STrigDel,SIntgPer,
          SLockMode,SLockData,SScalFact,SShuntR,SCurGain,
          SVoltGain,SApprFreq,SDisplay,SDispD,SDisp1,
          SDisp2,SDisp3,SDisp4,SDMAPara,SDTrig,SDTrunc,
          SDAve,SRun):;

     DispTyp = (Nothing, DPwr, DFreq, DSampN, DPeriod,
               DC1Ave, DC1SS, DC2Ave, DC2SS,
               DDecCt, DTrgCt, DCor1Ct, DCor2Ct,
               DEnd);

     Interrupt = (IntMA,Int1,IntFC,IntTC,IDMAErr,IntDMA,IntKB,I7);

     SwitchTyp = (OFF,ON);

Const Up = 'A'; Down = 'B'; Left = 'C'; Right = 'D';
          Home = 'E'; Run = 'F';
          CR = Chr(13); LF = Chr(10); OneSec = 1000;
          FUp = 'G'; FDown = 'H'; Dec = '.';
          SoftReset = 'P';

Var [External ARRAYS]  WattInit-ROM UpKey,DownKey,LeftKey,
          RightKey:Array[SHome..SRun] of StateTyp;

Var [External PARAMETERS]  WattInit-RAM SampF,PulseW,
          LockM,MplxSel:Byte; ConvDel,TrigLev:Integer;
          NumDels,NumICycles:Word; PICOCW1:Byte;
          Ch1Strt,Ch2Strt,DMAFr:Word; DMACount,
          DMACCor, DMATrgS:Integer; DMAFlag,DMAEr:Byte;
          Ch1Sum,Ch1SS:Integer4; Ch1SSOv:Integer;
          Ch2Sum,Ch2SS:Integer4; CH2SSOv:Integer;

Var [External ACCVARS]  ReadAcc2 Accum1:Integer4; AccExp,
          NUMSACC:INTEGER; READFLAG:BYTE; SAMPNUM: INTEGER4;
          AccResult:Real; J,K:Integer; IntVar:Interrupt;

Var [External KBVARS]  KBRead KeyFlag:Byte;

Var [External DISPVARS] Blink:SwitchTyp;
```

File PVARS.P

```
Data:Byte;  Key, ErrChr:Char;
State, NewState, OldState:StateTyp;
Message:String[32];
Numeric:Real;  I,T1:Byte;  Intg4:Integer4;
Blanks:String[10]; T2:Integer;
DispK, Disp1, Disp2, Disp3, Disp4:DispTyp;
DelayD, DispDelay:Integer;
PowerScale, CycSampPerSec, RSampF:Real;
DispFlag, DoFrac, TempDoFr, DoneFrac:Byte;
Trys, NumCycles:Integer; TDMACount,
RDMACount, Ch1Scale, Ch2Scale:Real;
ApprxF, ShuntR:Real;
VWhole, VFrac, ExpFrac:Integer4;
VDec, VExp, Digits, MD:Integer;
MDec:Char; MWhole, MFrac:String[8];
MExp:String[1]; RealNum:Real;
Symbols:String[11];
```

WATTMETER - MAIN PROGRAM TO CONTROL WATTMETER
OCT 29, 1981

```
{Udates - Nov. 20, 1981 - Major redraft to interface with
          multiplier-accumulator
          May 28, 1982 - Added DMA operation}


Program Wattmeter;

 {$ITYPE.P}

 {$P  PAGE}


Var [PUBLIC PASCALVARS]
 {$IPVARS.P}


PROCEDURE ResetDisplay;External;
PROCEDURE Position(P:Integer);External;
PROCEDURE WattInit;External;
PROCEDURE IntEnbl(I:Interrupt);External;
PROCEDURE DecStr(Var M:String; Intg4:Integer4); External; {Int->Str}


PROCEDURE SampFreq;External;
PROCEDURE PulsWidt;External;
PROCEDURE Multiplx;External;
PROCEDURE ConvCmD1;External;
PROCEDURE DelaySmp;External;
PROCEDURE SigCycle;External;
PROCEDURE TrigSet; External;
PROCEDURE SetCh1Sc;External;
PROCEDURE SetCh2Sc;External;
PROCEDURE SetShRes;External;
PROCEDURE SetApprx;External;
PROCEDURE SetDisp1;External;
PROCEDURE SetDisp2;External;
PROCEDURE SetDisp3;External;
PROCEDURE SetDisp4;External;
PROCEDURE SetDispD;External;
PROCEDURE DMATrgSr;External;
PROCEDURE DMATrunc;External;

 PROCEDURE RdC1Av ;External;
 PROCEDURE RdC1SS ;External;
 PROCEDURE RdC2Av ;External;
 PROCEDURE RdC2SS ;External;
```

```
PROCEDURE CalFrac1;External;
PROCEDURE CalFrac2;External;
PROCEDURE SDMA   ;External;

PROCEDURE SfPit2; External;{Sets NumICycles}

FUNCTION SQRT( X:REAL ): REAL; External;

PROCEDURE DoKey;

Begin
  Read(Key);
  If Data = 0 Then Begin
     If Key = Up Then NewState := UpKey[State];
     If Key = Down Then NewState := DownKey[State];
  If Key = Left Then NewState := LeftKey[State];
  If Key = Right Then NewState := RightKey[State]
End {Then}

Else Begin
  If Key = Left Then Data := Data + 1;
  If Key = Right Then Data := Data - 1
End;{If}

If NewState = SData Then Begin
  Data := 1; NewState := State
End;{If}

If Key = Home Then Begin
  NewState := SHome;
  Data := 0;
End; {If}

If (Key = FUp) Or (Key = FDown) Then Begin
  WriteLn;
  If State = SRun Then
    If DispFlag = 1 Then DispFlag :=0
    Else DispFlag := 1  Flip Flag

  Else {State <> SRun}

    If Key = FUp Then
      If DispK <> Nothing Then DispK := Pred(DispK)
        Else DispK := Pred(DEnd)
      Else {Key = FDown}

        DispK := Succ(DispK);
    If DispK = DEnd Then DispK := Nothing;
End; {If Key = (FUp Or FDown)}

If Key = Run Then Begin
  NewState := SRun;
```

```
  Data := 0;
  WriteLn;
End; {If}

OldState := State;
State := NewState;
Message := '';
WriteLN;
Case State of
  SHome:          Message := 'TOP OF LIST ';
  SSamPuls:       Message := 'SAMPLE PULSE';
  SSync:          Message := 'SYNCHRONIZATION';
  SIntgPer:       Message := 'INTEGRATION PERIOD';
  SScalFact:      Message := 'SCALE FACTORS';
  SDisplay:       Message := 'DISPLAY';
  SDMAPara:       Message := 'DMA PARAMETERS';
  SData:          Message := 'ILLEGAL STATE - HELP!';
  SRun:           Message := '';

  SSampFreq:      SampFreq;
  SPulsWidth:     PulsWidt;
  SDelay:         ConvCmD1;
  SSource:        Multiplx;
  STrigLev:       TrigSet;
  STrigDel:       DelaySmp;
  SLockMode:      Message := 'LOCK MODE - NIY';
  SLockData:      SigCycle;
  SShuntR:        SetShRes;
  SCurGain:       SetCh1Sc;
  SVoltGain:      SetCh2Sc;
  SApprFreq:      SetApprx;
  SDispD:         SetDispD;
  SDisp1:      SetDisp1;
  SDisp2:      SetDisp2;
  SDisp3:      SetDisp3;
  SDisp4:      SetDisp4;
  SDTrig:      DMATrgSr;
  SDTrunc:     DMATrunc;
  SDAve:       Message := 'DMA Ave';

  ELSE: {CASE}Message := 'WHAT STATE IS THIS?'
 End; {Case}

 If Length(Message)>0 Then Write(Message);
 DelayD := DispDelay; {Terminate Display Delay While}

End; {DoKey}


PROCEDURE DispPwr;

Begin
```

```
  Numeric := Accum1;
  For I := 1 to AccExp Do
    Numeric := Numeric * 256;
  Numeric := Numeric/SampNum;
  Numeric := (Numeric) * PowerScale;
  Write('P=',Numeric:9:5,'W');
End; {DispPwr}^R


PROCEDURE DispSampN;

Begin
  DecStr(Message, SampNum);
  Write('SN = ',Message,'')'
End; {DispSampN}


PROCEDURE DispFreq;
{Note This won't work if Lock Mode = Samp Num}


Begin
  Numeric := CycSampPerSec/SampNum;
  {NumICycles * NumSAcc * RSampF/SampNum}

  Write('F=', Numeric:8:4,'Hz');
End; {DispFreq}


PROCEDURE DispPer;
{Note This won't work if Lock Mode = Samp Num}


Begin
  Numeric := SampNum/CycSampPerSec;
  {SampNum/(NumICycles * NumSAcc * RSampF)}

   Write('Pe=',Numeric:8:4,'s');
End; {DispPer}


PROCEDURE GetCount;

Begin
  If DoneFrac = 0 Then Begin
     DoneFrac := 1;
     ErrChr := '';
     Case DoFrac of
       0    : RDMACount := DMACount;
       1..2 : Begin
                If DoFrac = 1 Then CalFrac1
```

```
                Else CalFrac2;
              If DMAEr = 0 Then Begin
                  DMACount := DMACount + DMACCor;
                  RDMACount := DMACount + DMAFr/65536
              End {If Then}

              Else Begin
                  RDMACount := DMACount;
                  ErrChr := 'E';
              End; {If Else}

            End; {Case 1..2}
      3     : Begin
              DMACount := TDMACount;          {Truncate real to integer}
              DMAFr := TMDACount - DMACount; {Difference between real}
              RDMACount := TMDACount;               {and integer}
            End; {Case 3}
    End; {Case}
  End; {If}
End; {GetCount}


PROCEDURE DispC1Av;

Begin
  GetCount;
  RdC1Av;
  Numeric := Ch1Sum * Ch1Scale/RDMACount;
  Write ('Av1=', Numeric:7:4,'V', ErrChr)
End; {DispC1Av}


PROCEDURE DispC1SS;

Begin
  GetCount;
  RdC1SS;
  Numeric := Ch1SSOv;
  Numeric := Numeric * 8.5899346E9;  {6#80000000 * 4 ?}

  Numeric := Numeric /RDMACount;
  Numeric := SQRT(Numeric) * Ch1Scale;
  Write('Rm1=', Numeric:7:5,'V', ErrChr)
End; {DispC1SS}


PROCEDURE DispC2Av;

Begin
  GetCount;
  RdC2Av;
  Numeric := Ch2Sum 8 Ch2Scale/RDMACount;
```

```
  Write('Av2=', Numeric:7:4,'V', ErrChr)
End; {DispC2Av}


PROCEDURE DispC2SS;

Begin
  GetCount;
  RdC2SS;
  Numeric := CH2SS0v;
  Numeric := Numeric * 8.5899346E9; {16#80000000 * 4 ?}

  Numeric := Numeric + Ch2SS;
  Numeric := Numeric /RDMACount;
  Numeric := SQRT(Numeric) * Ch2Scale;
  Write('Rm2=', Numeric:7:5,'V', ErrChr)
End; {DispC2SS}^R



PROCEDURE DispDecC;

Begin
  DecStr(Message, DMACount);
  Write('DecC= ', Message);
End; {DispDecC}


PROCEDURE DispCor1C;

Begin
  TempDoFr := DoFrac;
  DoFrac := 1; DoneFrac :=0;
  GetCount;
  Write('C1C=', RDMACount:8:6, ErrChr);
  DoFrac := TempDoFr;
End; {DispCor1C}


PROCEDURE DispCor2C;

Begin
  TempDoFr := DoFrac;
  DoFrac := 2; DoneFrac := 0;
  GetCount;
  Write('C2C=', RDMACount:8:6, ErrChr);
  DoFrac := TempDoFr;
End; {DispCor2C}


PROCEDURE DispTrgC;
```

```
Begin
  Write('TrC=', TDMACount:9:6);
end; {DispTrgC}


PROCEDURE Disp(P:Byte; DispV:DispTyp);

Begin
  Position(P);
  Case DivpV of
    Nothing: Write ('       ');
    DPwr   : DispPwr;
    DFreq  : DispFreq;
    DSampN : DispSampN;
    DPeriod: DispPer;
    DC1Ave : DispC1Av;
    DC1SS  : DispC1SS;
    DC2Ave : DispC2Av;
    DC2SS  : DispC2SS;
    DDecCt : DispDecC;
    DTrgCt : DispTrgC;
    DCor1Ct: DispCor1C;
    DCor2Ct: DispCor2C;

    Else   : Write('Display Err')'
  End; {Case}

End; {Disp}


PROCEDURE TestNSA;
{Adjust Number of Software Accumulations}


BEGIN
   {Set number of software accumulations such that the hardware
    accumulations are greater than 1.2ms and less than 3.6ms.
    Here Numeric = time for hardware accumulations.}

  NUMERIC := SAMPNUM/(RSAMPF*NUMSACC);
  IF ((NUMERIC < 1.2E-3) AND (NUMSACC>1)) OR (NUMERIC > 3.6E-3) THEN
    BEGIN
      INTG4 := NUMICYCLES * NUMSACC; {Num of input cyc in intg per}

      NUMERIC := NUMICYCLES * 1.6E-3/NUMERIC; {Num cyc in 1.6ms}

      NUMICYCLES := 2;
      WHILE NUMICYCLES < NUMERIC DO NUMICYCLES := NUMICYCLES *2;
        {Change num of cycles by powers of 2}

      NUMSACC :- INTG4 DIV NUMICYCLES;
      SfPit2; {Set new NumICycles}
```

```
      CycSampPerSec := NumICycles * NumSAcc * RSampF;
   END; {IF}

END; {TESTNSA}


PROCEDURE SetDMA;
{Set DMA parameters and start DMA and Mult-Acc}


Begin {SetDMA}

  If SampNum < 4000 Then Begin
     DMACount := SampNum:
     TDMACount := DMACount;
   End {If Then}

   Else Begin
     Numeric := SampNum/(NumICycles * NumSAcc);
     {Number of samples per input cycle}

     NumCycles := 4000/Numeric; {Truncate to num of whole cycles}

     TDMACount := NumCycles * Numeric;
     DMACount := TDMACount; {Trigger derived DMA count}

   End;  If Else

 Trys := 0;
 Repeat {Make sure MA intr does not occur during this sequence}

   ReadFlag := 1;
   IntEnbl(IntMA);
   SDMA; {Send parameters to DMA controller}

   DMAFlag := 1;
   IntEnbl(IntDMA);
   Trys := Trys + 1;
Until (ReadFlag = 1) OR (Trys = 10); {Give up after 10 Trys}

While (ReadFlag = 1) and (KeyFlag = 0) Do;
{Wait for interrupt to start DMA process}

ReadFlag := 1;
{Reset MA so MA data is taken at the same time as the DMA data}

IntEnbl(IntMA);
DoneFrac := 0;
End; {SetDMA}

{$P}
```

```
Begin{Wattmeter}

  WattInit;
  ResetDisplay;
  ReadFlag := 0;
  KeyFlag  := 0;
  DMAFlag  := 0;
  DMATrgS  := 0; {Set to Trg on TC}

DoFrac := 0; {No fraction calculation}

Ch1Scale := 5/2048;
Ch2Scale :=5/2048;
ApproxF := 1000;
ShuntR := 1;
Ch1Strt := 16#0E000;
Ch2Strt := 16#0C000;
Symbols := 'afpnum kMGT';
DispFlag := 0;
DispDelay := OneSec;
DispK := Nothing;
Disp1 := DPwr;
Disp2 := DSampN;
Disp3 := DFreq;
Disp4 := DPeriod;
SampNum := 1000;
RSampF := 300000;
NumSAcc := 2;
CycSampPerSec := NumICycles * NumSAcc * RSampF;
PowerScale := Ch1Scale*Ch2Scale/ShuntR;
Data := 0;
Blink := OFF;
Write('Top of List ');
Blanks := '          ';
State := SHome;
NewState := SHome;
Message := '';
IntEnbl(IntKB);
While True Do  Forever Begin
  {Set start of display delay loop to no. of ms for
    one integration period}

  DelayD := 1000 * SampNum/RSampF;
  {KeyFlag check for no display delay}

  If KeyFlag > 0 Then DoKey;
  While DelayD < DispDelay Do Begin
    For I := 0 to 62 Do; {Wait 1 ms}

    DelayD := DelayD + 1;
```

```
    If KeyFlag > 0 Then DoKey;
  End; {While}

    If State = SRun Then Begin
    SetDMA;
      While (ReadFlag = 1) And (KeyFlag = 0) Do; {Wait}

      If ReadFlag = 0 Then Begin
        If DispFlag = 0 Then Begin
          Disp(0,Disp1);
          Disp(15,Disp2);
        End {If}

        Else Begin
          Disp(0,Disp3);
          Disp(15,Disp4);
        End; {Else}

        TestNSA;
    End; {If}

  End {Then State = SRun}

  Else Begin {State <> SRun}

    If DispK <> Nothing Then Begin
      SetDMA;
      While (ReadFlag = 1) And (KeyFlag = 0) Do; {Wait}

      If ReadFlag = 0 Then Begin
        Disp(19,DispK);
       TestNSA;
    End; {IfReadFlag = 0}

  End; {If DispK <> Nothing}

 End; {Else State <> SRun}

End; {While Forever}

End. {Wattmeter}
```

WattParameters - Collection of Routines to Change
Parameters of Wattmeter
Nov. 3, 1981

```
{$V-}{REMOVE ARITHEMETIC CHECKS}

MODULE WattParameter;
{$ITYPE.P}


Var [External PASCALVARS]
{$IPVARS.P}


PROCEDURE SfppipA; External;{Sets Converter Command Delay}
PROCEDURE SfppipB; External;{Sets SampF, LockM, and PulsWidt}
PROCEDURE SfppipC; External;{Sets Multiplexer Selection}
PROCEDURE SfPit1;  External;{Sets NumDels}
PROCEDURE SfPit2;  External;{Sets NumICycles}

PROCEDURE SFDAC;   External;{Sets Trigger Level to Sel Value}
PROCEDURE SFDAC2;  External;{Sets Trigger Level to 2 Volts}

PROCEDURE DecStr (Var M:String; Intg4:Integer4); External;{Int->Str}
PROCEDURE SampFreq; Public;

Begin
  WriteLN;
  Write('Smp Frq=');
  If Data = 2 Then Data := 1; {Limit Data to < 2}

  If Data = 1 Then Begin
     If (Key = Down) and (SampF < 7) Then SampF := SampF + 1;
     If (Key = Up) and (SampF > 0, Then SampF := SampF - 1;
     SfppipB;{Reset SampF}

     Write(CHR(62));
  End;{If}

  RSampF := 300000;
  For I := 1 to SampF Do RSampF := RSampF/2;
  CycSampPerSec := NumICycles * NumSAcc * RSampF;
  Write(RSampF:6:3);
  Write('Hz  ');
  Message := ''
End;

PROCEDURE PulsWidt; Public;
```

```
Begin
  WriteLN;
  Write(Pls Wdt=');
  If Data = 2 Then Data := 1; {Limit Data to < 2}

  If Data = 1 Then Begin
    If (Key = Down) and (PulseW > 0) Then PulseW := PulseW - 1;
    If (Key = Up) and (PulseW < 7) then PulseW := PulseW + 1;
    SfppipB; {Set new Pulse Width}

    Write(CHR(62)); {Display > }

  End;{If}

  T2 := PulseW;{Correct Problem of Byte to Float Convertion}

  T2 := PULSEW; {CORRECT STORAGE PROBLEM}

  Numeric := 0.3333333 + T2/3;
  If PulseW < 3 Then Begin

   Numeric := Numeric * 1000;
   Message := 'ns  ';
End {Then}

Else Message := 'us ';
Write(Numeric:6:4);
Write(Message);
Message := ''
End;

PROCEDURE Multiplx; Public;

Begin
  WriteLN;
  Write('Mplx =');
  If Data = 2 Then Data := 1; { Limit Data to < 2}

  If Data = 1 Then Begin
     If (Key = Down) Then
       If MplxSel > 0 Then MplxSel := MplxSel - 1
          Else MplxSel := 7;
     If (Key = Up) Then
       If MplxSel < 7 Then MplxSel := MplxSel + 1
          Else MplxSel := 0;
   SfppipC; {Set new Multiplex Selection}

   If MplxSel >3 Then SFDAC2 {Set Trigger Level to 2 volts}
      Else SFDAC; {Set Trigger Level to Selected Value}
   Write (CHR(62)); {Display Blinking >}
End;{If}
Case MplxSel of
```

```
  0: Write('Ch 2 LF');
  1: Write('Ch 2 HF');
  2: Write('Ch 1 LF');
  3: Write('Ch 1 HF');
  4: Write ('2.344 kHz');
  5: Write('4.688 kHz');
  6: Write('18.75 kHz');
  7: Write('75.00 kHz');
End; {Case}
Message := ''
End;

PROCEDURE ConvDmD1; Public;

Begin
 WriteLN;
 Write('Conv Dly =');
 If Data = 3 Then Data := 2; {Limit Data to < 3}
 If Data = 2 Then I := 1;
 If Data = 1 Then I := 10;
 If Data > 0 Then Begin
    If Key = Down Then ConvDel := ConvDel - I;
    If Key = Up Then ConvDel := ConvDel + I;
    If ConvDel < -60 Then ConvDel := -60;
    If ConvDel > 195 Then ConvDel := 195;
    SfppipA; {Set New Converter Delay}
 End;{If}
Intg4 := ConvDel;
DecStr(Message, Intg4);
Message := '  ' + Message;
Message := Message[Length(Message)-3..Length(Message)];
If Data = 1 Then Message := Message [1..Length(message)-2] +
    CHR(62) + Message[Length(Message)-1..Length(Message)];
   If Data = 2 Then Message := Message[1..Length(Message)-1] +
      CHR(62) + Message[Length(Message)];
   Write(Message);
   Write('ns  ');
   Message :=''
End;

 PROCEDURE SetIVal(Var IVal:Word; Mult:Word; Min:Byte); Public;
 {Inc/Dec Value of IVal and Writes to Display}

 Begin
   If Data > 5 Then Data := 5;
   T2 := 10000 Div Mult;
   For I := 2 To Data Do T2 := T2 Div 10;
   If T2 < 1 Then T2 := 1;
   Intg4 := IVal;
   If Data > 0 Then Begin
     If Key = Down Then Intg4 := Intg4 - T2;
     If Key = Up Then Intg4 := Intg4 + T2;
```

```
    If Intgr > 65535 Then Intg4 := 65535;
    If Intg4 < Min Then Intg4 := Min;
    IVal := Intg4;
  End;
DecStr(Message,Intg4 * Mult);
Message := Blanks[1..6-Length(Message)] + Message;
If Data > 0 Then Message := Message[1..Data] + CHR(62)
  + Message[Data+1..6];
Write(Message);
End;

PROCEDURE DelaySmp; Public;

Begin
  WriteLN;
  Write('Trg Dly=');
  SetIVal(NumDels,1,1);
  SfPit1;
  Write('Sm');
End;

PROCEDURE SigCycles; Public;

Begin
  WriteLN;
  Write('Int Per=');
  If NumSAcc > 1 Then SetIVal(NumSAcc, NumICycles, 1)
    Else SetIVal(NumICycles, NumSAcc, 2);
  CycSampPerSec := NumICycles * NumSAcc * RSampF;
  SfPit2;
  Write('Cy');
End;

PROCEDURE SetDispD; Public;

Begin
  WriteLN;
  Write('Disp D= ');
  SetIVal(DispDelay,1,1);
  Write('ms');
End; {SetDispD}


PROCEDURE SetR(Var RealNum:Real; DLimit:Integer);

Begin
  If Data = 1 Then Begin {First entry to change #}

    Data := 3;
    Digits := 0;
    VWhole := 0;
    VFrac := 0;
```

```
    VDec := 0;
    VExp := 0;
    MFrac := '';
    MDec := ' ';
    ExpFrac := 1;
  End; {If Data = 1}
If Data = 2 Then Begin {Exit entry of new #}

  RealNum := VWhole + VFrac / ExpFrac;
  While VExp > 0 Do Begin
    RealNum := RealNum * 1000;
    VExp := VExp - 1;
  End; {While VExp > 0}

  While VExp < 0 Do Begin
    RealNum := RealNum / 1000;
    VExp := VExp + 1;
  End; {While VExp < 0}

  Data := 0;
End; {Data = 2}


If Data > 3 Then Data := 3; {Limit Data to 3}

If (Data = 3) And (Digits < DLimit) Then Begin
  If Key In ['0'..'9'] Then Begin
    MD := Ord(Key) - Ord('0');
    Digits := Digits +1;
    If VDec = 0 Then VWhole := MD + 10 * VWhole
      Else {VDec = 1 Begin}
        VFrac := MD + 10 * VFrac;
        ExpFrac := ExpFrac * 10;
      End; {Else VDec = 1}

End; {Data = 3 and Digits < DLimit}


If Data = 3 Then Begin
  If Key = Dec Then Begin
    VDec := 1;
    MDec := '.';
  End; {If Key = Dec}

  If Key = Up Then
    If VExp < 3 Then VExp := VExp + 1;
  If Key = Down Then
    If VExp > -6 Then VExp := VExp - 1;
  MExp := Symbols[VExp + 7];
  DecStr(MWhole, VWhole);
  If ExpFrac > 1 Then Begin
    DecStr(MFrac, ExpFrac + VFrac);
```

```
    MFrac := MFrac[2..Length(MFrac)];
  End; {If ExpFrac > 1}

   Write('>', MWhole, MDec, MFrac, MExp);
End; {Data = 3}


 If Data = 0 Then Write(RealNum:10:DLimit);

End; {SetR}


PROCEDURE SetCh1Scale; Public;

Begin
 WriteLN;
 Write('Ch1 SF=');
 SetR(Ch1Scale,7);{also Write out value}

 PowerScale := Ch1Scale*Ch1Scale/ShuntR;
End; {SetCh1Scale}


PROCEDURE SetCh2Scale; Public;

Begin
  WriteLN;
  Write('Ch2 SF=');
  SetR(Ch2Scale,7); {also Write out value}

  PowerScale := Ch1Scale*Ch2Scale/ShuntR;
End; {SetCh2Scale}


PROCEDURE SetShRes; Public;

Begin
  WriteLN;
  Write('ShuntR=');
  SetR(ShuntR,7); {also Write out value}

  PowerScale := Ch1Scale*Ch2Scale/ShuntR;
End; {SetShRes}


PROCEDURE SetApprxF; Public;

Begin
  WriteLN;
  Write('ApprxF=');
  SetR(ApprxF,6); {also Write out value}
```

```
End; {SetApprxF}

.{Module Terminator}
```

Parameters 2 - Collection of Routine to Change
Parameters of Wattmeter
Nov. 13, 1981

```
{$V-}{REMOVE ARITHMETIC CHECKS}

{Updates - Nov. 24, 1981 - Added Display Routines
           May 29, 1982 - Added DMA operation}

MODULE Para2;
{$ITYPE.P}


Var [External PASCALVARS]
{$IPVARS.P}

{$P}{PAGE}


PROCEDURE SfppipA; External;{Sets Converter Command Delay}

PROCEDURE SfppipB; External;{Sets SampF, LockM, and PulsWidt}

PROCEDURE SfppipC; External;{Sets Multiplexer Selection}

PROCEDURE SfPit1;  External;{Sets NumDels}

PROCEDURE SfPit2;  External;{Sets NumICycles}

PROCEDURE SfDac;   External;{Sets Trigger Level}

PROCEDURE InitDMA; External;{Sets DMA Trigger Source}


PROCEDURE Position(P:Integer); External;
PROCEDURE DecStr(Var M:String; Intg4:Integer4); External; Int->Str


PROCEDURE SReset; External; {Does an INTR RETURN}


{FROM REAL MATH PACKAGE}


procedure exponent(var r:real; var ev : integer1); external;
procedure addexpon(var r:real; ev : integer1);external;

Function invalid : real;
 var dum : record
```

```
          i4 : integer4;
          reel : real
         end;
begin
  dum.i4:=-1;
  invalid :=dum.reel;
end;

FUNCTION SQRT ( X:REAL ); REAL;
PUBLIC;
CONST FRT8M1S=0.46484146; {8**(1/4)-1}

      SQRT2 = 1.4142135623;
VAR   I,EV,EV1 : INTEGER1;
      YC:REAL;

BEGIN
  IF X=0.0 THEN SQRT:=0.0 ELSE
  IF X<0.0 THEN SQRT:=INVALID ELSE
  BEGIN
    EXPONENT(X,EV); {Reduce X TO [.5,1]}

      YC:=(X/2.0)+FRT8M1S;
      FOR I:=1 TO 3 DO
        YC:=(X/YC+YC)/2.0;
    ADDEXPON(YC,EV DIV 2); {YC*2**(EV/2)}

    IF ODD(EV) THEN IF EV>0 THEN YC:=YC*SQRT2
                    ELSE YC:=YC/SQRT2;
    SQRT:=YC;
  END;
End  {SQRT};

PROCEDURE TrigSet; Public;

Begin
  Write('Trg L=');
  If Data = 3 Then Data := 2; {Limit Data to <3}

  I := 16;
  If Data = 2 Then Begin;
    I := 1;
    Write(CHR(62));{Display 2 >'s for Data=2}

End;{If}

If Data > 0 Then Begin
   If Key = Down Then TrigLev := TrigLev - I;
   If Key = Up Then TrigLev := TrigLev + I;
   If TrigLev < 0 Then TrigLev :=0;
   If TrigLev > 255 Then TrigLev := 255;
   SfDac:{ Reset TrigLev}
```

```
   Write(CHR(62)); {a >}
End;{If}

Numeric := (Triglev - 128)*0.0390625;
If (Numeric < 1.0) and (Numeric > -1.0) Then Begin
   Numeric := Numeric * 1000;
   Message := 'mv'
End {Then}

  Else Message := 'V  ';
Write(Numeric:6:4);
Write(Message);
End;

PROCEDURE SetD(Var DispV:DispTyp); Public;

Begin
  If Data > 1 Then Data := 1;
  If Data = 1 Then Begin
     Write(CHR(62)); {a >}
     If Key = Up Then
       If DispV <> Nothing Then
         DispV := Pred(DispV)
       Else DispV := Pred(DEnd);
     If Key = Down Then
       DispV := Succ(DispV);
     If DispV = DEnd Then DispV := Nothing;
End; {If}

Case DispV of
  Nothing: Write('Nothing ');
  DPwr   : Write('Power   ');
  DFreq  : Write('Freq    ');
  DSampN : Write('Smp Num ');
  DPeriod: Write('Period  ');
  DC1Ave : Write('Ch1 Ave ');
  DC1SS  : Write('Ch1 rms ');
  DC2Ave : Write('Ch2 Ave ');
  DC2SS  : Write('Ch2 rms ');
  DDecCt : Write('Dec Cnt ');
  DTrgCt : Write('Trg Cnt ');
  DCor1Ct: Write('Cor1Cnt ');
  DCor2Ct: Write('Cor2Cnt ');

  Else   : Write('How Now ');
End; {Case}

Message := '';
End; {SetD}
```

```
PROCEDURE SetDisp1; Public;

Begin
  Write('Disp1 = ');
  SetD(Disp1);
End; {SetDisp1}


PROCEDURE SetDisp2; Public;

Begin
  Write('Disp2 = ');
  SetD(Disp2);
End; {SetDisp2}


PROCEDURE SetDisp3; Public;

Begin
  Write('Disp3 = ');
  SetD(Disp3);
 End; {Setdisp3}


PROCEDURE SetDisp4; Public;

Begin
  Write('Disp4 = ');
  SetD(Disp4);
End; {SetDisp4}


PROCEDURE DisplayE (Error:integer); Public;

VAR Error4:integer4;

Begin
  Error4 := Error;
  WriteLN;
  Write('Math Error at ',Error4:6);
  While KeyFlag = 1 Do Read(Key); {Clear Key Register}

  While KeyFlag = 0 Do; {Wait for key entry}

  Read(Key);
  If Key = SoftReset Then SReset; {Else do full Reset}

End;  DisplayE


PROCEDURE DMATrgSr; Public;
```

```
Begin
  Write('DMA Trg S = ');
  If Data = 2 Then Data := 1; {Limit Data to < 2}

  If Data = 1 Then Begin
  If (Key=Up) OR (Key=DOWN) Then Begin
     DMATrgS := DMATrgS + 1;
    If DMATrgS > 1 Then DMATrgS := 0;
    InitDMA;
  End; {If Key}

  Write('>');
 End; {If Data = 1}

 If DMATrgS = 0 Then Write('TC')
     Else Write('FC');
End; {DMATrgSr}


PROCEDURE DMATrunc; Public;

Begin
  Write('DMA Trnc=');
  If Data = 2 Then Data := 1; {Limit Data to < 2}

  If Data = 1 Then Begin
    If Key = Up Then
      If DoFrac > 0 Then DoFrac := DoFrac - 1
        Else DoFrac := 3;
    If Key = Down Then
      If DoFrac < 3 Then DoFrac := DoFrac + 1
        Else DoFrac := 0;
   Write ('>');
End;{If Data = 1}

Case DoFrac of
   0: Write('None');
   1: Write('Ch1 Cor');
   2: Write('Ch2 Cor');
   3: Write('Trigger');
  End; {Case}

End; {DMATrunc}

.{Module Terminator}
```


* U.S.Government Printing Office: 1987 0-181-076 (60086)