A MINIMUM CONFIGURATION MICROPROCESSOR BASED DATA ACQUISITION, POSTPROCESSING AND REPORTING SYSTEM

P.J. Zsombor-Murray
DATAC Computer Laboratory, Department of Mechanical Engineering,
McGill University, Montréal

Abstract

In order that engineers may effectively exploit microprocessor technology, the teaching of fundamental informatique should assume a number of new dimensions, beyond high level language programming, so as to include or anticipate the following: (1) Familiarization of all engineering students with the macrologic of microcomputers. (2) Hardware design as regards interfacing and system integration. (3) Low level programming. (4) Laboratory exercises making use of microprocessor based instrumentation and control and, (5) Design projects involving machinery or industrial processes which include microprocessor systems.

This proposition is developed in the context of a simple application (from the point of view of informatique). It is emphasized that the low cost and high availability and performance of microcomputer components tend to shift the onus of system design from informatique to the particular application discipline.

UN SYSTÈME MINIMUM DE MICROPROCESSEUR POUR L'ACQUISITION, LE TRAITEMENT ET LE RAPPORTAGE DES DONNÉES EXPÉRIMENTALES

Résumé

De sorte que les ingénieurs peuvent exploiter effectivement des microprocesseurs on suggère que le cadre de cours en informatique fondamentale acquérira quelques dimensions nouvelles. Au delà de la programmation en langages d'un haut niveau, on doit considérer ou bien faire préparation aux sujets suivants: (1) Que tous les étudiants en génie devriont apprendre la macrologique des microprocesseurs. (2) La conception de «hardware» qu'appartient aux entrefaces et à l'integration des systèmes. (3) La programmation de bas niveau. (4) Les expériences de laboratoire dans toutes les disciplines de science appliquée qui s'intéressent aux appareils de la mesure et du contrôle automatisés par microprocesseur et, (5) Les projets de conception des machines et des processus industriels qui comprennent les microordinateurs.

On développe cette proposition par un exemple, très simple au point de vue d'informatique. On fait ressortir que les composants des microordinateurs ne coûtent pas chèrs, sont en disponibilité immédiate et sont presque aussi puissants que les miniordinateurs, mais sans le même complexité. Pour ces raisons ci, la compétance en conception des systèmes de microordinateur souvant n'échoit pas à l'informatique mais aux disciplines des applications spécifiques.

A MINIMUM CONFIGURATION MICROPROCESSOR BASED DATA ACQUISITION, POSTPROCESSING AND REPORTING SYSTEM

by

P.J. Zsombor-Murray
DATAC Computer Laboratory
Department of Mechanical Engineering
McGill University
Montreal, CANADA
H3A 2K6

INTRODUCTION

Microprocessor applications hold the solution to many engineering problems. Attention will be given to simple techniques and problems of some industrial relevance. These will include topics suited to a general course on microprocessors. It is felt that all engineers must include some digital logic, software and microprocessor applications in their design training. This is not to produce computer designers but to provide for intelligent specification and selection of computer systems, within all branches of engineering.

Presently, few engineers in industry choose to design and construct entire computer systems, however simple. The prevailing attitude relegates this activity to specialists in informatique. It is hoped that suitable education might impart the same objectivity, with regard to how best to automate some phase of industrial activity, as is used to decide whether or not to design and build a piece of specialised production machinery in-house.

Three arguments are put forth in support of a general compulsory course requirement in microprocessor applications:-

1. It has been pointed out[12] that traditional courses, meant to impart skill and professional identity, have been eliminated in favour of engineering science content. Balance should be restored with some modern, skill imparting courses.
2. Informatique curricula, described by Mulder[8] and Sloan[10], are sequences in depth. Applications topics are not concentrated in any single course.
3. Rony and Larsen[9] state, from their experience with non-computer science/non-electrical engineering students, that "... Within only two quarters, our students -many of whom had no prior exposure to electronics and none of whom have

substantial undergraduate or graduate course time available for studying electronics- can learn sufficient digital electronics and microcomputer interfacing to undertake projects of modest complexity. Student response to the course has been enthusiastic."

A TYPICAL PROBLEM

The following is to illustrate what might reasonably be required of an engineer in industry as regards microprocessor applications. It was desired to automate the

1. Measurement,
2. Postprocessing and
3. Tabulation

of some statistics pertinent to a periodically varying -i.e. on and off- but largely random flow of gas through a porous medium. An experiment would normally comprise 10-20 periods, during which the flow is on, separated by intervals, during which there is no flow. The periods of flow last about 0.5-3.5 sec. while the intervals of no-flow invariably exceed 8 sec. An apparatus schematic is shown in Fig. 1. Sample flow velocity and vacuum signals, A_v and A_e, are described in the graphs, Fig. 1a.

Intermittent flow is produced by controlling the applied vacuum with a servo driven flow control valve. The open loop controlling signal to the valve servo is furnished by an analogue tape recording of the aspiration vacuum, P_e', generated during tests with a human subject. The automatic slave aspirator mimics the actions of the human for two reasons:-

1. Flow velocities and volumes associated with this phenomenon are important. It is, however, not practical to include a flow meter in experiments involving human subjects because of the associated pressure drop and
2. Furthermore, various porous samples are to be tested under identical, realistic conditions. It is not possible for the subject to duplicate exactly his performance in any given test nor, for that matter, to repeat it indefatigably.

ALTERNATE SOLUTIONS

Before deciding to invoke a microprocessor based solution, alternatives were investigated:-

1. Data were reduced manually, from strip chart recordings of the signals A_e and A_p -proportional to applied vacuum and flow velocity, respectively- using scale and planimeter,
2. A simple, commercial data acquisition system was bought so as to record raw data on an ASR/digital

cassette terminal which could be used, with a time sharing service, for postprocessing and

3. A proposal and quotation for a commercial minicomputer based system was sought.

Manual data reduction was too tedious. The simple system failed to operate and no satisfactory support could be obtained from its manufacturer. The minicomputer system was quoted at a minimum of $50k. The microcomputer system was developed, including software, and commissioned for a cost of $2k.

INDUSTRIAL AND EDUCATIONAL PERSPECTIVES

This project was carried out by the author with -eventually- enthusiastic collaboration of the client's engineering staff. None of us had had any microcomputer experience. At the outset, microcomputers were regarded with suspicion; the project was undertaken with misgiving. Belief that simple, custom built microprocessors have immediate and far-reaching relevance in general engineering education sprang from this and subsequent encouraging experiences. In order to dispel doubts that undergraduate students can undertake significant application exercises of this nature, the reader is refered to the work of two students[3] in the School of Computer Science at Simon Frazer University. This constitutes an excellent example of what can be accomplished, given proper stimulation and direction.

The intermittent flow project will be discussed in the following contexts so as describe some of the activities which the novice microprocessor user might expect to assume:-

1. Operating the system; specifications
    a) The report format
    b) The measurements and their relation to logged parameters
    c) Operating sequence
2. Hardware
    a) System nucleus
    b) Analogue to digital convertor interface
    c) External memory and interface
    d) Power supplies
3. Software
    a) General features of the processor
    b) Detailed flow chart
    c) Timing
    d) Special subroutine requirements
    e) Adaptability to other applications
    f) The cross assembler
    g) Computational alternatives for microprocessors

OPERATION OF THE MICROPROCESSOR SYSTEM

Fig. 2 is a table of results from an experiment conducted with the apparatus of Fig. 3. Acquisition of data from 2-track analogue tape recordings of the signals A_e and A_v of Fig. 1 was specified. Provision was made for the operator to type in eight items for record identification.

After headings have been printed the ADC channels are sampled continuously at 100 Hz. in anticipation of a flow onset. When flow is detected, time integration of signals A_e and A_v begins. Measurements are accumulated until flow ceases. The flow event is then logged.

$$\text{ENERGY EFFORT} = K_e \sum_{t_o}^{t_n} A_e(t)$$

$$\text{VOLUME} = K_v \sum_{t_o}^{t_n} A_v(t)$$

$$\text{DURATION} = 0.01 \sum_{t_o}^{t_n} t$$

$$\text{INTERVAL} = 0.01 \sum_{t_n}^{t_o'} t$$

Where		
	$A_e(t)$	Vacuum signal at time = t
	t_o	Time of flow onset
	t_n	Time of no-flow detection
	t_o'	Time of next flow onset
	$A_v(t)$	Differential pressure signal across flow measuring capillary at time = t
	K_e, K_v	Scaling constants

Time interval, $(t_{i+1} - t_i) = 0.01$ sec., over the range of summation is constant and

VELOCITY = VOLUME/DURATION

VOLUME and ENERGY EFFORT are scaled time integrals of A_v and A_e, respectively, while VELOCITY is a volume rate. Time from flow onset until no-flow detection is logged as DURATION. After the first flow event has stopped, time is also measured from no-flow detection until the next flow onset. This is logged as INTERVAL. The test is terminated on demand when the stop button is pressed. Totals of all columns, except VELOCITY, and all column averages are then printed

The entire procedure can be repeated by typing G after the system prompt character, *, which appears when the test ends. This is called a *warm start*. The entire operating program, except for utility functions packaged by the manufacturer in read-only-memory (ROM) is in volatile read-write memory (RAM). The program is loaded from an object file on digital magnetic tape via the cassette unit of the ASR terminal. As in the case of *warm start*, the newly loaded program is initialized so that the test procedure will *cold start* when G is typed.

MINIMUM CONFIGURATION HARDWARE; GETTING STARTED

The intermittent flow processor evolved from a Motorola Design Evaluation Kit, MEK6800D1[1]. Modifications to the kit itself consisted, mostly, of removal rather than addition of components; Fig. 4.:-

1. The multicomponent clock oscillator circuit was replaced by a 1 MHz., single package clock chip,
2. The serial interface chip, MC6850 ACIA, was eliminated; terminal communications were handled by a 16-bit parallel interface chip, MC6820 PIA, using manufacturer's firmware[11], MIKBUG: MCM6830L7 ROM, and
3. The onboard MCM6810 RAM, except for *scratch* used by MIKBUG, was eliminated in favour of a more convenient 4k-byte RAM kit with full address decoding and data and address buffering.

The evaluation kit was provided with both an RS232C and a 20 ma. current loop interface at both 110 and 300 baud. Before proceeding with system development, we used the kit, together with an ASR-33 teletype and a CRT terminal, as a trainer. Thus we became familiar with the functions of MIKBUG and with hand assembled programming. This educational *bootstrapping* was necessary to *cold start* green personnel.

ANALOGUE TO DIGITAL CONVERSION

Since two analogue data were to be measured, A_e and A_v, and since the kit contained another PIA with a pair of independent input -or output, since these are dynamically reconfigurable under program control- registers, these were interfaced to a pair of 6-bit analogue to digital convertors (ADC), Fig. 5. Control functions were implemented in some of the four left-over PIA data lines. Aside from the twelve obvious data lines and the supplying of logic and analogue reference power to the ADC's, the following additional connections were required, Fig. 6.:-

1. A stop button, SB, normally grounds the line to PA7. This holds a sign bit at $0 \equiv +$ in one of the PIA registers. So long as a non-negative reading on this channel persists, the program continues. Pressing the stop button changes the sign bit by connecting PA7 to +5v. thus stopping the program.
2. The command to actuate the ADC's is produced by toggling the output of PB7 low-high-low. This signal must be put through a TTL gate since the output of the PIA is not sufficient to initiate the simultaneous conversion of both ADC's; the intervening gate supplies the necessary current.
3. The range swiches, SR1 and SR2, and the trimming resistors, PR1 and PR2, scale the measurements, A_e and A_y, respectively. This analogue implementation of the scaling constants, K_e and K_y, mentioned previously, is preferable to suffering the truncation by division of small integer readings or the overflow risked through the multiplication of large ones.

ADDITIONAL MEMORY

A 4k-byte (4096x8) semiconductor RAM memory kit, Fig. 7, composed of 32 (1024x1)-bit 2102 chips, was built to provide sufficient memory for the flow processor. This memory is *plug-compatible* with a number of popular micro-computers but required modification for the MEK6800. This includes:-

1. Tying the eight separate data input and output lines, DIØ-DI7 and DOØ-DO7, respectively, together so as to connect to the eight line M6800 data bus,
2. Inverting the READ/WRITE signal input, R/W, so as to comply with the polarity of this signal on the M6800 control bus and
3. *ANDing* together the VALID MEMORY ADDRESS (VMA), *Phase II clock (data transfer cycle in progress)* $\phi 2$ and the R/W signals:-

   $$\text{SMEMR} = (\text{VMA}) \cdot (\phi 2) \cdot (\text{R/W})$$

   to enable the SET MEMORY READY OUT, SMEMR, signal during a read operation.

Modifications, implemented with spare gate capacity, entailed only four small wiring additions. No new components were required.

POWER SUPPLIES

Although the M6800 requires only +5v. logic power, this application required other voltages as well:-

1. ±12v. to supply the terminal interfaces; RS232C and 20 ma. current loop,
2. ±15v. for analogue reference voltage in the ADC's

3. And a crude +8v. supply to augment logic power for external RAM memory. The RAM kit was equipped with voltage regulators and the original +5v. regulated power supply had capacity only for the evaluation kit and the ADC's.

Clearly, a multi-voltage, single power supply would have been neater. Unit proliferation is typical of prototype systems.

SYSTEM INTEGRATION

The units of Fig. 8 were mounted on stand-offs on a 70x50 cm. base and added in the sequence shown. These were interconnected by a cable harness to which odd plugs and connectors were added as required. A panelled card cage with a properly designed back plane is more attractive but a base mounted assembly provides better access to all components; a useful feature for trouble shooting, of which a considerable amount is required during the learning process.

The two types of terminal interface were brought to a solid mounting, Fig. 9, and terminated in standard plugs. Two switches provide 10 or 30 character/sec. transmission selection.

OPERATING SOFTWARE

The major procedural blocks are illustrated in Fig 10. The procedure is fully described in Fig. 11. Relevant parameters are defined as follows:-

Parameter	Description
PIP = 0	No-flow condition exists; flag
= 1	Flow condition exists
PUFFNØ	Event number; column 1 on Fig. 2
SPFDUR	Sum of all event durations; total in column 2
SPFINT	Sum of all inter-event intervals; total in column 3
SUMV	Sum of all volume measurement values; total in column 4; integral of VACC
SUMVEL	Sum of all volume rate calculations
SUMP	Sum of all vacuum measurement values; total in column 5; integral of PACC
PFINT	Inter-event interval; column 3
PFDUR	Event duration; column 2
VPUF	Event volume; column 4
PPUF	Event vacuum; column 6
VACC	Volume measurement; threshold accumulation
PACC	Vacuum measurement; threshold accumulation

Parameter	Description (cont'd.)
measure	A function which returns measurements, PA and PB, from the ADC's
PA	Vacuum transducer measurement in PIA-A-register; signal A_e
PB	Velocity transducer measurements in PIA-B-register; signal A_v
FILCNT	Accumulation counter
FILMAX	Limit of FILCNT
PTHRSH	If PACC exceeds PTHRSH, flow is in progress
update	A function which updates PUFDUR, VPUF and PPUF
VPFAC	Volume units scaling factor; K_v
PPFAC	Vacuum units scaling factor; K_e
SMVFAC = VPFAC	
SMPFAC = PPFAC	
APFDUR	Average event duration
APFINT	Average inter-event interval
AUMV	Average volume aspirated per event
AUMVEL	Average volume rate of aspiration
AUMP	Average integral of aspiration vacuum per event
TOTALT, AVRGT	TOTAL and AVERAGE labels of last rows
PFVEL	Event volume rate; column 5
∅PTIME	Line output time of terminal; used to initialize an inter-event interval after processor prints an event log

The triangle symbols, WAITn, are programmed idle loops. These synchronize program execution by making all procedural paths equal. Delays were tailored by adding up individual instruction times. An exception occurs during printing. Although instruction count is lost, timing is maintained by initializing the subsequent inter-event interval; PUFINT = ∅PTIME, the time taken to print. The idle, WAIT4, compensates for the non-existent inter-event interval preceding the first event.

SUBROUTINES

Two of the processor subroutines bear brief mention. These are examples of software, which, though general enough to be commercially available, the user may nevertheless choose to write himself.

The analogue measurement routine:-

1. Writes three bytes in sequence to a PIA to which the ADC's are connected. The three write operations toggle the encode signal, PB7, so as to command measurements to be made,
2. Waits until the maximum conversion time of the ADC's has elapsed and

3. Reads the two PIA registers, which now hold the 6-bit readings, right justified as binary integers.

Note that this routine must also configure PB7 as output prior to toggling the encode signal, then reconfigure both registers, PA and PB, as input prior to reading the ADC's.

Another subroutine, the decimal formatter, facilitates the printing of tabulated numerical data. It prints a 16-bit 2's complement integer with minus sign, leading and trailing 0's and decimal point inserted as required. Two bytes of format information are used to print each number:-

Byte 1:- If negative, a decimal point will appear
- If positive, none is required
- Magnitude is the field width into which the number will be right justified

Byte 2:- If negative, decimal point is moved left; leading, most significant 0's inserted, as required
- If positive, decimal point is moved right; trailing, least significant 0's inserted, as required
- Magnitude is the number of places that the decimal point is to be moved; left or right
- Trailing, but no leading 0's may be inserted if Byte 1 is positive

ADAPTABILITY TO OTHER APPLICATIONS

The following is included to show how the hardware, and to some extent the software, developed for the intermittent flow processor can serve in other applications.

Response characteristics of oscillatory systems are commonly determined by experiment, Fig. 12. The system to be analysed is excited at a number of discrete, simple harmonic frequencies over the desired test range. A Bode Diagram is constructed, wherein amplitude ratio, A, and phase shift, ϕ, of the output signal with respect to the excitation signal are plotted -vs- excitation frequency, ω. Response properties of the test system may be discerned by identifying the break point frequencies via the amplitude ratio characteristic. Although tedious, such experiments are often performed manually. It was therefore decided to investigate the feasibility of a microprocessor based analyser. Specifications were tentatively defined:-

1. Pure harmonic excitation,
2. Excitation frequency range: 1 Hz. to 1000 Hz.,
3. Frequency increments: $\sim$10 Hz.,
4. System output signal sample of 128 measurements,
5. Use of Fast Fourier Transformation (FFT) to compute amplitude and phase of fundamental response harmonic and
6. Tabulation of Bode Plot coordinates; an interfaced plotter would be preferable, its accompanying complexity, for the present, would not.

Hardware for this analyser is summarized in Fig. 12. Two existing ADC's are sufficient. These, capable of 20 kHz. simultaneous sampling rate, resolve 20 points/wavelength of a 1000 Hz. signal. The primitive line toggling ADC triggering procedure is too slow however. A new measurement subroutine, which exploits the PIA input and output strobe signals, CA1, CA2, CB1, CB2, must be written.

Excitation signal generation hardware, which includes:-

1. An additional PIA to communicate with a
2. Digital to analogue convertor (DAC) which selects the output frequency of a
3. Voltage controlled oscillator (VCO),

must be added. Tests on vibrating mechanical or structural systems would also require:-

4. Power amplification of excitation signal,
5. An applied force generator or shaker and
6. An accelerometer.

But the analyser could probably test simple passive, electrical circuits with only the first three additions. The PIA, DAC and VCO are all single package integrated circuits and require only a few discrete components to complete their interface to the microprocessor.

A version of the well-known Cooley-Tukey FFT routine, written in FORTRAN II, Fig. 13, was used as a model for an integer version written for the M6800. By using suitable 1, 2 and 3-byte arithmetic and by developing an integer Sine/Cosine subroutine, the FFT was implemented in less than 1500 bytes of memory, including 512 bytes for the 128 data samples and their subsequent transformations; i.e. two bytes for both real and imaginary components. A transformation is performed in about 1.5 seconds.

The FFT will not be elaborated further. However, the Sine/Cosine routine and the Arctangent and Square Root functions, performed by pseudo division, will be discussed in some detail. Consider that the two latter functions are required to compute amplitude and phase from the transformed data:-

$$A = (I^2 + R^2)^{\frac{1}{2}} \quad \text{and}$$

$$\phi = tan^{-1}(I/R) \quad \text{where}$$

I and R are the imaginary and real parts, respectively.

SCIENTIFIC FUNCTIONS FOR MICROCOMPUTERS

Some complicated calculations and scaling of data are inevitable in all but the simplest applications. There are a number of ways to perform calculations. Those deemed most appropriate for small systems will be emphasized. Algorithms, developed for hand calculators, can be efficiently used with microcomputers.

HARDWARE FLOATING POINT PROCESSOR

Floating point processors, implemented in extensive parallel, high speed logic and with scientific function capability, are fast and effective but very expensive. They can usually be justified only at the top end of the minicomputer range. It is unlikely that a small microcomputer system would require such an accessory.

INTERPRETER

BASIC interpreters for microprocessors come in varying degrees of capacity. 32-bit floating point arithmetic and scientific subroutines are available. This software is aimed at a rapidly expanding personal computer market and is suitable to many business applications. Lest it be concluded, however, that a powerful BASIC interpreter is indispensible, certain undesireable byproducts should be noted:-

1. Substantial increase in memory, hence power, required,
2. The interpreter may not communicate conveniently with the utility routines in ROM, e.g. MIKBUG,
3. The memory size increase may prohibit *skeleton*, or partially decoded, addressing; a particularly attractive feature which reduces both the wiring and loading of address lines in minimum configuration systems,
4. A procedure, whose timing is based on instruction cycle accounting, cannot execute interpretively; imposition of a real time clock and interrupt routines have to be accepted,
5. Interpretive decisions, i.e. IF, THEN, ELSE, may be unacceptably slow and
6. User written routines would suffer data intercommunication constraints.

The foregoing criticism was voiced with the confidence that the many excellent features of high level interpreters are common knowledge.

CALCULATOR CHIP INTERFACE

A scientific calculator chip can enhance a microprocessor[4] where high precision and low speed are a desireable compromise. Such a peripheral was built and tested and has operated successfully as a calculator simulator, Fig. 14. Considerable software is required, however, to provide general programmed access; this has not yet been attempted. The attractive features of this device include:-

1. Calculator chips, even with 12 digit mantissae, are cheap; some cost as little as $5.,
2. The interface is not expensive and is easy to build,

3. Operation is conceptually simple; a PIA simulates keystrokes for input and
4. Accepts the serial bit stream which would normally drive the display; decoding is straightforward.

The microprocessor and the interfaced calculator chip are not entirely compatible partners. When keystroke input encode and display signal output decode involve conversion to and from binary numbers, the programming task becomes significantly more formidable. On the hardware side, the need for another power supply, 7.5v. this time, arises.

There might be considerable application, however, for a calculator chip designed to operate in conjunction with a microprocessor, either as an addressed peripheral on the data bus or at least more compatible with a PIA. This adaptation might accept and present data in a number of successive bytes. User specified input and output format, in various floating point and integer configurations, might be permitted. Consider a hypothetical 14-pin bug with eight bi-directional data lines and four control lines; Fig. 15. Possibly a chip select (CS) and a clock input (ϕ2) might be added to the pin count if it were desireable to put D0-D7 directly on the data bus. The proposed 14-pin layout may well suffice, interfaced to a PIA. An operating sequence to multiply two numbers, as illustrated in Fig. 15, might be as follows:-

1. Configure D0-D7 as output from PIAA
2. Set M = 3; i.e. M0 = 1 and M1 = 1
3. Place *CLEAR function code* on D0-D7 (assume that an appropriate set of function and operation codes, not tabulated in Fig. 15, have been defined)
4. Toggle ACPT MOD (TAM)
5. Place *1-BYTE: SCALE B8 function code* on D0-D7
6. TAM: Set M = 0
7. Place *A* on D0-D7
8. TAM: Set M = 3
9. Place *2-BYTE: SCALE B0 function code* on D0-D7
10. TAM: Set M = 0
11. Place *Least significant half of B* on D0-D7
12. TAM: Place *Most significant half of B* on D0-D7
13. TAM: Set M = 1
14. Place *MULTIPLY function code* on D0-D7
15. TAM: Set M = 3
16. Place *3-BYTE: Floating Point code* on D0-D7
17. TAM: Set M = 2
18. Configure D0-D7 as input to PIAA
19. TAM: *C+0 = sign/exponent result*
20. TAM: *C+1 = most significant half result*
21. TAM: *C+2 = least significant half result*

This procedure is cumbersome. However very inhomogeneous data format examples were selected for operands and result. All I/O was handled by PAn and PBn lines. If RDY was connected to the interrupt flag lines, CA1 or CB1, and the toggling of ACPT MOD was done automatically, by CA2 or CB2, whenever the PIA was loaded or emptied, the procedure would be greatly simplified.

ALGORITHMS

Calculations, including scientific routines, can be implemented in software. This is often acquired n'th hand and has usually reaped some benefits of natural selection. Software can also be tailored by the user to serve specific application needs. This often compensates somewhat for the programming effort and extra memory requirement.

Algorithms may be classified as table lookup (TLU), recursive or a mixture of both. The Sine/Cosine routine described later is an example of a table lookup. The Division and Square Root part of the pseudo division examples are essentially recursive. The pseudo division Log_e and Arctangent examples use both lookup and recursion. Generally TLU's are fast and present the user with a size-vs-precision trade-off while iterative procedures constitute a speed-vs-precision compromise.

MICROPROCESSOR SINE AND COSINE SUBROUTINE

A Sine/Cosine subroutine was written for an 8-bit microcomputer in order to implement a Fast Fourier Transformation algorithm. Raw data to be transformed is acquired via 6-bit analogue to digital convertors. This amplitude resolution, of better than 2% of full scale, was deemed sufficient for parameter identification in many electromechanical systems. Considering the relatively low precision of the raw data, a TLU, wherein angle and function are scaled over one quadrant at a precision similar to that of the full scale magnitude of the measured signal amplitude, seems to present a good design compromise among simplicity, size and execution speed.

The table contains a sequence of sixty five 8-bit integers which represent Sine values at sixty four equal angular intervals, spanning the range 0 to $\pi/2$, i.e. the first quadrant. This sequence also represents Cosine values in the fourth quadrant. Therefore Cosines are handled by adding a count of $64 = \pi/2$ to angle arguments and treating them as Sines. Multiple rotations are handled by successive subtractions of 256. Using absolute arguments for lookup, the function values are simply post-negated if the arguments were negative. The quadrant into which the angle falls is determined by successive subtractions of 64 from the angle. The function is negated if two or three subtractions are required to establish a first quadrant residue; i.e. if the angle is in the third or fourth quadrant. If an odd number of subtractions was required, i.e. 1 or 3 for angles in the second or fourth quadrants, the angle is complemented, before lookup, by subtracting the first quadrant residue from 64. This Sine/Cosine algorithm is illustrated in the flow chart, Fig. 16.

The routine operates on a 16-bit, 2's complement representation of the angle. A full revolution corresponds to an integer count of 256. The function value is returned as an 8-bit, 2's complement integer. The following illustrates an example of the evaluation of *cos*(770°):-

ANGLE		
degrees	integer units	
+ -770	+ -548	
90	64	add $\pi/2$
-680	-484	take absolute value of angle and
680	484	set first negative flag and
- 360	- 256	subtract 2π
320	228	
- 270	- 192	subtract $3(\pi/2)$
50	36	since $3 > 1$, set second negative flag; since 3 is odd, complement angle
90	64	
- 50	- 36	
40	28	
0.643	41	evaluate *sin*: $41/64 = 0.641$
-0.643	- 41	negate for first flag
0.643	$\underline{\underline{41}}$	negate for second flag; returned value

MODIFIED DIVISION (PSEUDO DIVISION)

A division algorithm, which is adaptable to calculation with operands of any precision, can be modified so as to compute:-

1. Y/X
2. $(Y/X)^{\frac{1}{2}}$
3. $log_e(1+Y/X)$
4. $tan^{-1}(Y/X)$

This method was developed, in decimal arithmetic, by Meggitt[5] who also extended it, using a modified multiplication algorithm (pseudo multiplication), to calculate $X(e^p-1)$, *sin*(p), *cos*(p), *tan*(p) and Xq^2. This sort of procedure, implemented in ROM, is used in electronic calculators. Milgram's[6] binary pseudo division was used in a short, fast subroutine for microcomputers using 24-bit, 3-byte operands. This subroutine is described in the flow chart, Fig. 17. The nomenclature is as follows:-

Y	numerator (input)
X	denominator (input)
j	iteration count (places after integer point)

A_j	remainder (at step j)
B_j	divisor
Q_j	quotient
q_j	binary coefficient contributed to quotient (i.e. $Q_j = \sum_i q_i 2^{-i}$)
n	last iteration ($j_{max}=n$; Q_n contains n bits)

Obviously

$$A_o = Y$$
$$B_o = X$$
$$Q_o = 0$$

and

$$q_j = A_j/B_j$$

and

$$A_{j+1} = 2(A_j - q_j B_j)$$

Although

$$B_j = X$$

throughout, for division, B_j must be modified at every step to obtain values of the other functions. For (a) *log*$_e$ and (b) *tan*$^{-1}$, the bits, q_i, of the quotient serve as pointers to a table of:-

(a) *log*$_e(1+2^{-j})$,
(b) *tan*$^{-1}(2^{-j})$,

respectively. Viz.:-

j	$1+2^{-j}$	(a) *log*$_e(1+2^{-j})$	2^{-j}	(b) *tan*$^{-1}(2^{-j})$
0	2.	0.63914	1.	0.785398
1	1.5	0.405465	0.5	0.463648
2	1.25	0.223144	0.25	0.244979
3	1.125	0.117783	0.125	0.124355
4	1.0625	0.0606247	0.0625	0.0624188
5	1.03125	0.0307717	0.03125	0.0312398
6	1.015625	0.0155043	0.015625	0.0156237
7	1.0078125	0.0077821	0.0078125	0.00781227
8	1.00390625	0.0038987	0.00390625	0.0039062

E.g., to eight binary places for (a):-

```
        Y = 3        .405465
        X = 4        .117783
                     .030772
                     .003899
                     -------
        Q8 = 169     .557919
     = 10101001₂     .003899 (+ least significant bit of Qn)
                     -------
                     .561818
```

We see that

$0.557919 < (log_e(1.75) = 0.559616$; from tables) < 0.561818

Similarly, an example for (b):-

```
        Y = 1        .244979
        X = 3        .026419
                     .007812
                     .003906
                     -------
        Q8 = 83      .319116
     = 01010011₂     .003906 (+ least significant bit of Qn)
                     -------
                     .323022
```

We see that

$0.319116 < (tan^{-1}(1/3) = 0.321751$; from tables) < 0.323022

Notice that these computations are precise, in the integer sense of truncation, since $Q_n \leqq$ *true value* but if a constant, equivalent to that contributed by the least significant bit, is added to Q_n, the result exceeds *true value*.

A note of qualification, concerning the Square Root, Log_e and Arctangent computations should be added. Whereas the division of Y/X produces an error free result regardless of the justification of the operands, i.e.:-

$Y = (000 \ldots\ldots 001)_2$ or $(010 \ldots\ldots 000)_2$ and

$X = (000 \ldots\ldots 011)_2$ or $(110 \ldots\ldots 000)_2$

both yield

$Q_n = (010 \ldots\ldots 101)_2$.

This is not so with the three other functions. In these, Y and X must be left justified so as to achieve maximum precision in Q_n. Also, since, in their course, both A_i and B_i have occasion to increase during their iteration, the user must take precaution so as to control carry-out from the most significant bytes of A and B, respectively.

SOFTWARE DEVELOPMENT TOOLS

Hand assembly of programs, up to about 100 instructions in length, is feasible. This is quickly done by writing down the symbolic instruction and address mnemonics and translating these, with the aid of an instruction reference card, into hexadecimal machine format. If there are no user discernable microcoded instruction fields of 1, 2, 3, 5, 6 or 7-bits, hand assembly is particularly easy since, in this case, every symbolic instruction will correspond to one or more hexadecimal digits. Not all microcomputer architectures embody this convenience.

A microcomputer cross-assembler which runs on a fairly large host computer and produces:-

1. A loadable object tape,
2. An assembly listing on a high speed printer and
3. Which accepts card input

is probably the single most useful programming aid. Cross-assemblers are often written in portable FORTRAN IV and may be adapted to host machines of various word length. Large programs are invariably composed of independent modules which have to be verified and integrated in stages. Therefore card input is particularly desireable because a deck, with the aid of a listing, can be conveniently and extensively edited.

CONCLUSION

Although some scepticism has been expressed concerning the feasibility of in-house system building by the user[7], serious efforts are afoot to define standards for high level microcomputer software and hardware[2]. These efforts, together with the availability and low price of microprocessors and other digital, linear and hybrid *LSI* components, raise considerable optimism that it is possible to effectively treat the design and application of systems, composed of these components, in the general engineering curriculum.

INTERMITTENT FLOW THROUGH A POROUS MEDIUM

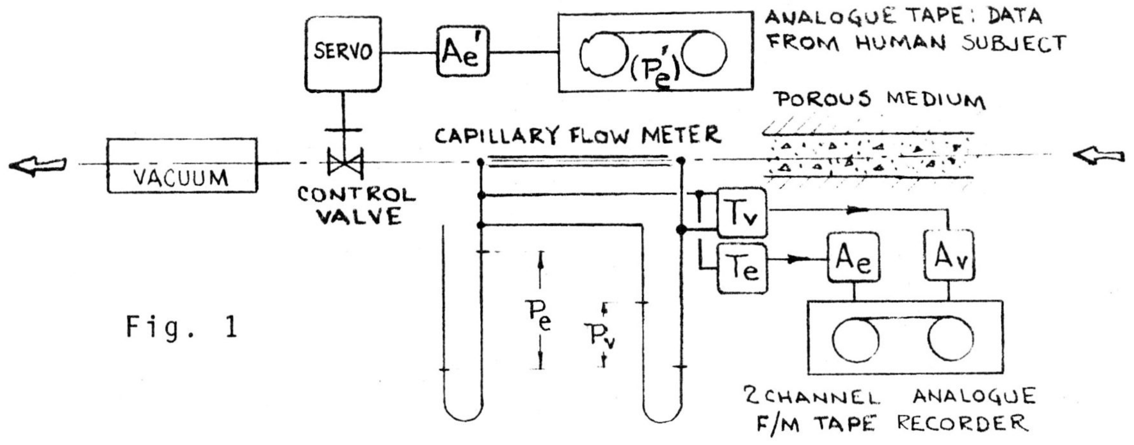


Fig. 1

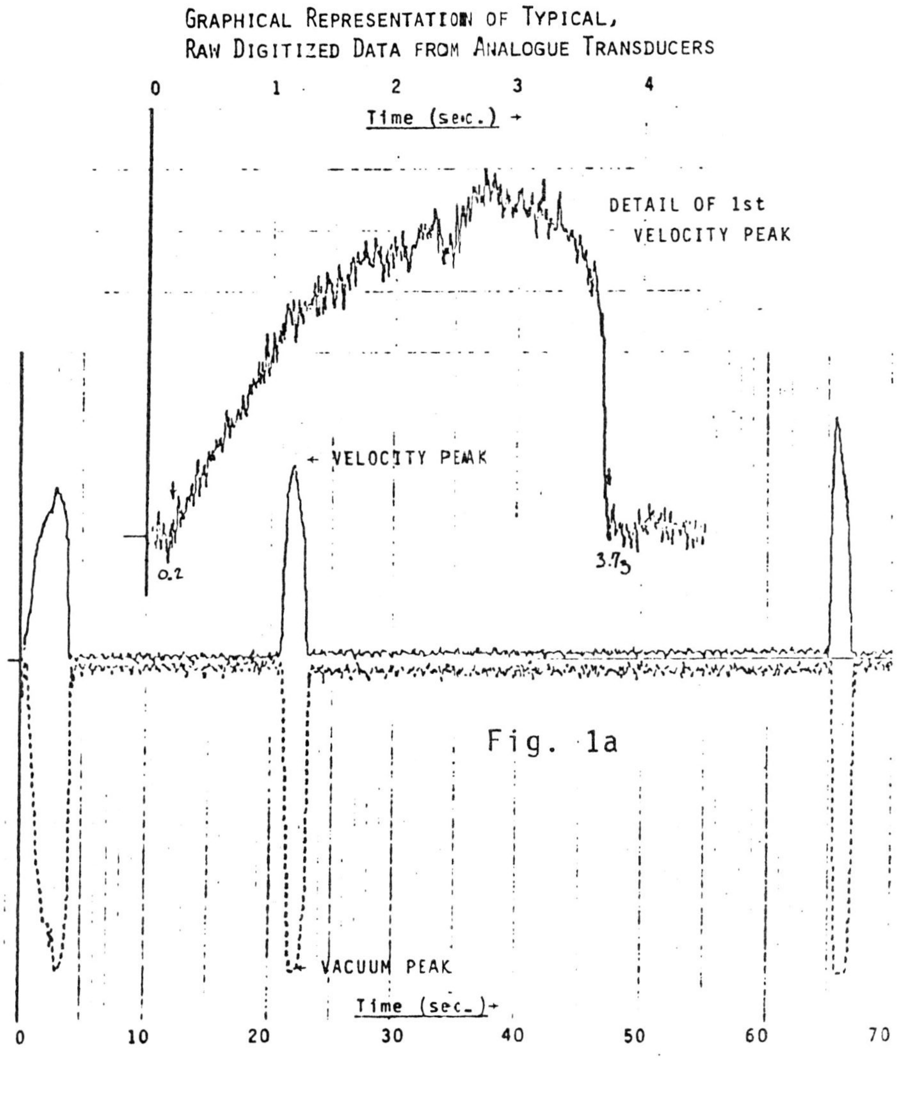


Fig. 1a

FORMATTED REPORT OF AN EXPERIMENTAL RUN
PRODUCED BY THE MICROPROCESSOR SYSTEM

Fig. 2

```
*G


ANALOGUE TAPE NO.      000000000          DATE           10/12/76/
SUBJECT NO.            000000000          TIME           18:37:45:
START @                ..... 189 FEET     END @                250 FEET
TEST SPECIMEN          XXXXXXXXX          TEST LENGTH    YYYYYYYYY MM.

EVENT NO.  DURATION    INTERVAL   VOLUME    VELOCITY       ENERGY EFFORT
            (SEC.)      (SEC.)    (ML.)   (ML./SEC.)   (CM. WATER*SEC.)


    1         1.6                 50.6       31.6              49.1
                         13.3
    2         1.0                 37.0       37.0              34.0
                         23.8
    3         0.9                 17.1       19.0              15.9
                         14.5
    4         1.7                 57.9       34.0              51.8
                         20.0
    5         1.6                 32.8       20.5              32.1
                         18.7
    6         2.0                 50.2       25.1              62.6
                         20.3
    7         2.8                 61.5       21.9              74.8
                         23.7
    8         1.8                 47.2       26.2              54.9
                         18.6
    9         1.1                 46.8       42.5              42.0
                         35.8
   10         1.9                 38.5       20.2              34.8
                         12.0
   11         2.7                 60.0       22.2              59.2
                         17.8
   12         3.9                 83.8       21.4              72.9
TOTAL        23.0       218.5    583.4                        584.1
AVERAGE       1.9        19.8     48.6       26.8              48.6



*
```

TEST SYSTEM CONFIGURATION

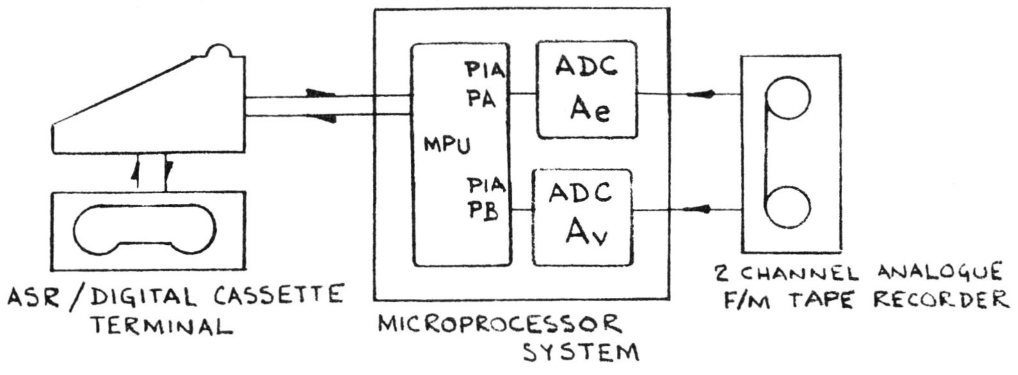
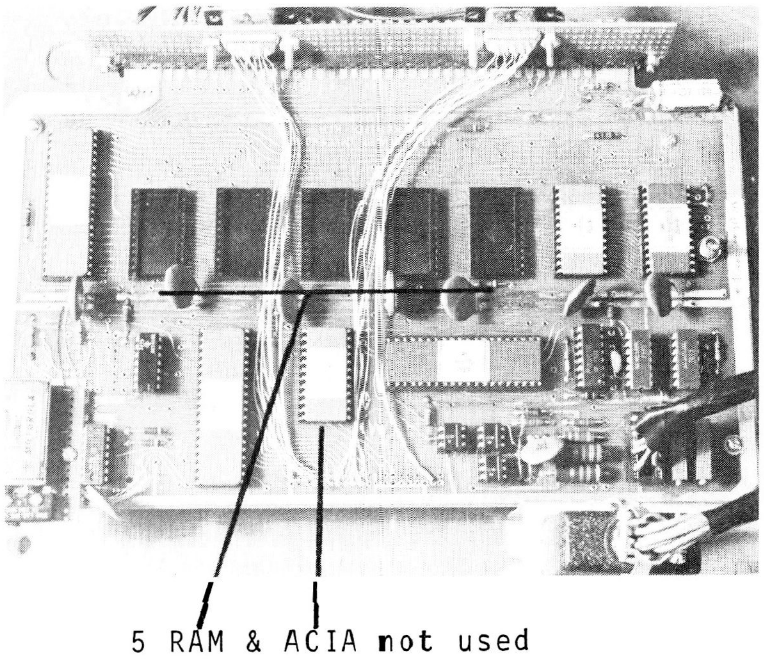
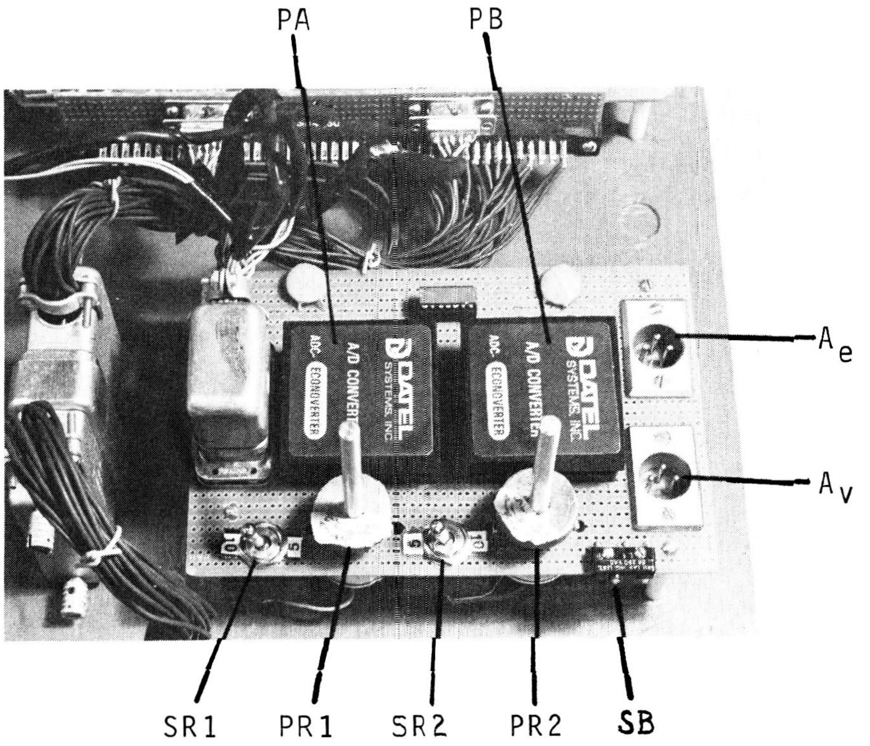

PIA
PA
ADC
Ae
MPU
PIA
PB
ADC
Av
ASR / DIGITAL CASSETTE TERMINAL
MICROPROCESSOR SYSTEM
2 CHANNEL ANALOGUE F/M TAPE RECORDER

Fig. 3

EVALUATION KIT Fig. 4





ANALOGUE TO DIGITAL CONVERTORS Fig. 5

ANALOGUE TO DIGITAL CONVERTOR INTERFACE

INPUT/OUTPUT CONNECTIONS

PIN	FUNCTION
1	E.O.C. (STATUS)
2	OFFSET CONTROL
3	START CONVERT
4	INTERNAL CLOCK OUT (SERIAL OUTPUT)
5	BIT 1 OUT (MSB)
6	BIT 2 OUT
7	BIT 3 OUT
8	BIT 4 OUT
9	BIT 5 OUT
10	BIT 6 OUT (LSB)
17	+5V POWER INPUT
18	+15V POWER INPUT
19	−15V POWER INPUT
20	POWER GROUND
21	F.S. ADJUST
22	ANALOG GROUND
23	ANALOG INPUT 2
24	ANALOG INPUT 1

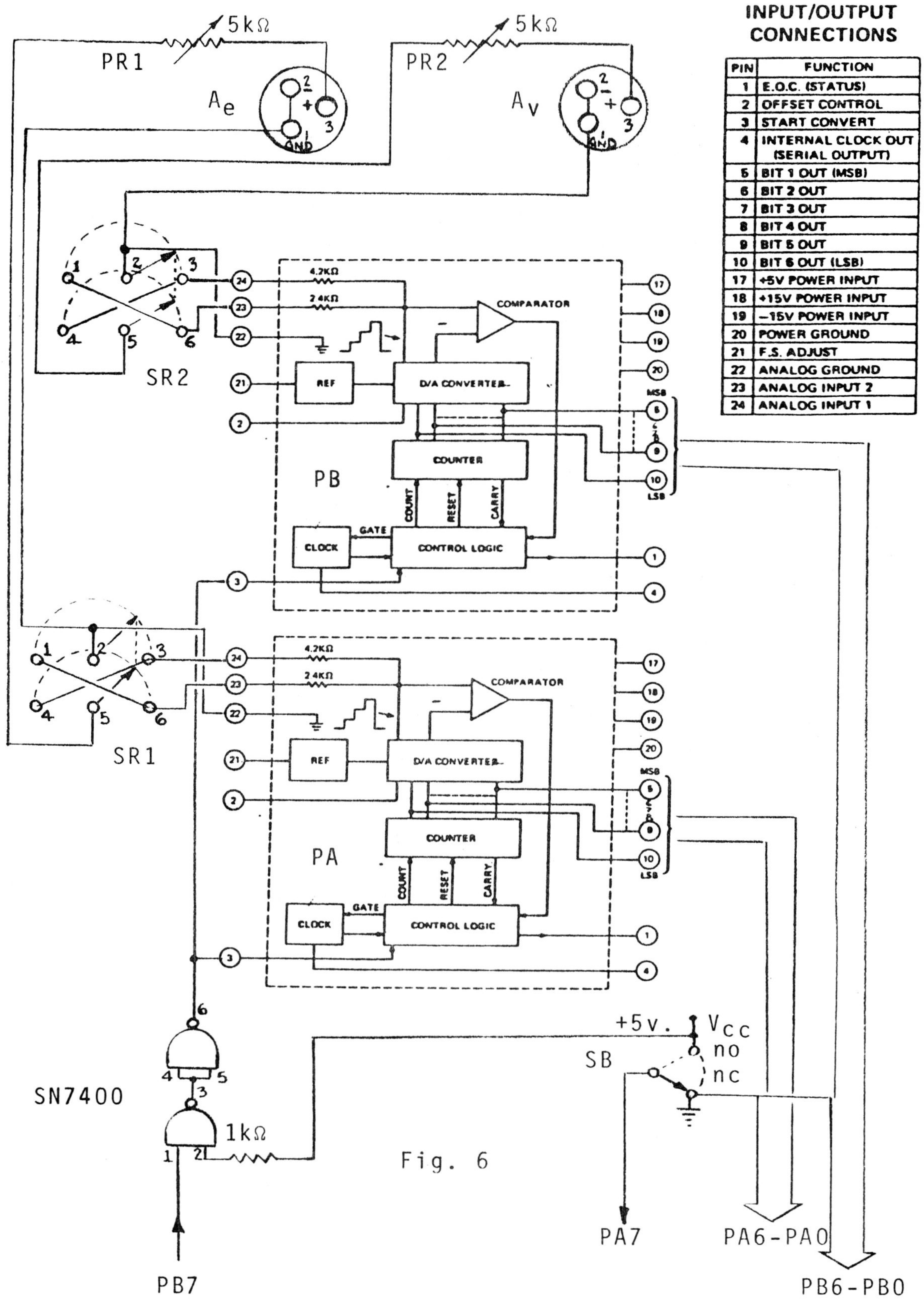


Fig. 6

MEMORY MODIFICATIONS; RAM KIT

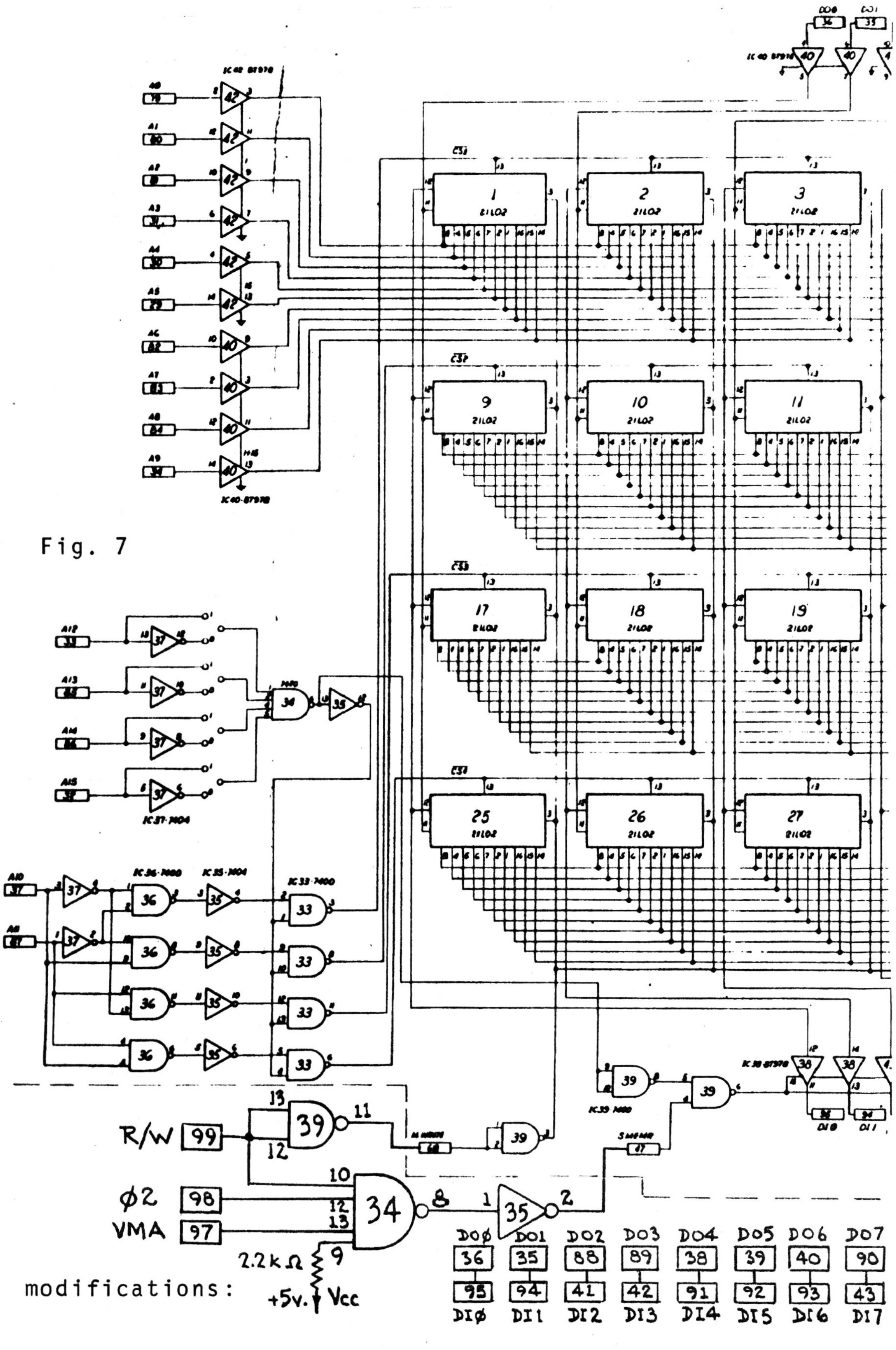
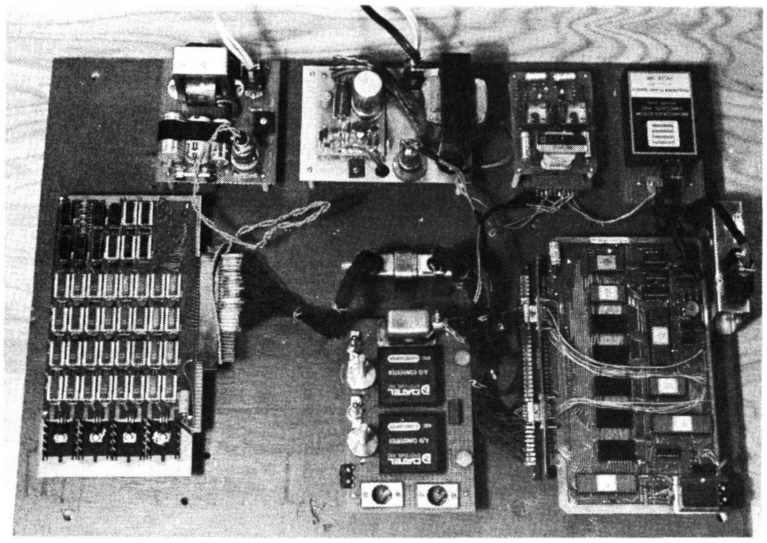


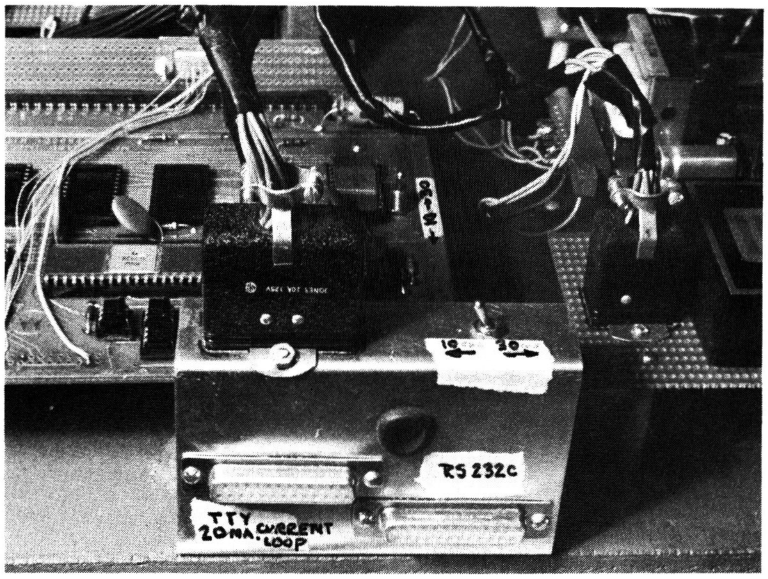
Fig. 7

modifications:

INTERMITTENT FLOW PROCESSOR Fig. 8



TERMINAL INTERFACE Fig. 9



INTERMITTENT FLOW PROCESSOR (GENERAL)

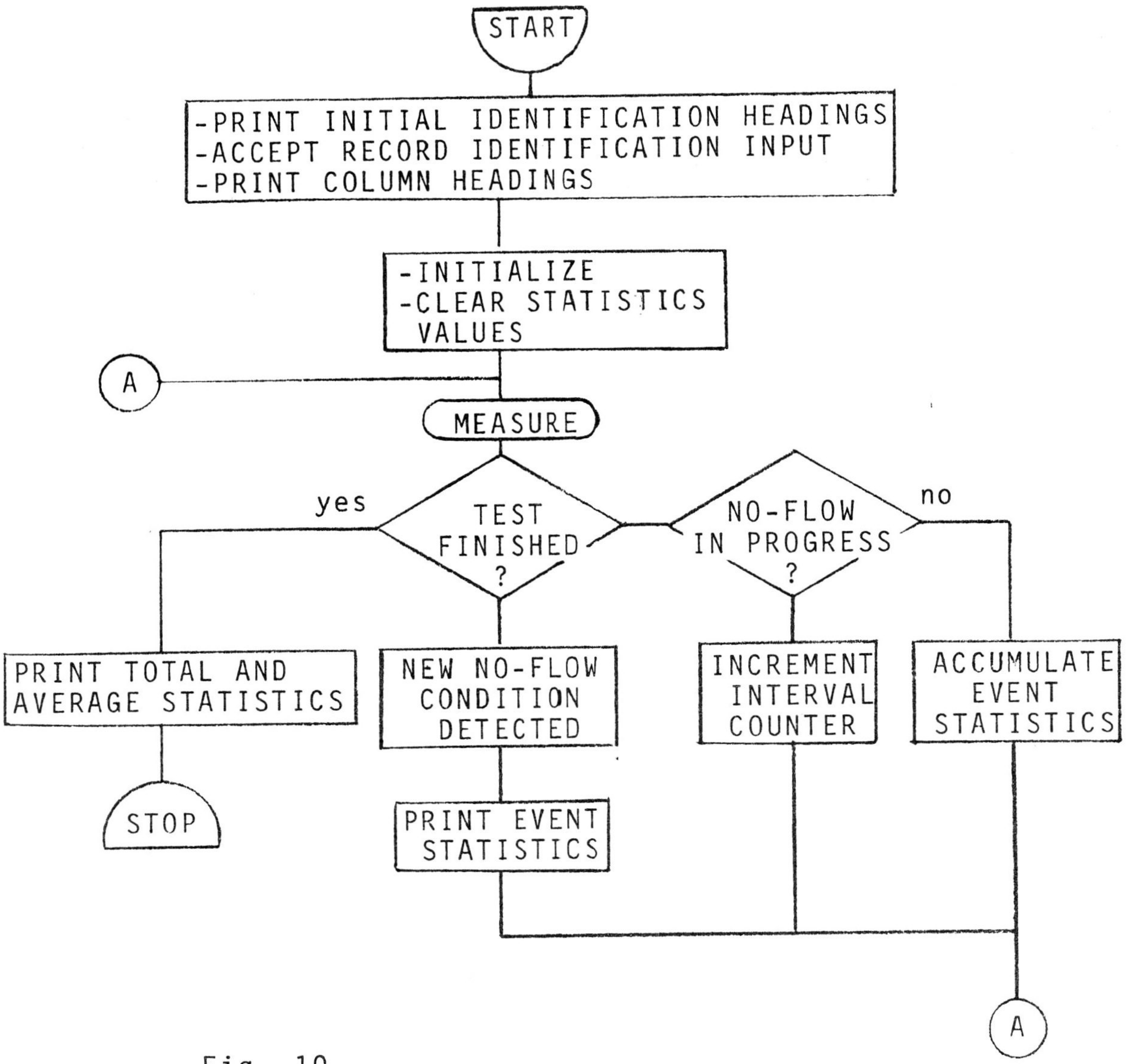

START
-PRINT INITIAL IDENTIFICATION HEADINGS
-ACCEPT RECORD IDENTIFICATION INPUT
-PRINT COLUMN HEADINGS
-INITIALIZE
-CLEAR STATISTICS VALUES
A
MEASURE
yes
TEST FINISHED ?
NO-FLOW IN PROGRESS ?
no
PRINT TOTAL AND AVERAGE STATISTICS
NEW NO-FLOW CONDITION DETECTED
INCREMENT INTERVAL COUNTER
ACCUMULATE EVENT STATISTICS
STOP
PRINT EVENT STATISTICS
A

Fig. 10

INTERMITTENT FLOW PROCESSOR (DETAIL)

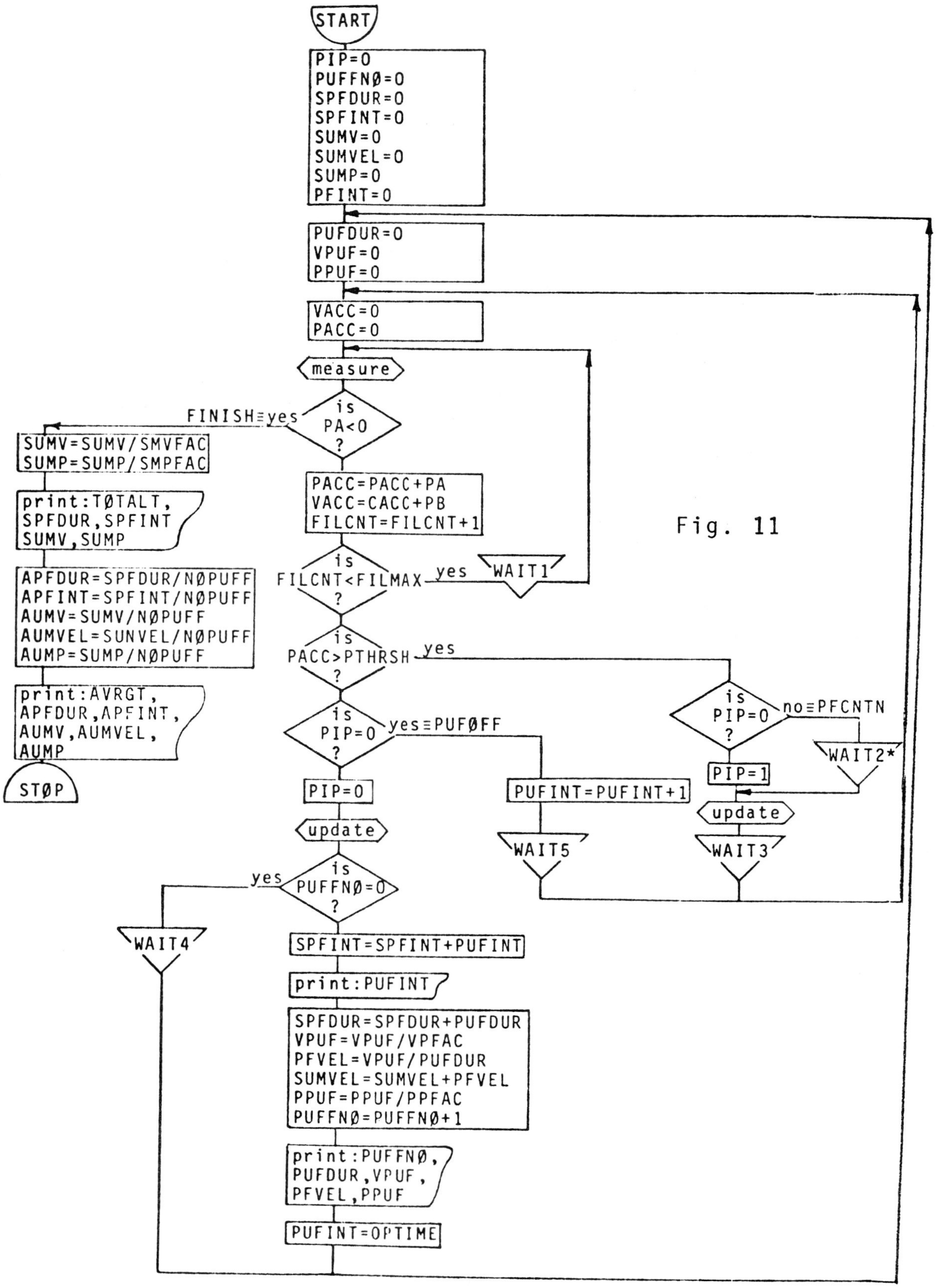


Fig. 11

BODE ANALYSER

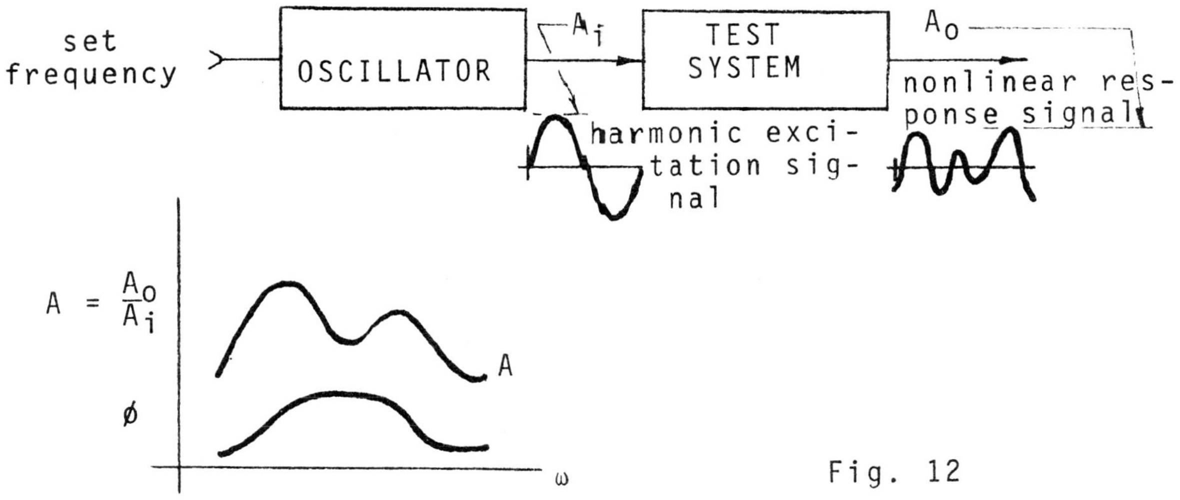
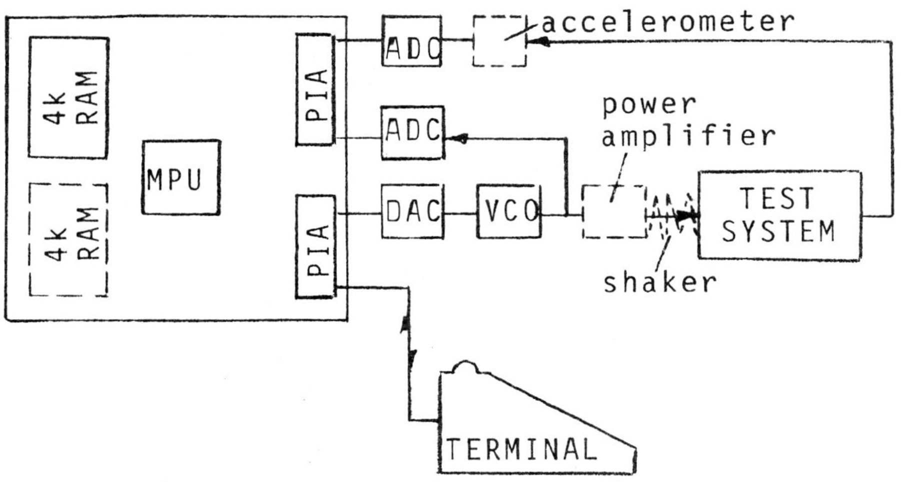


Fig. 12



FAST FOURIER TRANSFORM ROUTINE

```
      SUBROUTINE FFT(NN,DATA,ISIGN)
      DIMENSION DATA(N)
      PI2 = 0.5*3.14159
      N = NN*2
      J = 1
      DO 50 I = 1,N,2
            IF (I - J) 10,20,20
 10         TEMR = DATA(J)
            TEMI = DATA(J + 1)
            DATA(J) = DATA(I)
            DATA(J + 1) = DATA(I + 1)
            DATA(I) = TEMR
            DATA(I + 1) = TEMI
 20         M = N/2
 30         IF (J - M) 50,50,40
 40         J = J - M
            M = M/2
            IF (M - 2) 50,30,30
 50         J = J + M
      MMAX = 2
 60   IF (MMAX - N) 70,90,90
 70   ISTE = 2*MMAX
      DO 80 M = 1,MMAX,2
            THET = ISIGN*(M - 1)*3.14159/MMAX
            WR = SIN(THET + PI2)
            WI = SIN(THET)
            DO 80 I = M,N,ISTE
                  J = I + MMAX
                  TEMR = WR*DATA(J) - WI*DATA(J + 1)
                  TEMI = WR*DATA(J + 1) + WI*DATA(J)
                  DATA(J) = DATA(I) - TEMR
                  DATA(J + 1) = DATA(I + 1) - TEMI
                  DATA(I) = DATA(I) + TEMR
 80               DATA(I + 1) = DATA(I + 1) + TEMI
      MMAX = ISTE
      GO TO 60
 90   RETURN
      END
```

Fig. 13

INTERFACED CALCULATOR CHIP

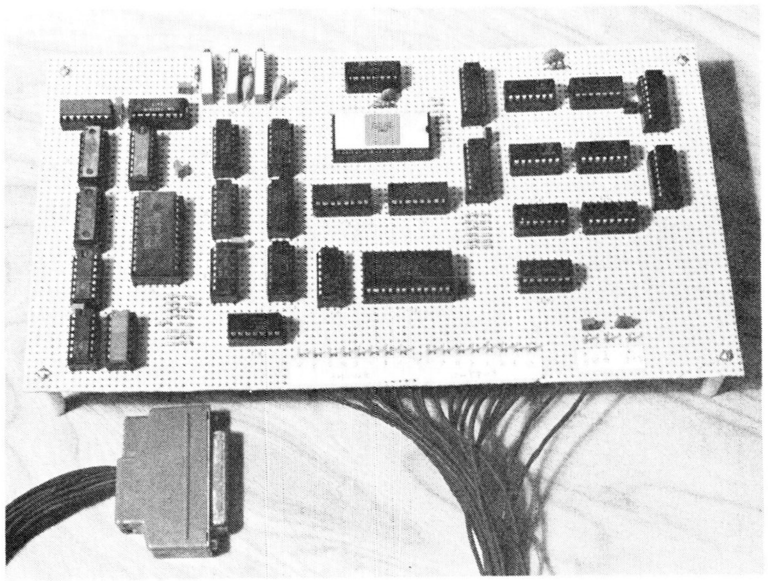
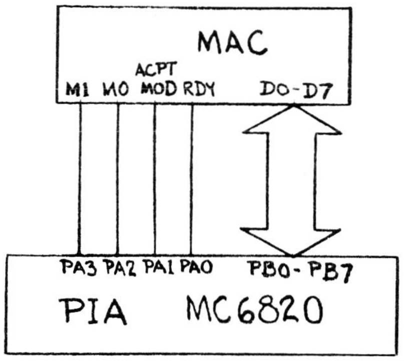
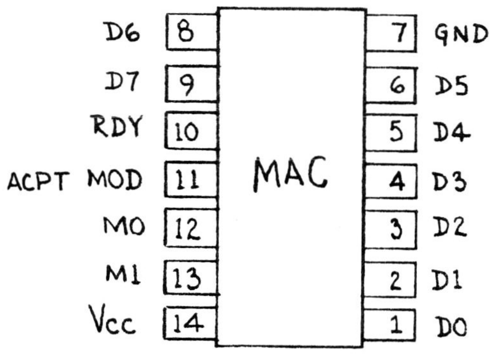


Fig. 14

MICROCOMPUTER ADAPTED CALCULATOR CHIP (MAC)





D_0 - D_7	8-bit data or command byte; tristate lines accept or produce: (a) Sequence of operand input bytes, (b) Format/precision selection or other control function byte, (c) Operator byte or (d) Sequence of result output bytes.
RDY	A (_|‾) transition signals that a sequence of operations, defined by mode select (M_0,M_1), at time of most recent ACPT MOD strobe (_|‾), is complete.
ACPT MOD	Read (M_0,M_1); then read (D_0-D_7) or write (D_0-D_7) when strobe (_|‾) occurs.

M_0	M_1	
0	0	Accept operand, 1 byte at a time/strobe, as per currently prevailing status as defined by most recent function byte input.
0	1	Accept operator byte
1	0	Transmit result, 1 byte at a time/strobe, as per current status.
1	1	Accept function byte input to define format, precision, etc.

Example:- C = A x B

A: 1-byte unsigned integer, scaled B0

B: 2-byte 2's complement integer scaled B8

C: 3-byte floating point

	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0
A	$.2^{-1}$	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
B	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	$.2^{-1}$	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
sign and exponent	±	e_6	e_5	e_4	e_3	e_2	e_1	e_0
mantissa, MS 1/2	$.b_{15}$	b_{14}	b_{13}	b_{12}	b_{11}	b_{10}	b_9	b_8
mantissa, LS 1/2	b_7	b_6	b_5	b_4	b_3	b_2	b_1	b_0

Fig. 15

SINE/COSINE SUBROUTINE

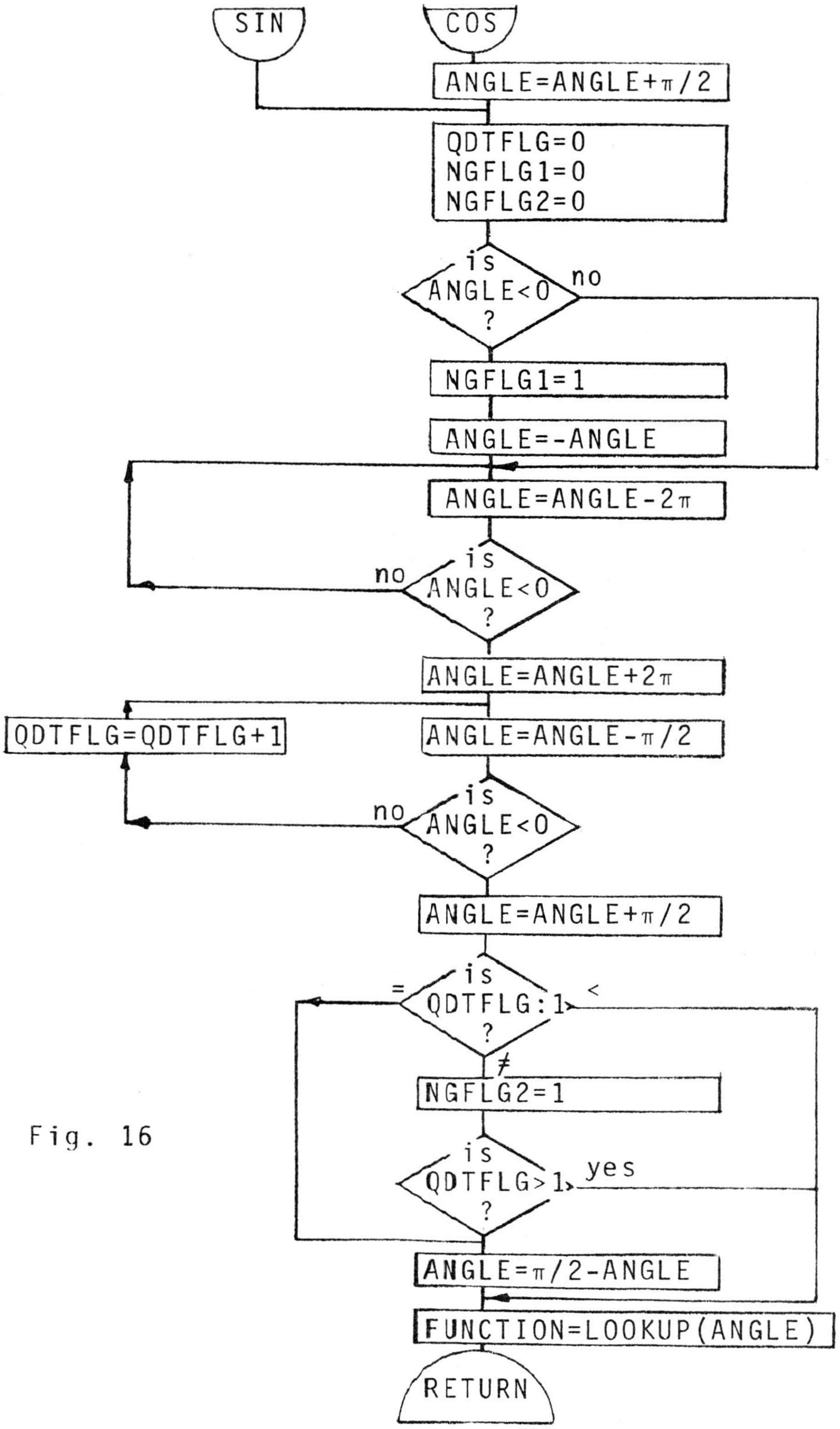


Fig. 16

PSEUDO DIVISION

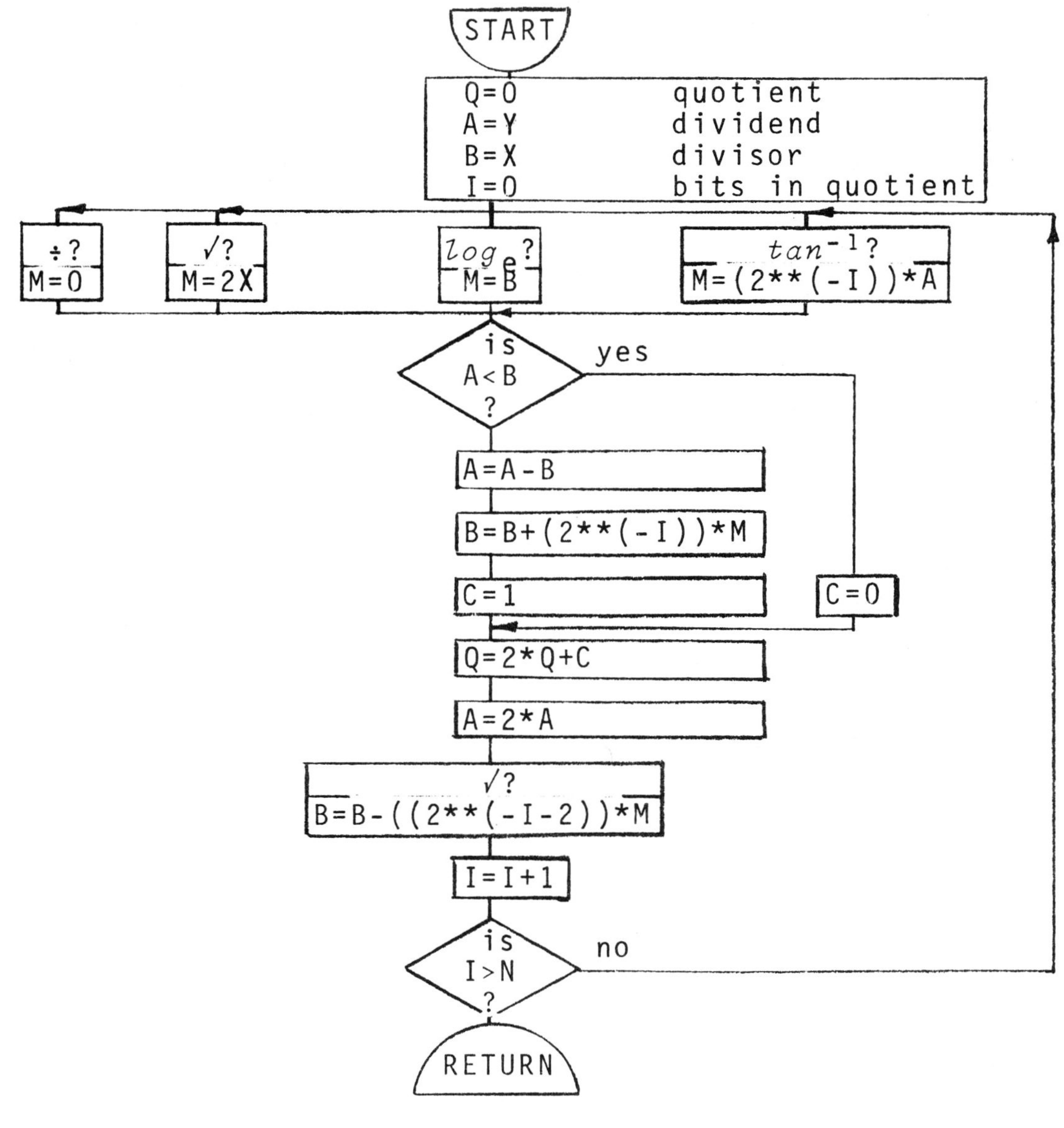


Fig. 17

REFERENCES

[1] Anon., "Assembly Instructions for the Motorola M6800 Design Evaluation Kit", Motorola Semiconductor Products, Inc., (/75)

[2] W. DEJKA, "Workshop Report: Microprocessor Standardization Concepts", Computer, (2/77), pp. 54-56

[3] I. GANAPATHY and N. JAFFE, "Enhancing the Graphic Capabilities of a Storage Scope Using Microprocessors", Proc. CIPS/CSA Canadian Computer Conference Session '76, (5/76), pp. 401-431

[4] R. GUTHRIE, "Build this Mathematical Function Unit", Byte, Part I: (9/76), pp. 26-33, Part II: (10/76), pp. 74-80

[5] J. MEGGITT, "Pseudo Division and Pseudo Multiplication Processes", IBM Journal, (4/62), pp. 210-226

[6] M. MILGRAM, "Lecture #10; Microcomputer Software", Course 308-765, McGill School of Computer Science, (2/77), pp. 1-5 and Private Communication, (3/77)

[7] R. MERRITT, "Minis and Micros ...competing for control", Instrumentation Technology, (2/77), pp. 45-52

[8] M. MULDER, "Model Curricula for Four-Year Computer Science and Engineering Programs: Bridging the Tar Pit", Computer, (12/75), pp. 28-33

[9] P. RONY and D. LARSEN, "Teaching Microcomputer Interfacing to Non-Electrical Engineers", Computer, (1/77), 53-57

[10] M. SLOAN, "Survey of Electrical Engineering and Computer Science Departments in the U.S.", Computer, (12/75), pp. 35-42

[11] M. WILES and A. FELIX, "Engineering Note 100 MCM6830L7 MIKBUG/MINIBUG ROM", Motorola Semiconductor Products, Inc., (/75)

[12] P. ZSOMBOR-MURRAY, "Computer Exercises in Engineering Graphics", Proc. CIPS/CSA Canadian Computer Conference Session '76, (5/76), pp. 462-493