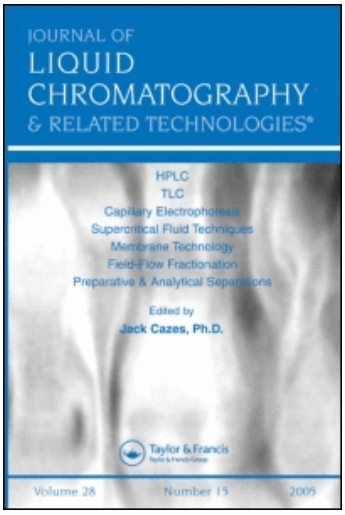


Journal of Liquid Chromatography & Related Technologies
Publication details, including instructions for authors and subscription information:
http://www.informaworld.com/smpp/title~content=t713597273

Hardware and Software for Microprocessor Controlled HPLC: Interfacing of Converters (A/D and D/A) for Data Acquisition and Display

Ram P. Singhal[a]; David B. Smoll[a]
[a] Department of Chemistry, The Wichita State University, Wichita, Kansas

To cite this Article Singhal, Ram P. and Smoll, David B.(1986) 'Hardware and Software for Microprocessor Controlled HPLC: Interfacing of Converters (A/D and D/A) for Data Acquisition and Display', Journal of Liquid Chromatography & Related Technologies, 9: 12, 2661 — 2693
To link to this Article: DOI: 10.1080/01483918608076891
URL: http://dx.doi.org/10.1080/01483918608076891

JOURNAL OF LIQUID CHROMATOGRAPHY, 9(12), 2661-2693 (1986)

HARDWARE AND SOFTWARE FOR MICROPROCESSOR CONTROLLED HPLC: INTERFACING OF CONVERTERS (A/D AND D/A) FOR DATA ACQUISITION AND DISPLAY

Ram P. Singhal and David B. Smoll
Department of Chemistry
The Wichita State University
Wichita, Kansas 67208

ABSTRACT

To allow direct acquisition and "real-time" processing of chromatographic data, hardware interfaces and a software operating system are described here for a dual-processor Hplc controller. The interfaces consist of a multiplexed, analog-digital converter, and digital-analog converters with built-in event marker circuits. The analog amplifiers for the converters are specifically designed to permit interfacing of conventional Hplc detectors and recorders to increase the flexibility of the system. A digital logic design is developed that minimizes interface servicing during "real-time" program execution. The software operating system consists of machine-language "kernel" routines for communication between the two processors, and high-level language programs to simplify the oper-

This is a second paper in this series (see ref. 12 for first paper).

Address all correspondence to RPS. Inquire RPS for availability of complete program listings on floppy disk or hard copy (Tel: 316-689-3120). Program listings are also described in the dissertation of D. B. Smoll; it can be loaned from Ablah Library, The Wichita State University, Wichita, KS 67208.

Abbreviations: A/D, analog-to-digital; D/A, digital-to-analog; HPLC, Hplc, High-performance Liquid Chromatography; LC, liquid chromatography; DHC, data handling microcomputer; see text for other abbreviations.

Note: Apple is a registered trademark of Apple Computer, Inc. and Explorer-85 is a registered tradmark of Newtronics Research and Development, Ltd.

 0148039l9/86/0912-2661$3.50/0

ator-interface. The "kernel" is designed to provide the necessary routines for execution of programs, storage and retrieval of chromatographic data, and display of results. The high-level language programs allow tailoring of data processing according to the application by the investigator.

INTRODUCTION

Chromatography is used for both quantitative and qualitative analyses of components (peaks) of a complex mixture. Peak heights, peak areas, and retention factors are determined to identify the components and determine their concentration (1-4). However, manual acquisition of the information embodied in the chromatogram, especially when resolving a complex mixture of solutes, is slow, imprecise, and cumbersome. The investigator can become overwhelmed trying to calculate, correlate, and analyze the data generated from a single chromatogram.

These difficulties prompted us to search for a convenient means of analyzing chromatographic data. The procedure should allow us to collect, analyze, store, and retrieve the data accurately and reliably. Though mini and mainframe computers are used to analyze and correlate data (5), the initial cost of these computers, and subsequent additional cost and the problem of interfacing them with existing or new chromatography equipment make such devices unattractive. Moreover, space and environmental requirements, limited access, and complexity of the equipment further make these computers unsuitable for routine laboratory applications (6).

Advances in microprocessor technology have provided a very powerful and affordable alternative. Data base, data spread-

sheets, and other application programs for chromatography have been developed recently for a limited number of personal computers (6-11). However, these software packages are often designed for specific aainstruments and fail to offer provisions for expansion and modification in order to meet specific needs of different investigators employing diverse chromatography equipment.

To solve this problem, a cost-effective, microcomputer-based chromatography controller and processor is described here which also provides necessary hardware and software interfaces. Recently, we have described hardware for asynchronous monitoring of chromatographic events in "real-time" (12). Here, hardware is designed to enhance versatility of the LC/computer/processor system (DHC) in order to permit use of chromatography equipment commonly available in the laboratory. The complete system contains software designed to provide the required software tools and operator interface for custom software development and implementation.

A. Hardware Design

The chromatographic controller/processor efficiently utilizes the specific features of two independent central processing units (CPU's) linked (programmed) in a master - slave confiquration. This configuration offers the full advantage of the register--oriented instruction set, the expandable vectored interrupt system, and the high-interfacing capability of an 8085-based Explorer-85, employed as the slave microcontroller; the memory--oriented intsruction set, and extensive software and hardware support of a 6502-based Apple microcomputer, used for the master

microcomputer (13-15). This arrangement provides "real-time" programming capability, instrument interfacing, floating-point arithmetic, and peripheral support necessary for data acquisition, processing, and retrieval.

Hplc equipment interfacing is achieved through analog-to--digital (A/D) converters, digital-to-analog (D/A) converters, interval timers, and an interrupt controller. Standard interfacing techniques and circuitry that employ readily available medium-scale and large-scale integrated circuits (IC's) are used as described in the literature (16-18). The circuitry is designed to promote "real-time" programming of the Explorer microcomputer by minimizing the programming required to service interfaces and interrupts (19).

To increase the flexibility of the analog interfaces, the analog input amplifiers for the A/D converter are designed to permit easy and accurate calibration. The input bias (quiescent--input current) of standard operational amplifiers (op amps) produces a substantial aoutput offset if some provision is not included to null it out. Unfortunately, conventional output--offset nulling circuits for op amps also depend on input circuitry resistance when used on DC-coupled amplifiers (20). This interaction between nulling circuits and input resistance complicates calibration when an input-level adjustment is used to adjust the gain of the op amp. The problem is generally solved by using an AC-coupled chopper amplifier (21). However, as an alternative, we have overcome this problem by using a track-and-hold

amplifier/buffer for the analog inputs of the A/D converter interface constructed from a differential transistor array IC. We find this circuit simple and inexpensive. It satisfactorily eliminates the input-bias current, and thereby makes the output offset independent of the gain adjustment for the DC-coupled op amp.

B. Operating System Design

The purpose of an operating system is to supervise and control computer operation (22). This includes scheduling of tasks, responding to events, e.g., interrupts and commands, and maintaning data bases. These functions are performed as a supplement to the application software which can be considered as another task managed by the operating system. In this context the operating system functions as a system supervisor and makes the execution of application programs invisible to the operator. In the environment of our dual-microprocessor DHC, the operating system also manages communication between the two computers.

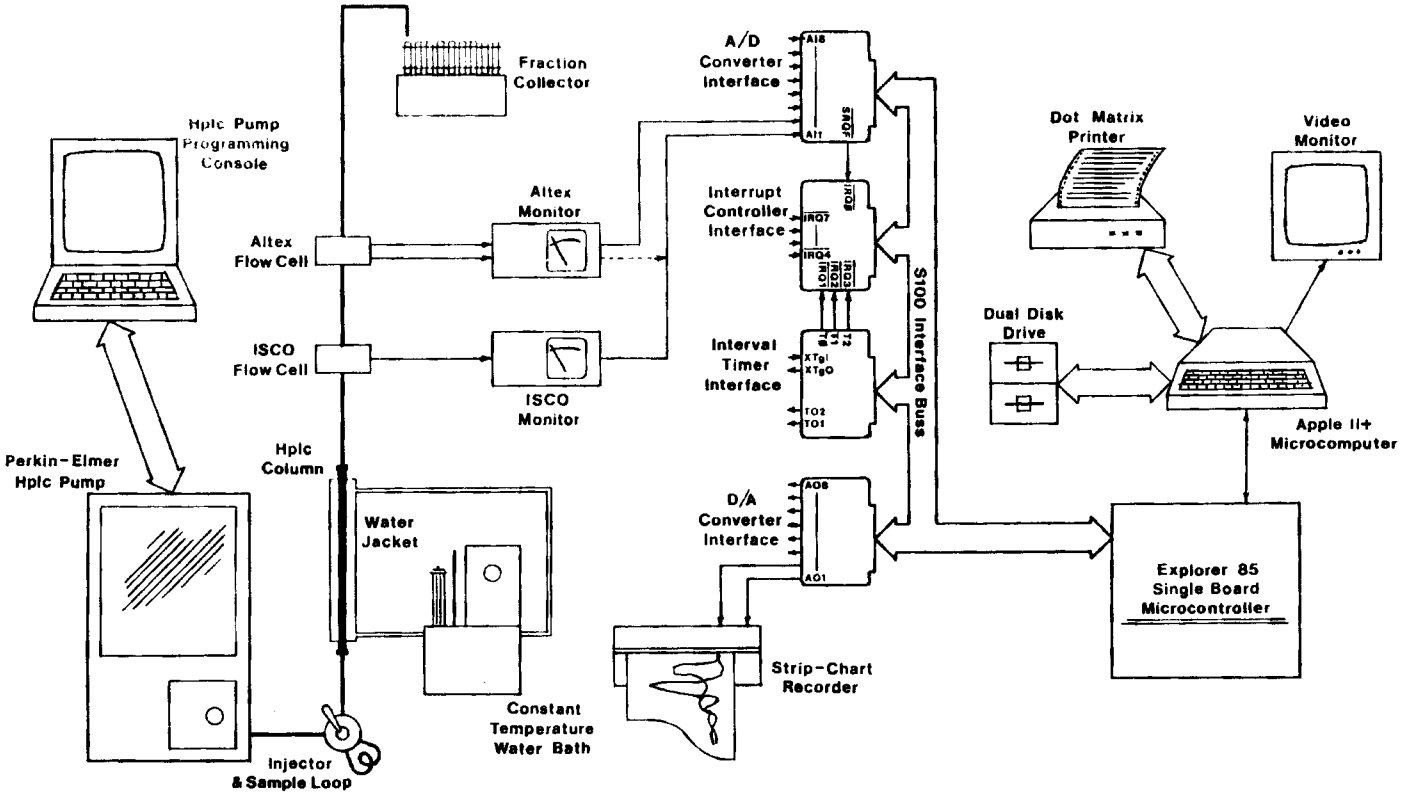
The operating system for the Hplc DHC described here consists of two software packages -- the System Supervisor and Turnkey. The System Supervisor is responsible for communication between the Explorer and the Apple computer for development and execution of Explorer programs, as well as transfer of data and programs from the Apple computer to the Explorer and vice versa. The supervisor provides a "kernel" consisting of independent routines which can be used by other programs in the Apple computer. Its modular design allows the program to remain highly unstructured, permitting random execution of kernel routines (23). The supervisor,

including the kernel, is written in machine language to minimize computer memory required as well as to maximize execution speed for "real-time" monitoring of the Explorer operation.

The Turnkey program is designed to simplify the operator--interface with the DHC. It is written in Applesoft BASIC as a highly structured program. This structure promotes the constant use of "menus" and "prompts" to offer user-friendly operation of the DHC (24). The use of a high-level language such as BASIC allows it to be easily tailored to experimental requirments by the investigator, and offers a "floating-point" package for data processing.

MATERIALS AND METHODS

The computer equipment consisted of two interconnected computers which included peripherals and interfaces. The Hplc equipment was interfaced directly to an Explorer-85 single-board microcontroller obtained from Newtronics Research and Development Ltd. (New Milford, CN) as is illustrated in Scheme I. This microcontroller included a standard S-100 Buss System (25) and was modified to include an inverted RS-232 serial-data port and 12K of RAM memory. The full set of input/output (I/O) port addresses were provided by modifying its on-board memory-address decoder. The data buss, address buss, and control signals for the interfaces were derived directly from the 8085 CPU through the S-100 Buss. The Explorer was programmed using MCS 80/85 machine--language to acquire, manipulate, and display chromatographic data.



Scheme I. An Overview of Microcomputer-Assisted High Performance Liquid Chromatography System.

Data and programs were transferred from the Explorer to an Apple II+ computer through the RS-232 serial port of the Explorer and an Apple Super Serial Interface Card (Apple Computer Inc., Cupertino, CA). The Apple computer was equipped with two (floppy) disk drives and a disk controller and was expanded to a full 48K RAM. Once a set of data was transferred to the Apple computer, it was saved on a 5-inch floppy disk for further processing. Communication between the Explorer and Apple computers was achieved by programming the Apple computer with a 6502 machine-language program and using the Explorer's resident operating system monitor.

Other peripherals of the Apple computer included a video display (model BM-12AUW, 15 MHz, BMC International, Japan) and a dot-matrix printer (Prowriter, model 8510, Leading Edge Products Inc., Canton, MA) interfaced with an Apple II Parallel Interface Card (Apple Computer Corp., Cupertino, CA).

The analog-output signal of the UV monitor was converted to an 8-bit digital value by an Analog-to-Digital (A/D) interface built with an ADC0809 8-bit, 8-channel, Multiplexed A/D Converter (National Semiconductor Corp., Santa Clara, CA). The calibration of the A/D converter was performed through an amplifier/buffer in a manner to yield full-scale absorbance equal to full-scale monitor analog output.

A set of data was displayed on a strip-chart recorder after the DHC processed the data (ISCO, model 613, single-pen recorder or a Linear, model 232, dual-pen recorder). The display was achieved by converting the digital data back to an analog voltage with a two channel Digital-to-Analog (D/A) interface. Each D/A

interface channel was built using a DAC1020 10-bit Binary Multiplying D/A converter (Nation Semiconductor Corp., Santa Clara, CA). The output of the D/A converter was amplified and combined with an event-mark signal using a summing operational amplifier. The calibration of each converter was carried out using this amplifier, matching 255, the full-scale digital value of DHC, to a full-scale recorder value.

Timing of the program cycles (DHC) was provided by an Interval Timer/Interrupt Controller interface (12). The Interval Timer with interface furnished three totally independent and synchronized timers which controlled the 8085 Central Processing Unit (CPU) of the Explorer-85 via an Interrupt Controller. The Interrupt Controller provided a total of eight programmable vectored interrupts for the system. Of these, three interrupts were used by the timers and one by the A/D converter. The remaining interrupts were not used at this time, although they are available for future expansion.

HARDWARE INTERFACE DESIGN

A. Description of A/D Hardware Interface

The hardware for the A/D converter scaled analog voltages and converted them to an 8-bit digital value. This interface had eight separate and independent analog channels but only one converter. Consequently, only one channel could be read and converted at one time, requiring approximately 100 microseconds. If a conversion "start" instruction was issued prior to completion of

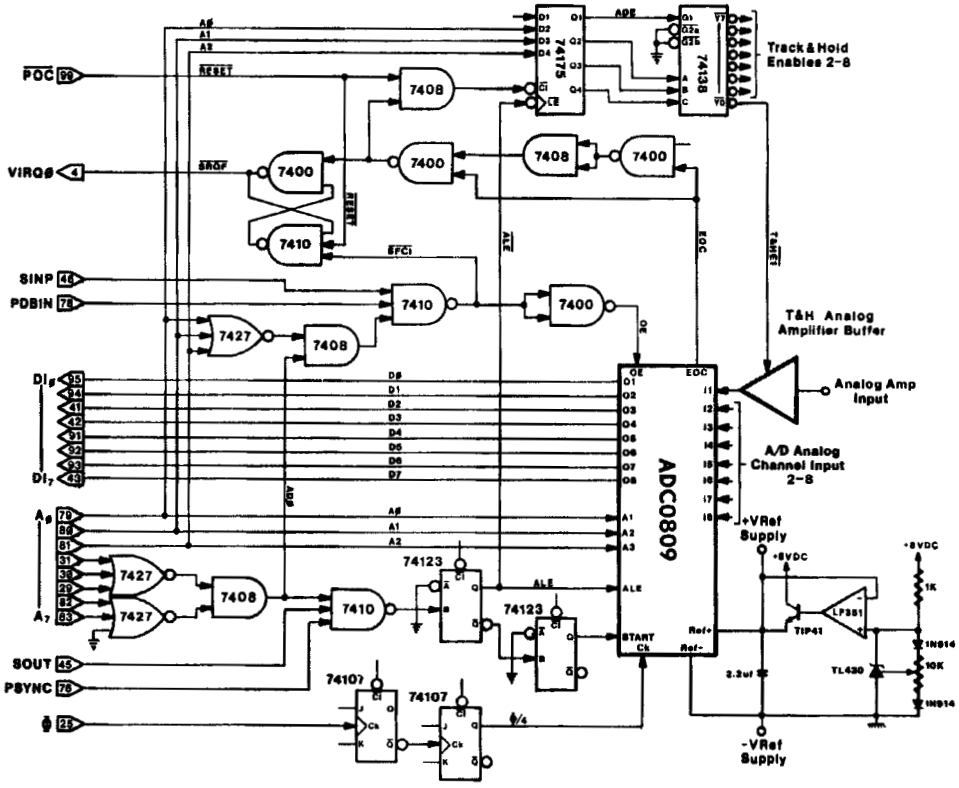
a conversion, the present conversion would be aborted and a new one started. However, more than one analog-input channel could be used since channels were independently addressable. The hardware required only that the conversion be completed and that the output register be read before starting a new conversion.

1. Digital Logic

The function of the digital logic was to generate necessary electrical signals for the conversion only after receiving the correct instruction with the appropriate address. The conversion process was initiated with an OUTput machine-language instruction using the address of the desired analog channel ($00 - $07). (Each hexadecimal value is specified throughout this article by a dollar sign before the value.) No other instructions were necessary until the conversion was completed; thereafter, an INput instruction addressing port ($00) read the output register of the converter.

A/D Converter. Each analog-input channel of the A/D converter was connected to a separate and independent analog--amplifier/buffer as shown in Scheme II. The use of the ADC0809 A/D converter provided a total unadjusted error of ±0.5 LSB (26). The reference-voltage inputs, Vref(+) and Vref(-), were connected to a separate, regulated 5VDC reference supply isolated from the 5VDC logic power supply. This provided the converter with a stable voltage having less than 10 mv of ripple and noise as measured between the two reference terminals.

Reset. The reset logic cleared all interface logic upon power up of the Explorer, and it prepared the interface for a con-



Scheme II. Diagram of Analog-to-Digital Converter Interface Logic. Interface converts analog voltages from Analog Amplifier/Buffer (see Scheme III) to an 8-bit digital value for use by the microcomputer. Control logic is shown here for selection and control of track-and-hold stage of appropriate amplifier. All unused logic inputs are connected to +5 volts DC unless otherwise indicated.

version "start" instruction. The logic used the POC line of the S-100 Buss activated on pressing the "RESET" key of the Explorer (see Scheme II). The logic, when activated, cleared the Service Request Flag and the Track-and-Hold Address latch, thereby disabling its address decoder.

Analog Channel Addressing. The channel-addressing was handled by the addressing logic within the A/D-converter interface. This hard-wired logic presented the 3-bit channel address on address lines A0, A1, and A2 to the converter's address-decoder unit while using the remaining address lines, A3 through A7, SOUT, and PSYNC signals to generate an Address Latch Enable (ALE) signal. The ALE signal was then used to latch the analog-channel address into the converter's address latch. Immediately thereafter, on the falling edge of ALE, a converter "start" signal was generated to initiate the conversion process. This signal generation required approximately 0.2 microseconds from the falling edge of the PSYNC signal.

Track-and-Hold Addressing. The 3-bit analog channel address was latched simultaneously into the Track-and-Hold Address Latch, along with a fourth Address Decoder Enable (ADE) bit by the above ALE signal. The ADE bit enabled a 3-8 decoder to decode the latched 3-bit address which selected one of the eight Track-and--Holds. The selected Track-and-Hold was used to provide a stable analog input to the A/D converter during the conversion.

End-of-Conversion Signal. The converter was driven by a 768 kHz clock derived from the system clock of the S-100 Buss. The converter required no further servicing until the conversion was done. At this time, the End-of-Conversion (EOC) line of the converter returned "high". (A "high" condition for TTL logic is defined as a voltage greater than +2.4 VDC.) The EOC line was used to set the Service Request Flag (SRQF) of the interface and

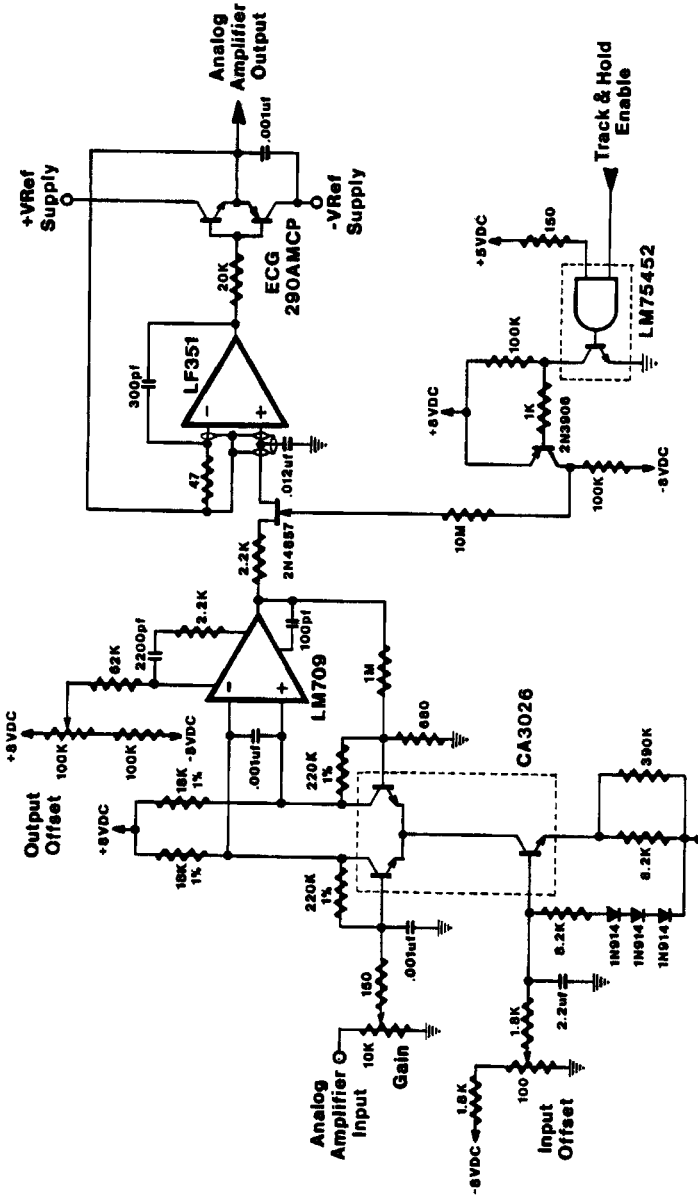
to clear the Track-and-Hold address latch. This disabled the 3-8 decoder and allowed the selected Track-and-Hold to resume tracking its input. The SRQF was communicated to the CPU via an interrupt of the interrupt controller (VIRQ0 line of the S100 Buss).

Output Register. The address of the output register was placed on the address lines and decoded by the address-decode logic of the interface on execution of an INput instruction addressing this register. This decoded address was combined with the SINP and PDBIN signals of the S-100 Buss to generate an Output Enable (OE) signal, which enabled the output register of the converter to place the digital value of the last completed conversion on the Data-In-Buss. Simultaneously, the SRQF was cleared, and the interface was prepared for the next "start" instruction and another conversion.

2. Analog, Zero-Offset, Track-and-Hold Amplifier/Buffer

Processing of the analog-input voltage for each analog-input channel of the A/D converter was handled by a separate Zero--Offset, Track-and-Hold Amplifier/Buffer as shown in Scheme III. These amplifiers allowed totally independent calibration of each analog-input channel for full-scale conversion. The overall nonlinearity of the amplifier was demonstrated to be less than 0.1% over full-scale.

Gain Stage. The amplifier/buffer consisted of three stages connected in series to provide three functions (Scheme III). The first stage was a 60 dB gain stage with an adjustable input level which provided the overall gain for the amplifier. A transistor



Scheme III. Diagram of Track-and-Hold Analog Amplifier/Buffer Electronics. Input voltages are amplified, and the output of the amplifier/buffer is connected to the analog inputs of A/D Converter Interface (see Scheme II for detailts). The track-and-hold stage is enabled during the conversion process.

array (CA3026 Dual Independent Differential Amplifier, RCA Corp., Somerville, NJ) was used on the input of an LM709 op amp (National Semiconductor Corp., Santa Clara, CA) in order to null out completely the input offset. This rendered the output offset totally independent of the gain setting and provided better than 60 dB supply rejection at the input. Furthermore, the output offset was adjustable, permitting zeroing of the amplifier regardless of the offset of the input voltage. Use of two temperature-conpensating diodes in the constant-current emitter circuit of the transistor array virtually eliminated thermal drift amplifier over a thirty degree range (20 — 50°C).

Track-and-Hold Stage. The output of the gain stage was coupled to the noninverting input of a high-input impedance (10ΩG) bi-FET LF351 op amp (National Semiconductor Corp., Santa Clara, CA) through a FET chopper. This track-and-hold stage, normally in the tracking mode, held its input voltage as long as a logic "high" signal was applied to its TTL-compatible-logic input. Drift was reduced to 10 mv/sec with the use of input guarding and a polystyrene charge-holding capacitor of this stage (27).

Output Buffer Stage. The final stage was a matched complementary transistor pair (TCG290AMCP, New-Tone/ECG Electronics, Bloomfield, NJ), which followed the output voltage of the track--and-hold stage. Connection of the transistor collectors to the isolated VRef supply prevented the final output voltage from exceeding the analog-voltage range of the converter's analog inputs, and provided a stable voltage with less than 10 mv of

ripple and noise. The current amplifying capability of this push--pull output also significantly reduced modulation of the output due to rapid consecutive readings.

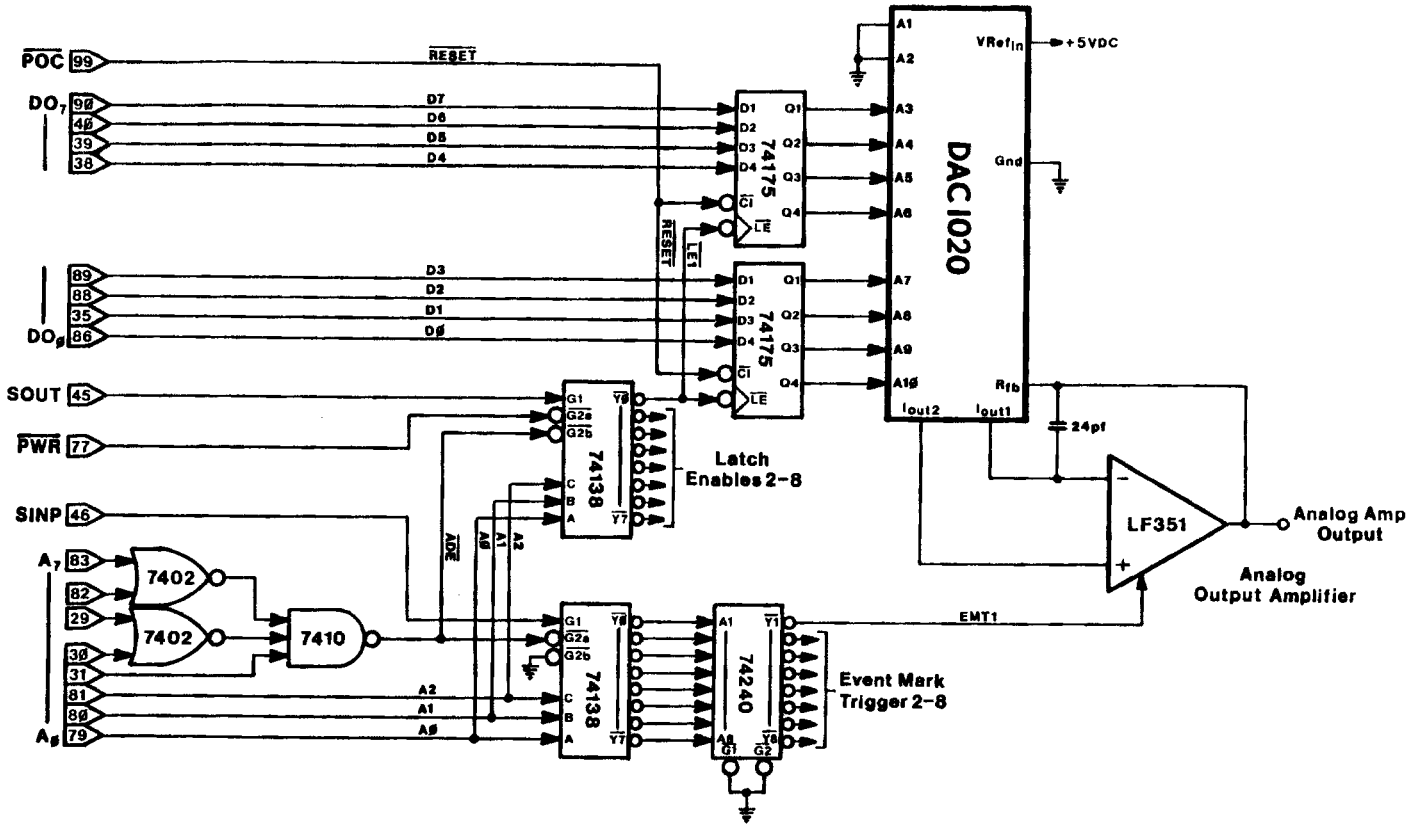
B. Description of D/A Hardware Interface

The D/A converter hardware handled the conversion of 8-bit digital data to analog voltages and amplified them to a final maximum voltage of one volt into a 10ΩK input impedance. Each of the possible eight channels was totally independent and could be used concurrently. The interface circuitry also included an Event Mark circuit which, when addressed with an INput instruction, produced a sharp descending mark on a strip-chart recorder wherever the pen was located.

1. Digital Logic

The function of the digital logic was to provide necessary signals to read the Data-Out-Buss of the S-100 Buss and to trigger the event marker when addressed with the correct instruction. The D/A converter and event marker used the same address-decode logic but different status-control signals from the CPU. Consequently, each pair had the same address ($08 - $0F) but responded to different instructions. The data latches of the D/A converters were addressed with an OUTput instruction, since they received data, while the event markers were addressed with an INput instruction.

D/A Converter. The D/A interface is shown in Scheme IV with data latches and associated logic. The two most significant data lines of the 10-bit converter were connected to logic "low", thus converting it to an 8-bit converter. (A "low" condition for TTL



Scheme IV. Diagram of Digital-to-Analog Converter Interface Logic. Interface converts digital values from the microcomputer to analog voltages for amplification by Analog Output Amplifier (see Scheme V). Control logic is shown for selection and control of event-mark circuit located in the analog amplifier. All unused logic inputs are connected to +5 volts DC unless otherwise indicated.

logic was defined as a voltage less than +0.8 VDC.) The VRef terminal was connected directly to the +5 VDC supply of the interface. The settling time for the converter was stated to be 0.5 microseconds, and nonlinearity was 0.05% -- guaranteed over the temperature range, 0°to +70°C (28).

Reset. The reset logic was activated by the "RESET" key of the Explorer-85 through the POC line. Its sole purpose was to clear the data latches of the interface when the power was first turned on. Therefore, after pressing the "RESET" key the outputs of the D/A converters were set to zero.

D/A Converter Addressing. Addresses for all D/A converters were decoded by one 3-8 decoder using address lines A0, A1, and A2 (see Scheme IV). The remaining address lines, aaaaA3 through A7, were decoded as an "active low" Address Decoder Enable (ADE). This ADE signal was combined with SOUT and PWR to enable the D/A Address Decoder. This decoder selected the Latch Enable (LE) line of the data latches decoded from address lines A0, A1, and A2.

Event Marker Addressing. The event-marker addressing was handled in a similar manner as above, except that a separate 3-8 decoder was used. The ADE signal generated above and SINP signal of the S-100 Buss were used to select one of eight Event Marker Trigger logic inputs of the analog-amplifier circuits. (Data lines of the S-100 Buss were not used by this logic; therefore, the data read into the accumulator when addressing the event marker was meaningless.)

Data Latches. Data values on the Data-Out-Buss were latched into 8-bit data latches when its LE line was selected and returned

to a logic "high". As a result, the data value was presented to the data inputs of the D/A converter for immediate conversion.

2. Analog Output Amplifier

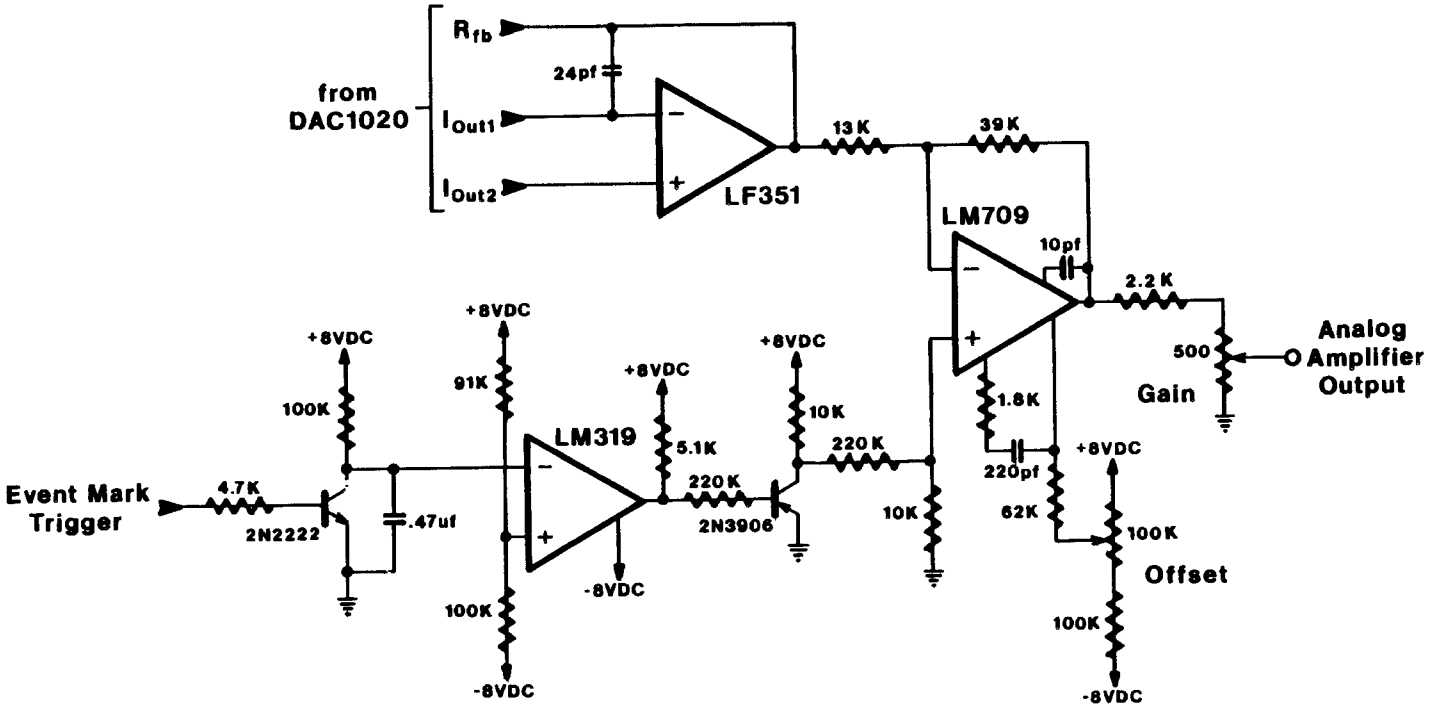
This amplifier consisted of two inverting op amps which, together, produced a noninverted output (see Scheme V). The first stage, an LF351 bi-FET operational amplifier, was connected to the D/A converter in the usual way (29). An output-offset adjustment was provided on this stage to be used to zero the output. Full--scale adjustment of the output was adjusted with a potential divider on the output of the last stage. The last stage was also used to sum the event-mark pulse into the output voltage.

3. Event Marker Amplifier

The event-marker circuit was designed to be used with a strip-chart recorder. This circuit was triggered by a TTL-level compatible logic "high" at its trigger input, but required a minimum of 1 ma drive current. The circuit produced a negative going pulse which was summed with the output voltage of the D/A converter. The descending mark produced on the recorder could be lengthened by issuing more than one trigger signal in succession.

OPERATING SYSTEM DESIGN

The software for the DHC consists of separate programs for the Explorer and the Apple computer. These programs can be further subdivided into operating systems and application programs for each microcomputer. The application programs are described in subsequent papers. This work will only deal with the operating systems for the two microcomputers.



Scheme V. Diagram of Analog Output Amplifier Electronics. Input voltages from D/A Converter Interface are amplified to a maximum of 1 volts (see Scheme IV). Event-mark circuit allows issuance of a short descending mark on a strip-chart recorder without changing the digital value in D/A converter.

A. The Explorer's Operating System Monitor

The operating system for the Explorer was purchased with the Explorer and it resided in its onboard read-only-memory (ROM). This machine-language system monitor decoded low-order ASCII characters transmitted through its RS-232 port. It provided basic functions for Explorer software development, memory manipulation, program execution, and operator-interface. It was used by the Explorer as the communication software interface between the Explorer and the Apple computer.

Upon power-up, the Explorer required an initialization sequence to initialize its CPU and to set its baud rate for its serial port. As soon as this procedure was completed, its resident system monitor could transmit or receive over the RS-232 channel.

B. Description of System Supervisor Program

The Apple computer software provided communication between the Explorer and the Apple computer using a machine-language program which resided between $8800 and $9600 in the Apple computer. This System Supervisor program was written with a modular design with well defined entry points and parameter tables. This design allowed the use of its routines by higher level languages with high level "CALL" and "USR" instructions (30).

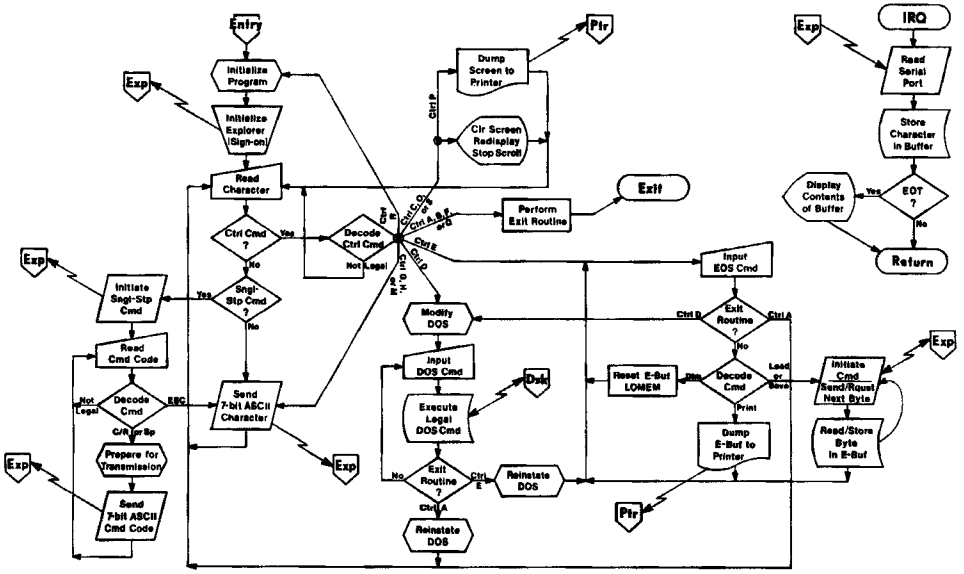
The program consisted of two primary modes: the Monitor Mode and the Command Mode. These modes provided necessary routines for communication between the Explorer, the Apple computer, disk drives, and other peripherals. Each routine was designed for use within the supervisor or by programs outside of the supervisor.

Data was transmitted from the Apple Computer to the Explorer or viz using a serial interface card as 8-bit low order ASCII characters. The transmission receiving routines were interrupt driven routines that used the maskable interrupt (IRQ line) of the Apple computer to alert it when data was being transmitted from the Explorer. The baud rate of all Explorer - Apple serial transmissions was 9600 baud (bits per second).

1. Monitor Mode

The primary mode of communication was the Monitor Mode. This mode provided a direct operator-interface with the Explorer. In this mode all alphanumeric characters entered via the Apple keyboard were transmitted directly to the Explorer. If the Explorer was in its resident System Monitor program, the ASCII code was decoded as an Explorer Monitor Command. All characters transmitted back to the Apple computer were displayed after being converted to high-order ASCII. Display of these characters was deferred until a low-order End-of-Transmission code, period ($2E) or dash ($2D), was received. By postponing display in this manner, a faster Baud rate was achieved. This mode was used to enter and "debug" Explorer programs, to issue commands to the Explorer, and to display transmissions from the Explorer.

The System Supervisor was normally entered to the Monitor Mode as shown in its flow diagram, Scheme VI. The IRQ Service Vector for the 6502, at $03FE and $03FF, was changed to the program IRQ Service Routine. The RAM memory in which the program and



Scheme VI. Flow Diagram of System Supervisor Program. Machine-language program residing in the Apple computer is responsible for communicating with the Explorer, printer, and for storage and retrieval of Hplc data on floppy disks.

data buffers resided ($6600 - $95FF) was protected from usage for BASIC programs by changing the value of HIMEM (located at $73 - $74). After a space character ($20) was sent to the Explorer by the Apple computer, the Explorer was initialized and communication between the Explorer and the Apple computer could continue.

The monitor mode did not generate error messages or codes. Instead, this task was delegated to the Explorer program. In most

instances, when either an error in transmission or illegal code was detected, a command default was performed which was indicated by the issuance of a default (asterisk for the Explorer System Monitor or a question mark for most programs) and a new prompt. This was often used to abort a monitor command, since it was not possible to edit commands in this mode.

2. Command Mode

Control characters ([Ctrl]) were used as command codes and were not transmitted to the Explorer with the exception of [Ctrl G], [Ctrl H], and [Ctrl M]. Each command mode could be entered by pressing the [Ctrl] character for that mode except when executing the Explorer single step command.

Monitor Commands. Pressing the [Ctrl] key while typing another key (character), entered the letter as a monitor command. These commands were used to exit the System Supervisor program to other Apple programs, to manipulate the Apple video screen, to print data, and to enter the other command modes.

The Monitor Command mode did not generate error messages or codes. The use of an illegitimate [Ctrl] character resulted in no transmission and a return to the monitor mode.

DOS Commands. The function of this command mode was to handle transfer of files between the Apple computer and disk drives without leaving the System Supervisor program. The command strings consisted of standard Apple DOS 3.3 command strings. (For full description, syntax, and error codes, consult the Apple II DOS manual.) The decoding of the command strings was accomplished

by entering the command strings through the "GETLN" subroutine (Apple computer ROM address $FD6A) without changing the DOS I/O hooks. When the supervisor program was used in conjunction with a BASIC program, the ISBASRUN routine ($AA65) of DOS was patched to always return an Immediate Mode return code, and the "ON ERR GO TO" Basic command was employed to intercept errors.

DOS command mode was entered by entering [Ctrl D]. After a command string was executed, the program returned to the DOS command mode and displayed a new prompt with cursor. However, control was transferred to the Apple BASIC Monitor when a BASIC program was loaded, saved, or run. The program was re-entered by executing a "CALL -28572".

The DOS error-handler detected errors and generated error messages and codes because the DOS command interperter was used to decode these commands. If the execution of a non-DOS command string was attempted and the DOS error-handler did not intercept it, an "ILLEGAL COMMAND" error was generated with no action taken. The program returned to the DOS command mode with the display of a new prompt and a cursor.

EOS Commands. The primary function of this command mode was to facilitate transfer of large blocks of binary data and programs between the Explorer-85 and the Apple computer with the exception of the "Print" and "Dim" commands. Data values were transferred from (or to) consecutive-memory locations in the Explorer to (or from) the E-buffer in the Apple computer as a single-byte binary number. The E-buffer resided in protected RAM of the Apple com-

puter at $6600 to $8800 (default values). However, the buffer size could be extended to $8E00, if necessary, without generating a "Buffer Overflow Error". Furthermore, a "Dim" command could further extend the size of this buffer by lowering its starting address as low as $0800. (This was outside the protected RAM.)

The EOS command mode was entered by pressing [Ctrl E] and consisted of four basic command strings. The commands "LOAD" and "SAVE" transferred the blocks of data and required the Explorer to be in its Resident System Monitor. Otherwise, a "Command Initiation Failure" error was generated and the command was aborted. Once the data had been transferred to the E-buffer, the block of raw data could be printed with the "PRINT" command. The fourth command, the "DIM" command, already mentioned as a means to extend the size of the E-buffer, was also useful for storing multiple sets of data in the E-buffer at the same time.

The command strings consisted of the command followed by a list of parameters. The syntax for most command strings was similar to the binary data file transfer DOS commands -- "command" A$[starting memory address],L$[length of data block]. The "DIM" command, on the other hand, only required the E-buffer's starting address to be specified. Only these commands were accepted, otherwise, an "Illegal Command" error was generated. Any ommition of spaces, commas, or parameters produced a "Syntax Error". In addition to this command string entry checking, transmission validity checking ensured valid data transfer. This was accomplished by comparing data echoed by the Explorer with the original data.

Upon detection of a transmission error, an error handling routine generated an "I/O Error" message. The data count where the error occurred, and an error code specifying the nature of the error, was displayed.

The EOS commands were also written as modular routines to allow use by high level programs. When used in this manner, specific parameter locations were loaded with the starting memory address and byte count prior to a program "CALL" to the appropriate machine language routine. After completion of the command, the program returned to the next program statement. This high level program utilization of these routines also provided the same transmission validity testing and error handling as if the command was issued from the System Supervisor.

C. Description of Turnkey Operating System.

The Turnkey program was designed to automate and simplify the operational-interface and data processing of DHC. The program was written in "APPLESOFT BASIC" for the Apple computer as three separate segments in order to conserve the Apple computer memory. The segments were written as prompting programs which loaded programs and data from menus. The first segment of the Turnkey which was reponsible for loading the patched version of DOS, loading the System Supervisor, and initializing the Explorer was automatically booted into the Apple computer upon power-up. The other segments were loaded and run by the Apple computer depending on which Explorer program or procedure was selected.

The Turnkey program utilized the machine-language routines of the System Supervisor to automatically initialize the Explorer, load Explorer programs, and save chromatographic data. These routines were entered with high level "CALLs" and "USR" defined statements after loading the routine parameter table with "POKEs". Control was returned to the next "BASIC" statement in the Turnkey program after completion of the Supervisor's routines. Error detection and error handling within the Turnkey program were handled by "ON ERR GO TO" satements.

DISCUSSION

Liguid chromatography (LC), since its conception, has been employed to isolate and purify components of complex mixtures. Information derived from chromatography has furnished quantitative and qualitative information about each component. The problem has been how to conveniently acquire and assimilate this data suitably for routine investigation. To simplify and facilitate data collection, quantitation, and characterization, we have satisfactorily interfaced a typical Hplc detector to a dual-processor controlled data-handling computer (DHC).

A. Hardware for Data Handling Equipment

The DHC described here can be divided in two hardware components -- data processing equipment and equipment monitoring interfaces. The primary responsibility of the data processing equipment is to manipulate the data and control the interfaces. The dual-processor configuration used here offers: (a) the speed of

execution of the register-oriented-8085-instruction set of the Explorer, (b) and an abundance of support chips, and (c) a highly interfaceable buss (the S100 Buss) for interfacing equipment. Therefore, the Explorer provides the components needed for controlling the equipment and the data processing required during "real-time".

The Apple computer, on the other hand, complements the speed and the interfacing capability of the Explorer. The data processing power of the memory-oriented-6502-instruction set and the software support of the Apple computer provide a "user-friendly" environment for programming of non-time-critical processing of chromatographic data. In addition, peripherals such as disk drives, CRT monitors, and printers are readily available for the Apple computer; hence, they provide a convenient medium for data storage and display. As a consequence, the Apple computer provides the necessary components for the operator-interface to the DHC.

The interface between the DHC and the Hplc equipment is provided by the equipment monitoring interfaces. This hardware is specifically designed to meet the requirements that we established for the DHC. Calibration of analog amplifiers are simplified and extended over a wide voltage range to accommodate a variety of instruments. This feature increases the adaptability of the DHC to different monitors and recorders. The expansion of the Explorer's vectored interrupt system provides a means to communicate external events to the Explorer's CPU. Hence, effective, asyn-

chronous "real-time" monitoring of events and instruments is possible. Finally, the digital logic of the interfaces is designed to reduce interface servicing during the experiment to a single two-byte machine-language instruction. For example, functions such as analog channel addressing, activation of appropriate track-and-hold, and converter initiation, are performed simultaneously with a single OUTput instruction. Moreover, the A/D converters output register is read and its SRQF is cleared simultaneously with a single INput instruction. Similarly, event-marks are issued, and the timers are started and synchronized with minimum disruption of program execution. In this manner, the design of the interfaces promotes efficient "real-time" programming of the DHC.

B. Software for Data Handling Equipment

The hardware of our DHC is made available to the investigator through the operating system. This is achieved through a series of interactive programs which reside in the memories of the Apple and the Explorer. A machine-language "kernel" is provided for communication between the two computers and the peripherals by System Supervisor programs. These programs furnish a means to load, store, and display programs and chromatographic data. Since they are written in a modular form, their routines are also available for other programs.

The other programs co-reside in the Apple computer with the System Supervisor. Their primary function is to provide a versatile operator-interface with the Apple computer and the Explorer

using the supervisor's "kernel" routines. They are automatically booted into the computer after power-up of the DHC to provide a Turnkey Operating System. Since they are written in BASIC, they make full use of the "user-friendly" software support provided with the Apple computer. The Turnkey programs are quite flexible and can be written with the specific laboratory application in mind. They are designed to allow the user to expand the DHC's current program capabilities.

ACKNOWLEDGEMENTS

This research was supported by grants from The Wichita State University and National Institutes of Health, grant no. #GM36099.

REFERENCES

1. Engelhardt, H., in High-Performance Liquid Chromatography, Chemical Laboratory Practice, Springer-Verlag, New York, (1979), p. 6.

2. Mayer, S. W. and Tompkins, E. R., Ion exchange as a separation method. IV. A theoretical analysis of the column separation process, J. Amer. Chem. Soc., **69**, 2866, (1947).

3. Cohn, W. E., The anion exchange separation of ribonucleotides, J. Amer. Chem. Soc. **72**, 1471, (1950).

4. Singhal, R. P. and Cohn, W. E., Analytical separation of nucleosides by anion-exchange chromatography: influence of pH, solvents, tenperature, concentration, and flow rate, Analt. Biochem., **45**, 585, (1972).

5. Singhal, R. P., Ray, R. C. and Dobbs, L., Computer program for storage and retrieval of the nucleic acid structure, Compt. Prog. Biomed., **14**, 227, (1982). For an extensive list of references, refer to The Applications of Computers to Research on Nucleic Acids, Soll, D. and Roberts, R. J., eds., IRL Press, Oxford, (1982).

6. Long, E. C., Managing laboratory data on a desktop personal computer: part three, Amer. Lab., **16**, 96, (1984).

7. Long, E. C., Managing laboratory data on a desktop personal computer: part two, Amer. Lab., **16**, 112, (1984).

8. Walling, P. L., Gentille, T. E. and Gudat, A. E., Amer. Lab., **14**, 23, (1982).

9. Harder, M. E. and Koski, P. A., Amer. Lab., **15**, 27, (1983).

10. Tehrani, A. Y., Automating an existing analytical chromatograph for preparative HPLC, Amer. Lab., **17**, 43, (1985).

11. Li, J., Hillier, E. and Cotter, R., Computerized HPLC detection in pharmaceutical research, Amer. Lab., **7**, 93, (1985).

12. Smoll, D. B. and Singhal, R. P., Hardware for microprocessor controlled HPLC: interfacing of an interval timer and interrupt controller to the S100 Bus System, J. Liquid Chromatogr., **6**, 2095, (1983).

13. Artwick, B. A., in Microcomputer Interfacing, Lsaacson, P., Ed., Prentice-Hall, Inc., Englewood Cliffs, (1980), p. 33.

14. Buchsbaum, W. H. and Weissenberg, G., in Microprocessor and Microcomputer Data Digest, Reston Publishing Co., Inc., Reston, (1983), p. 1.

15. Cassell, D. A., in Microcomputer and Modern Control Engineering, Reston Publishing Co., Inc., Reston., (1983), p. 197.

16. O'Conner, P. J., in Digital and Microprocessor Technology, Prentice-Hall, Inc., Englewood Cliffs, (1983), p. 11.

17. Artwick, B. A., in Microcomputer Interfacing, Lsaacaon, P., Ed., Prentice-Hall, Inc., Englewood Cliffs, (1980), p. 144.

18. Ogdin, C. A., in Microcomputer Design, Prentice-Hall, Inc., Englewood Cliffs, (1978), p. 103.

19. Ogdin, C. A., in Microcomputer Design, Prentice-Hall, Inc., Englewood Cliffs, (1978), p. 145.

20. Stout, D. F., in Handbook of Operational Amplifier Circuit Design, Kaufan, M., Ed., McGraw-Hill Book Co., New York, (1976), p. 2.1.

21. Morrison, R., in DC Amplifiers in Instrumentation, John Wiley & Sons, Inc., New York, (1970), p. 138.

22. Peterson, J. L. and Silberschatz, A., in Operating System Concepts, Addison-Wesley Publishing Co., Reading, (1980), p. 1.

23. Campbell, S., in Microcomputer Software Design, Prentice-Hall, Inc., Englewood Cliffs, (1983), p. 46.

24. Cohen, A., in Structure Logic and Program Design, John Wiley and Sons, New York, (1983), p. 133.

25. Poe, C. P. and Goodwin, J. C. in The S-100 and Other Micro Buses, Howard W. Sams and Co., Inc., (1979), p. 19.

26. National Semiconductor's Linear Databook, National Semiconductor Corp., Santa Clara, (1980), p. 8.45.

27. Underwood, R. K., in Linear Applications, Vol. 1, Vander Kooi, M. K., ed., National Semiconductor Corp., Santa Clara, (1972), p. 63-5.

28. National Semiconductor's Linear Databook, National Semiconductor Corp., Santa Clara, (1980), p. 8.114.

29. National Semiconductor's Lineal Databook, National Semiconductor Corp., Santa Clara, (1980), p. 8.98.

30. O'Conner, P. J., in Digital and Microprocessor Technology, Prentice-Hall, Inc., Englewood Cliffs, (1983), p. 541.