DRL No.
DRD No. SE-2

DOE/JPL/957031--85/1
Distribution Cat. UC-63

DOE/JPL/957031--85/1

DE85 007127

QUARTERLY TECHNICAL REPORT
for the period
10/1/84 through 12/30/84

INK JET PRINTING OF SILVER METALLIZATION FOR PHOTOVOLTAICS

Purdue Research Foundation

Principal Investigator:

R.W. Vest (317/494-7009)

JPL Flat Plate Solar Array Project

January 25, 1985

"The JPL Flat Plate Solar Array Project is sponsored by the U.S. Department of Energy and forms part of the Solar Photovoltaic Conversion Program to initiate a major effort toward the development of flat plate solar arrays. This work was performed by agreement between NASA and DOE."

FOREWORD

The research described in this report represents the effort for the first three months on Contract No. 957031 with the Jet Propulsion Laboratory, Pasadena, CA, under the technical cognizance of Paul Alexander. The research was conducted in the Turner Laboratory for Electroceramics, School of Materials Engineering and School of Electrical Engineering, Purdue University, W. Lafayette, IN under the direction of R.W. Vest. The research was carried out by Dr. S. Singaram, D.A. Binford and K.F. Teng.

TABLE OF CONTENTS

1. INTRODUCTION AND SUMMARY

During this quarter, significant progress was made in the continuing development of the ink jet printing system for thick film circuits. The unit being used in this research is a prototype ink jet printer developed on a contract with the Naval Avionics Center. One of the first tasks completed early in the quarter was the complete documentation of this ink jet printing system as it existed. It was determined that this was an essential step in deciding what modifications were needed to the system and how these modifications would be implemented. This printing system documentation has been included as Appendix A to this report, and will be referred to for clarification of changes which have already been completed during this quarter. Figure 1 gives an overall view of the ink jet printer as it existed at the beginning of this contract and Fig. 2 shows more details of the spray heat and X-Y table.

After this initial step was completed, design modification studies were started for electronic, mechanical, and programming aspects of the system. These studies were completed at the end of the second month. The areas needing improvement were discussed and applicable changes decided upon. Some of these improvements were completed during this quarter and others have only been started. It should be noted that, although the general areas needing improvement have been identified and some changes decided upon, the exact details of how other changes will be implemented have not yet been decided. During the next months, these details will be discussed further and the modifications put in place accordingly.

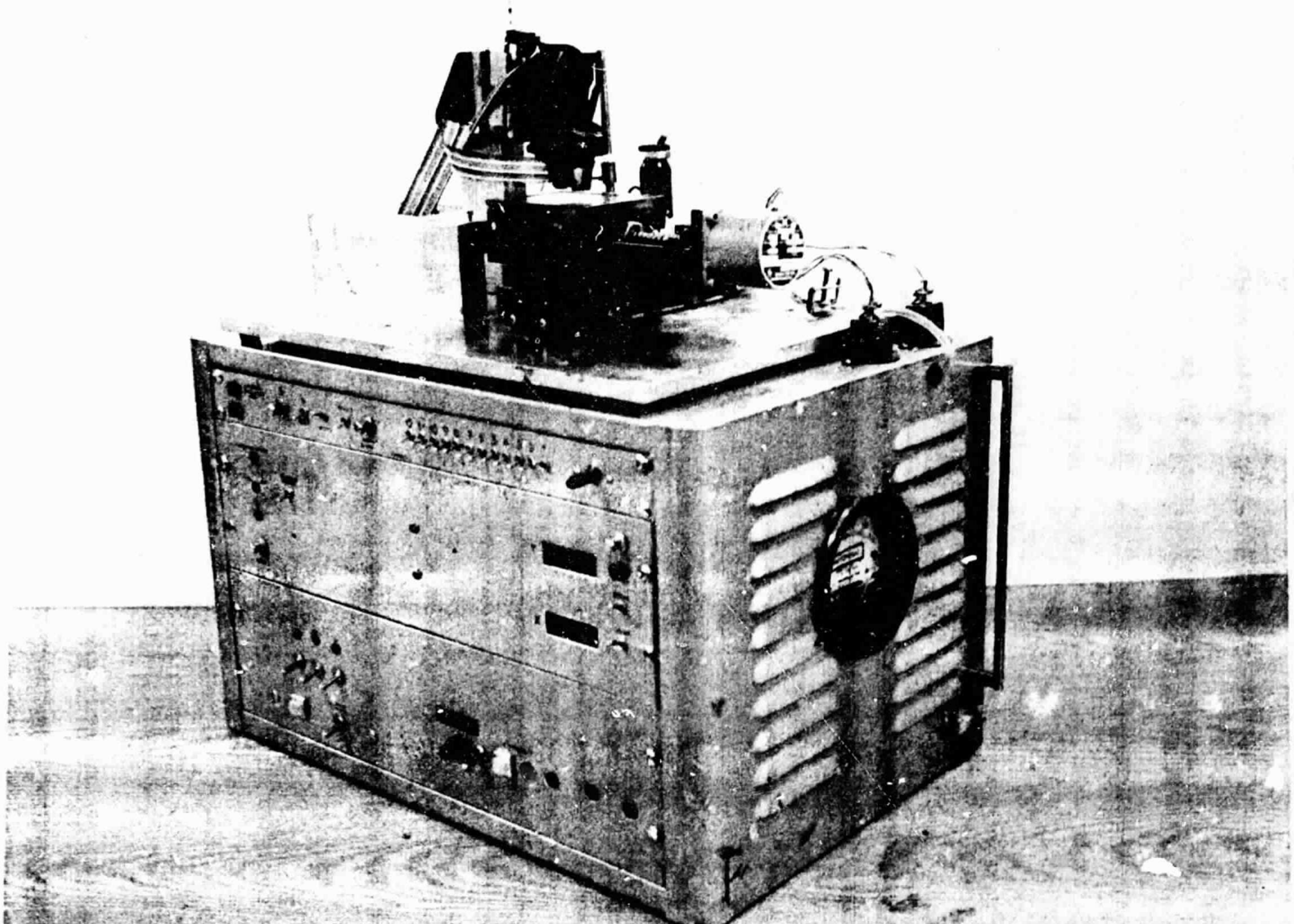


Figure 1. Turner Laboratory Ink Jet Printer.

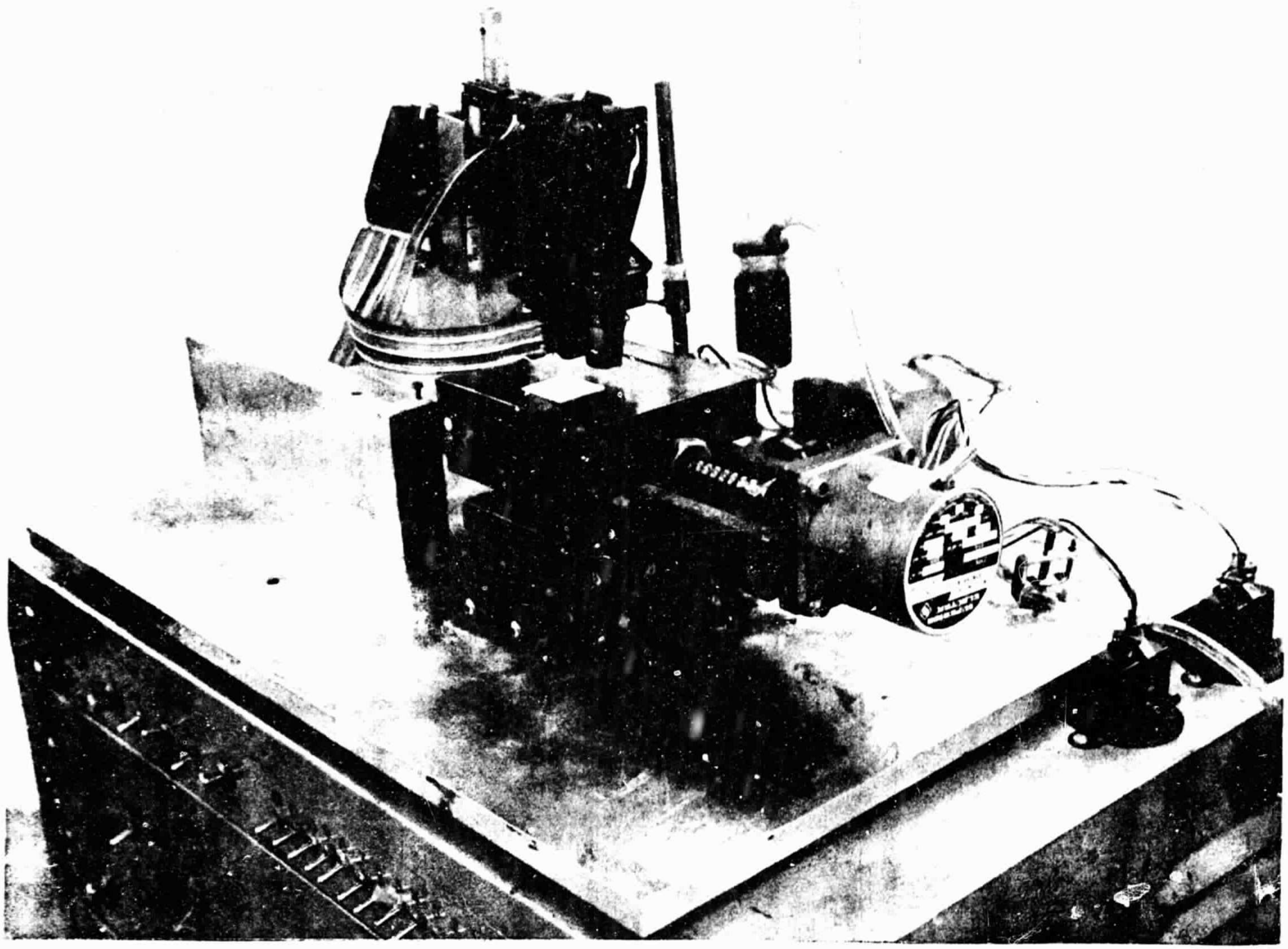


Figure 2. Spray Head and X-Y Table.

The first section of this report details the modifications which have already been made to the ink jet printing system. These include both mechanical and electronic and programming changes which were decided upon during the design modification studies. In some cases, these changes have been made only on a temporary basis until testing can verify that they work in the manner intended. As testing verifies their applicability, they will be implemented in a more permanent manner. Also, it should be pointed out that these circuit changes themselves may undergo additional modifications as total system packaging considerations and other factors dictate the need.

Other changes which will be made as a result of the design modification studies and those modifications which have been started but not completed are discussed in Section 3. These include not only the mechanical and electronic changes but, also, the programming modifications which often must be made along with the electronic changes and additions. Changes in software are always an ongoing process. For example, if greater print speeds are to be achieved then programming routines to control table acceleration and deceleration must be written to accomplish this. Some of the preliminary study to accomplish this was started early in the quarter and the actual programming is still in the process of being developed. Programming changes which have already been tested and implemented are included in the Section 2.2.

2. PROGRESS

2.1 Mechanical Modifications

The pressure control system for supplying the ink to the ink jet head assembly is a very critical element for the correct functioning of the ink jet system. This slight pressure (less than 0.25 inches of water) is used to offset the static vacuum present at the ink jet nozzles due to the ink supply being at a level below the nozzles. As additional testing is done to optimize the various parameters associated with the printing process, accuracy in controlling and monitoring this slight pressure will be essential. For this reason, a model 602-1 differential pressure transmitter was purchased from Dwyer Instruments, Inc. It has a minimum range of 0-0.4 inches of water. Power was supplied to the unit by connecting it to a transformer with a 20 volt secondary. A 500 ohm resistor was connected as a receiver for the transmitter and the voltage across the resistor monitored with a digital voltmeter. This assembly was calibrated for the 0 to 0.4 inches of water range which corresponds to 2.0 volts to 10.0 volts on the digital voltmeter.

As a part of this change, the entire pressure control and monitoring system was moved out of the original equipment cabinet (see Fig. 1) and temporarily mounted in a second cabinet to facilitate the changes. The new Dwyer pressure transmitter was connected to the 19 ml glass bottle which contains the MOD ink being printed. This connection was made temporarily by using the purge line as shown on Figure A11 of Appendix A. This change will permit more accurate evaluation of the existing pressure control system. As additional printing studies are done, other improvements in the ink supply and pressure control system may have to be made.

Other mechanical modifications to the existing ink jet printing system have been completed. It was decided that a more rigid mounting bracket for the ink jet spray head was needed so that more accurate and repeatable printings could be made. An aluminum mount was machined and attached to the aluminum plate which the positioning table is mounted on. Along with this change, a portion of the Siemens ink jet head not necessary in this application was cut away to make its mounting simpler. The head is attached to a precision adjustment mechanism so that the head to substrate spacing can be changed and this mechanism is attached to the new aluminum mount.

For many of the same reasons, a new brass block was machined for the positioning table. The inner region of this block has a recessed section where the substrate is placed. The block is machined in such a way that the substrate can be positioned in only one corner of the recessed section. The new block also contains both a vacuum chuck and the original 40 watt cylindrical heating element to keep the block at a temperature between 30 and 35^{o}C.

Finally, two micro switches were mounted on the positioning table so that the table could be positioned to an initial start point prior to each print sequence. These switches are ´debounced´ before going to the microcomputer board. The actual initialization is accomplished in the microcomputer programming. This programming and the actual connections into the SCCS-85 board will be described in the next section.

2.2 Electronic Modifications and Programming

In the existing ink jet printing system, the trigger signal for the pulse driver board circuits either came from a manual trigger circuit or from an external square wave generator (see Fig. A3 of Appendix A). This method did not allow the frequency of the trigger signal to be microprocessor controlled. As additional work is done to determine the optimum ratio between the trigger frequency and the X-Y table stepping frequency, it will be necessary to control these two frequencies carefully. The table stepping frequency is already controlled by the microcomputer board.

To eliminate the external generator and control the trigger rate from the microprocessor, one of the three programmable timers which are part of the SCCS-85 microcomputer board were utilized. Since these timers are referenced to the on-board crystal controlled clock, the frequency of the trigger signal for the ink jet nozzles could be assured. The timer output selected was OUT 2. It exits the SCCS-85 board at J3 pin 2. This signal was applied directly to the normally open contacts of the twelve ink jet head switches which are tied together. The switches still function as before and route the signal on to the selected nozzle channel of the pulse driver board. The two 'AND' gates located on the inverter board which were used to gate the trigger signal on and off were eliminated (see Fig. A3 of Appendix A). The SCCS-85 output port line PC 3 which had controlled these gates was routed to the programmable timer input GATE 2 after the on-board jumper connecting GATE 2 to +5 volts was removed. When GATE 2 is high the OUT 2 signal is enabled. GATE 2's status (either low or high) is controlled through programming. It is taken high only when the table is moving and printing is desired.

In order to utilize the programmable timer which is on the SCCS-85 board, a few changes to the main program were necessary. These changes were really just a simple initialization sequence for the timer. The timer mode was established and the frequency of its output signal set. Its mode was set so that it outputs a continuous square wave signal when enabled by the gate. Since the positioning table is currently being used at a base speed only of 400 steps per second, the frequency of the square wave signal from OUT 2 was set at 100 Hz. This frequency can, of course, be easily changed but the one to four ratio between nozzle frequency and table stepping frequency has in the past proven to be a good general rule for obtaining smooth, continuous printed lines. This relationship is something which will be studied further.

Another problem encountered in the past has been the inability to establish an exact starting point for printing on a substrate. The manual 'joystick' controller could be used to position the X-Y table to a general start point under the ink jet head prior to a printing sequence, but it was impossible to go to an exact point time after time. It was decided that the table could be positioned to a start point or origin by using some kind of initialization routine prior to any print sequence. It order to accomplish this some hardware additions were necessary, along with a few programming changes and additions to the main program.

As stated in the previous section, two precision switches having reasonably low differential travel specifications were mounted on the positioning table. These single pole double throw devices were mounted in such a way that their common terminals switched from the normally closed contacts to the normally open contacts when the table was positioned at the desired origin. They have fine adjustment mechanisms so that an exact initial starting point can be set. In order to be interfaced with the SCCS-85 microcomputer board, the switches were first 'debounced' using a circuit similar to the circuit for 'debouncing' the MANUAL TRIGGER switch shown on Fig. A3 of Appendix A. These circuits were built temporarily on the existing inverter board. The output line from the X-axis switch circuit was connected to the RST 5.5 interrupt input on the SCCS-85 board and the line from the Y-axis switch circuit was connected to the RST 6.5 interrupt input on the SCCS-85 board. These two interrupt inputs now make a low to high transition when the table gets to the origin for each respective axis. This completed the necessary hardware additions.

The programming which had to be added to complete this initialization process made use of the Superior Electric indexer board JOG command in order to move the table to the desired start position. The RST 5.5 and RST 6.5 input lines on the SCCS-85 microcomputer board are system interrupts. These two interrupts first had to be enabled and unmasked. The routine to accomplish initialization gives a JOG + command to the X-axis indexer board in an endless loop, moving the table along the X-axis toward the precision switch. When the switch is activated and the RST 5.5 interrupt line goes high, the endless JOG + loop is broken and the X-axis motor is switched off. The routine called by the interrupt also immediately masks the 5.5 interrupt so that additional switch activations would be ignored. The RST 5.5 interrupt remains masked but the RST 6.5 interrupt must again be enabled because the entire interrupt system is disabled any time any interrupt is received. Similarly, a JOG + command is given repetitively to the Y-axis indexer board, moving the table along the Y-axis until the switch is activated. The RST 6.5 input goes high generating another system interrupt. This switches off the Y-axis stepping motor and, as did the 5.5, masks itself so that additional transitions of the RST 6.5 line are ignored. The positioning table is now set to an origin from which it will begin a print sequence. After receiving the last interrupt, the microprocessor proceeds with the main program.

One other electronic modification was completed. There were several features built into the Siemens driver board which could not be used in this application. By eliminating this unneeded circuitry, the power requirements for the system could be reduced and the system further simplified. For these reasons, circuit traces on the board were cut and other modifications made so that power is applied only to the twelve pulse driver channels. The heater, temperature sense, wiper motor, and ink level sense lines to the ink jet head were removed, leaving only thirteen necessary lines to the ink head. Along with this change, the PRINTER READY l.e.d., the LOW INK l.e.d., and the MOTOR switch were discarded. With this change in place, the +5 volt supply is no longer needed on the driver board.

One last modification was made to the Siemens driver board. The original single turn potentiometers R1-R12, which control the amplitude of the output pulses from the 12 driver channels, were replaced with 15 turn potentiometers of the same 5000 ohm value so that more accurate control of the pulse amplitude could be accomplished.

2.3 Ink Development

Lots of silver neodecanoate and bismuth 2-ethylhexanoate were synthesized for use in the first test ink. Further ink development studies must await completion of the initial mechanical and electronic design modifications.

3. PLANS

As described in the preceeding sections, the original prototype ink jet printing system has already undergone several changes. Many other design modifications will be made in the coming months. Most of these will be implemented in order to reduce the number of connections and minimize interwiring (hence improving reliability), improve serviceability, improve system printing accuracy and repeatability, and generally to meet future system goals. Many of the system improvements have been decided upon as a result of the design modification studies. Other changes, however, will be put in place as experience with the system and future plans dictate. Some of these design changes have already been partially implemented or, in some cases, put in place on somewhat of a temporary basis for testing. Other modifications, particularly software modifications, will require some further investigation and will be made over many months.

During this quarter, some preliminary investigations were initiated to determine how a computer aided design routine might be integrated into the ink jet printing system. The final goal would be to be able to go directly from a circuit design created on a display terminal to the substrate. This would make the ink jet printing system a very valuable tool for circuit prototyping. It was found that there are several programs available within Purdue's Engineering Computer Network for laying out circuits on graphic terminals. Some of these programs may require additional equipment not currently available. This aspect will be looked into further. In any case, programs will have to be written so that the microcomputer board can receive and properly interpret the circuit information that would come from a graphics display terminal. This could require some basic restructuring of the entire main operating program. Work will continue in order to determine what additional equipment might be needed for this process and what additional programming will be required.

In the area of mechanical modifications, several jobs have either been started or are being planned. It has already been stated that other modifications may have to be made to the ink pressure control and monitoring section after ink printing studies are started. However, along with the changes already made, one other aspect of the system will be changed in the very near future. Two new pinch valves have been ordered to replace the Sporlan solenoid valves currently being used in the pressure control system. In the past, there had been a periodic problem due to the electrical noise (or line transients) being generated by the valves switching on and off. The new valves operate on 12 volts direct current instead of 115 VAC and operate on much less power. This should alleviate the noise problem encountered in the past. They, also, will avoid the two tubing connections at each valve since they operate by pinching the tube.

Another mechanical aspect of the system which will be improved is in the area of general system packaging. Some preliminary layout work has already been done. This change will involve almost all components of the system. This is being done primarily to minimize connections and interwiring between components, to improve system cooling, to make components more easily accessible, and to generally improve system operation. This change will include redesigning the power supply and relocating it. Additionally, all the components included in the positioning table drive system will be consolidated onto one rack-mountable panel. This panel will also include the two displays indicating table position. The system packaging changes will be an ongoing process ,most of which should be completed in the next several months.

Several electronic modifications are in the planning stages. The pulses to the piezoceramic drivers in the ink jet head must be able to be controlled accurately. The addition of the higher quality potentiometers to the Siemens driver board was a step in this direction. However, the input pulse to the Siemens driver board is largely responsible for the shape of the output pulse so its pulse width must be precisely controlled. The pulse driver board (Fig. A4 of Appendix A) which is composed of twelve identical circuits is responsible for generating these pulses. These twelve circuits are 'one-shot' multivibrators which use a simple external resistor-capacitor circuit for controlling their output pulse width. The Siemens Corporation specifies that the pulses to trigger the Siemens driver board have a pulse width of 22.5 μs. A check of the existing circuitry indicated a variation of 20 to 26 μs. To more accurately control this pulse width, the present SN74121N multivibrator I.C.'s will be replaced by dual precision monostable multivibrator I.C.'s. This will, first of all, reduce the chip count from twelve to six since these units have two complete circuits per package. More importantly, these integrated circuits use linear CMOS techniques allowing more precise control of output pulse width. This in combination with the external 15 turn potentiometers for initial calibration will dramatically improve the accuracy of the outputted pulse.

As part of the new packaging for the positioning table drive components, the existing LED displays will be replaced by new liquid crystal displays in order to reduce overall system power requirements. A new up/down counter and LCD display driver integrated circuit has been ordered to provide the correct drive signals to the displays. They will connect to the indexer boards in very much the same way as the displays do currently. Special bezels will be used to mount the new displays into the eighth inch aluminum rack panel.

As a means of planning for the future when an attempt will be made to use all twelve ink jet nozzles for printing, circuitry is being added now so that the nozzles can be turned on and off via the microprocessor. This will be accomplished by adding two octal data latches to the same circuit board which contains the new CMOS multivibrators. Three control lines and the eight data buss lines from the SCCS-85 microcomputer board will control these latches. Twelve manual switches will be included so that the nozzles may still be turned on manually. Light emitting diodes will probably be incorporated to give a visual indication of which nozzles are on. It should be pointed out that before all twelve nozzles can be utilized for printing, much more program development will be necessary.

Program development continues on a regular basis. Even though programs have been written for the nozzle triggering system and for the table initialization process, these programs may require additional work in order that these systems work in the most efficient and concise manner. Other areas requiring programming changes are being investigated. A program addition may be made to pattern data in such a way that the ink jet print head is moved away from the substrate after the print is completed. This will facilitate the removal of the substrate.

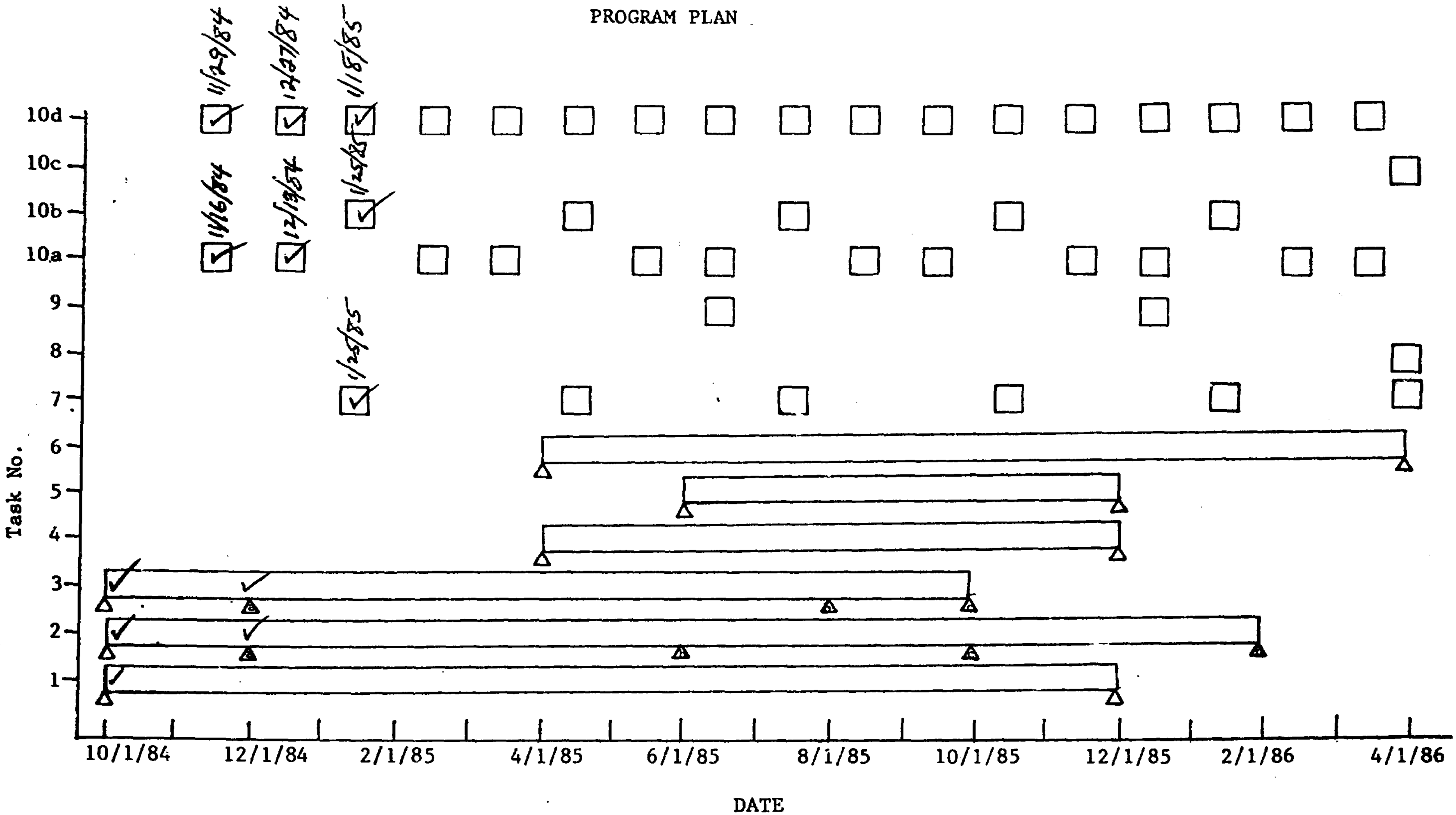
More work will be done in achieving the goal of much higher print speeds. This may involve a new degree of programming complexity up until now not needed. Currently the positioning table is being used at a base speed only. The Superior Electric indexer boards are capable of fairly high print speeds, but acceleration and deceleration must also be programmed into the system when higher speeds are used. The acceleration and deceleration parameters are entered into the indexer board in ASCII code as are other instructions. Nonetheless, a large amount of additional software development will be necessary before higher print speeds can be reached. Some preliminary testing routines are currently being written and work will continue through the next quarter to accomplish this goal.

4. SCHEDULE

The description of tasks and the updated milestone chart are attached.

LIST OF TASKS

1. Ink Development and Processing Studies
2. Electronic Modifications and Programming
   a. Design modification studies
   b. Electronic assembly
   c. Operational demonstration
   d. Computer software generation
3. Mechanical Modifications
   a. Design modification studies
   b. Mechanical assembly
   c. Operational demonstration
4. Film Thickness and Line Width Studies
5. Printing Speed Studies
6. Fabricating and Characterizing Cells
7. Specifications
8. Data for Economic Evaluation
9. Personnel for Meetings
10. Documentation
    a. Monthly technical reports
    b. Quarterly technical reports
    c. Final technical report
    d. Monthly financial reports



PROGRAM PLAN
11/29/84
12/27/84
1/18/85
11/16/84
12/13/84
1/25/85
1/25/85
10d
10c
10b
10a
9
8
7
6
5
4
3
2
1
Task No.
10/1/84
12/1/84
2/1/85
4/1/85
6/1/85
8/1/85
10/1/85
12/1/85
2/1/86
4/1/86
DATE

APPENDIX A

Documentation of the Turner Laboratory Ink Jet Printer

as of October 1, 1984

TABLE OF CONTENTS

1. INTRODUCTION

There are many inherent advantages to using ink jet printing techniques on hybrid microcircuits. First of all, this process can easily be completely computer controlled allowing a higher degree of automation than now possible with conventional screening techniques. This would, in turn, yield a potential cost savings, greater repeatability and reliability, and the abiltity to move rapidly from initial design stages to circuit prototypes. Another major advantage of the ink jet printing process is better uniformity of the thickness of the deposited films since surface topography is no longer a factor in influencing film thickness. If circuit performance can be improved by varying the thickness of the films in various regions of the same circuit then this technique will allow one to accomplish that with relative ease.

There are many potential advantages to ink jet printing, but designing and implementing a workable system requires overcoming some significant problems. First of all , it should be pointed out that consideration of using this technique of printing was made possible by the development of metallo organic decomposition (MOD) inks since this process dictates the use of inks which do not contain particulates. However, for these MOD inks to be used with an ink jet spray head their viscosities had to be much lower than that required for screening and their surface tension was a much more critical parameter in this application. Additionally, the ink jet spray heads presently available were designed primarily for the printing of alphanumeric characters, not for printing the continuous, uniform patterns required for most hybrid microcircuits. This meant that a commercially available head had to be modified. Other problems also had to be dealt with such as designing an ink supply system for the head which would provide the neccesary meniscus at the ink jet nozzles and also allow the MOD inks to be contained in an inert environment. Another complex task was coordinating the pulsing of the ink jet nozzles with the movement of an X-Y table directly below the ink jet head in order to print the required pattern on a substrate. Some progress has been made in this area but additional work will have to be done, particularly as an attempt is made to increase the printing speed. Some of the mentioned problems have now been resolved, but others will have to be investigated further and solutions found.

The existing system is described in this Appendix, including complete circuit diagrams and explanations, software documentation, and general operational aspects. The system has been described section by section with most sections prefixed by a general overview of that portion of the system. There is also a discussion of a few of the problems that require additional investigation and study.

2. INK JET PRINTING SYSTEM OVERVIEW

The ink jet printing system, although having limitations in its current state of development, is capable of printing well defined patterns onto substrates using MOD inks. The ink jet printer used in this study was a Siemens Pt80i head which has 12 nozzles (76 μm diameter) arranged in two staggered rows of six. Each nozzle has its own piezoelectric driver making this head a drop-on-demand type. Ink is supplied to the head under slight static vacuum so that flow is resisted by the surface tension of the meniscus at the nozzle. A droplet is ejected by means of a pressure wave generated by an impulse from the piezoelectric driver. Substrates were mounted on an X-Y table which moved in 25 μm steps in response to input pulses. The desired patterns were generated by programming the motion of the table and the firing of the jets. So far the system has been used only at reasonably slow print rates (substrate velocities between .00254 and .01016 meters per second) although higher rates are possible. In its simplest form, the system consists of two major blocks. The first section contains all the necessary components to fire and control the ink jet nozzles, and the other section is responsible for the movement of the X-Y positioning table below those nozzles. In actuality, of course, the system is much more complex. A SCCS-85 microcomputer board has been integrated into the system. Its function is basically to take operational and circuit pattern data given to it via its RS-232 input, process that data, and then provide the proper signals to the two above mentioned sections so that they will work together in a manner which will print the desired pattern on the substrate.

All the major sections of the system are indicated in Fig. A1. A manual triggering circuit was provided as a means of manually firing the ink jet nozzles for initial testing purposes. Most of the testing was done using an external oscillator as the triggering source. Also, a joystick control was included in the system for manually controlling the table even though most of the experimental work was done with the table in an automatic control mode. Two four digit l.e.d. displays were

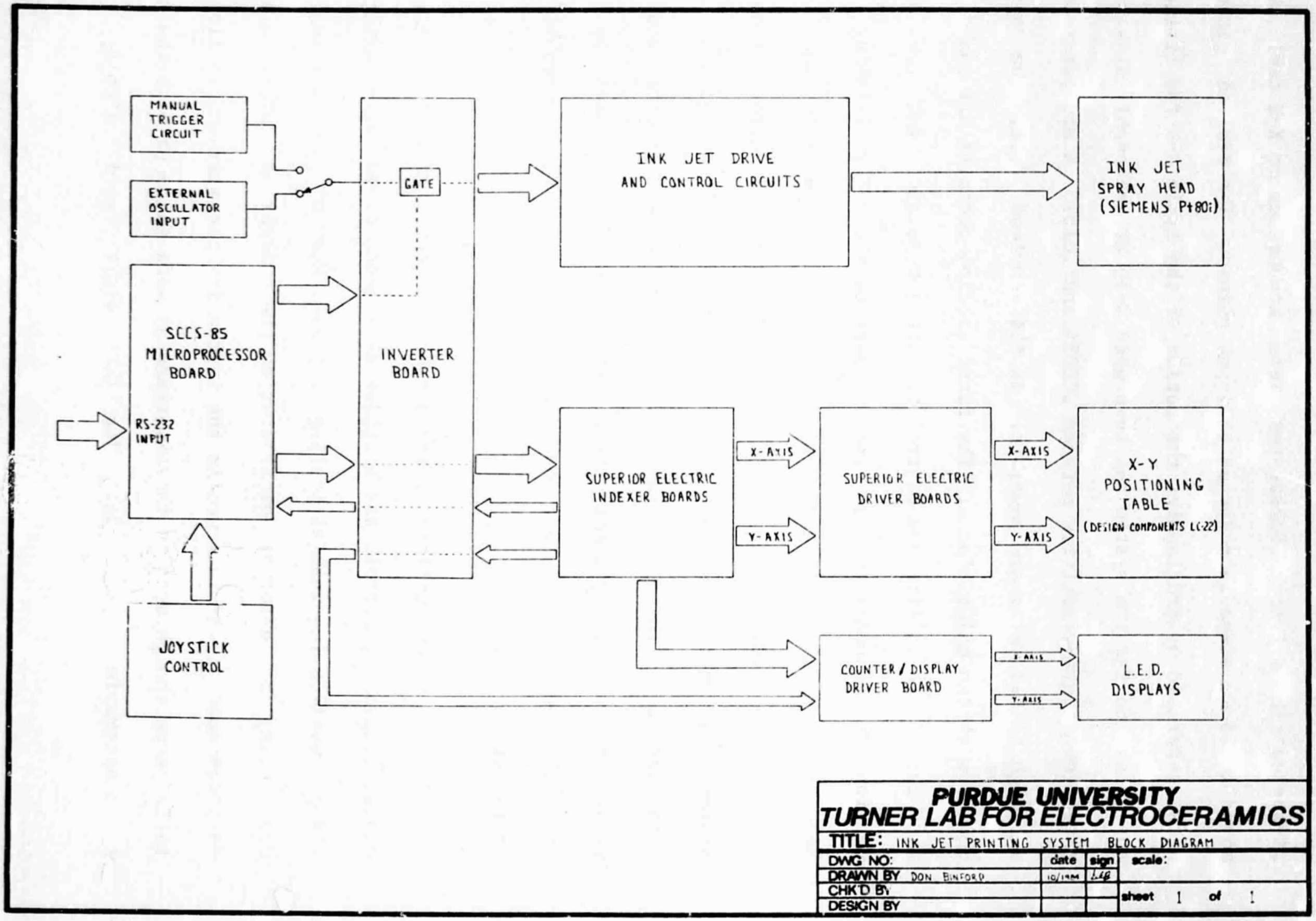


Figure A.1 Ink Jet Printing System Block Diagram.

incorporated into the system to give a visual indication of table position. Each unit represents one motor step or about 25 µm. Direction and count information for the X and Y channels of the counter/display driver board comes from the X-axis or Y-axis Superior Electric indexer boards respectively. All of these sections mentioned so far, although important, are not an integral part of the main operating system.

At the heart of the operating system is the SCCS-85 microcomputer board. The main operating program and pattern data are downloaded into the microprocessor board via the RS-232 input from a host computer system. The SCCS-85 then provides signals to the ink jet drive section and the X-Y table positioning section. In the first case, the SCCS-85 provides only a single control line which enables a gate on the inverter board. This allows the external oscillator triggering signals to pass on to the ink jet drive and control circuits. There, one or more tracks of the ink jet spray head are selected and the triggering signals then routed on to the appropriate drive channels. The drive electronics provide the necessary pulses to fire the corresponding piezoelectric elements in the Siemens head. Thus the droplets of ink are ejected onto the substrate. In the case of the table control section, the SCCS-85 provides 10 lines of information, via the inverter board, to the two Superior Electric indexer boards. The 8-bit parallel data bus goes to both boards, then there is a single enable line to each board. Additionally, there are two lines back to the SCCS-85 (again via the inverter board) from the indexer boards to provide proper sequencing of signals. Output signals from the X-axis and Y-axis indexer boards provide the necessary information to the respective X and Y axis Superior Electric driver boards. These then, in turn, provide the drive signals to the Design Components positioning table. It is also the indexer boards which provide the position information to the l.e.d. displays.

With all these sections working together correctly, a preprogrammed pattern can be printed onto a substrate. The actual operation of the system to accomplish this is fairly simple. A 10 cm square brass block which is mounted on the X-Y table is preheated to somewhere between 30 and 35°C by a heating element mounted in the block and controlled by a variable transformer. The substrate is positioned in a recessed inner region of the block and held in place by a vacuum applied to a hole beneath the substrate. The table is manually positioned to a predetermined start point in respect to the ink jet head. The MOD ink is put in the ink supply system and the supply system checked for proper operation. An oscillator is connected to the external oscillator input and adjusted for the correct triggering signal. All power supply switches are turned on, the MAN/EXT switch is set to EXT, the ENABLE and MOTOR switches are turned on, one of the twelve ink jet track switches is turned on, and the SCCS-85 is reset. The microprocessor is instructed to load and the main operational program is downloaded from the host computer. The micro is again instructed to load and the pattern program is downloaded. A final command to the microprocessor starts the program running and the circuit pattern is printed onto the substrate. As the program runs the l.e.d. displays track the table movement. Once the printing stops, the vacuum is switched off and the substrate removed and fired as necessary.

It is hoped that the above description has provided the reader with a general understanding of the ink jet printing system that was developed in the Turner Laboratory. The sections which follow describe in more detail the major blocks of the system. There is also a description and listing of the software for the SCCS-85 board.

3. SCCS-85 MICROPROCESSOR SECTION

3.1 SCCS-85 Microcomputer Board

The SCCS-85 is a very versatile Intel 8085-based microcomputer system contained on a 11.43 cm X 17.78 cm board. Its designed-in flexibility allows it to be used in a wide variety of control applications. With no modifications at all, it is configured to operate as a small computer communicating via RS-232 with a user supplied terminal. Up to four kilobytes of RAM (random access memory) may be installed on the board itself and a memory capacity of up to 65K bytes is possible by extending the SCCS-85 bus to additional cards.

The SCCS-85 circuit board is a unit designed by Robert Rindfuss that was purchased locally. It is revision 2 of the original circuit board design. The integrated circuits, I.C. sockets, connectors, and other miscellaneous electronic components were purchased from various suppliers and mounted on the circuit board according to its included instructions. Specific modifications must be made to the circuit board according to the user's individual needs. Most of these are accomplished by cutting circuit board traces and/ or jumpering pins or specified feed through terminals.

This microcomputer board is divided into seven functional groups. They are the following:

1. CPU (central processing unit) group
2. ROM (read only memory) group
3. RAM (random access memory) group
4. SERIAL IO (input/output) group
5. PARALLEL IO group
6. TIMER group
7. DMA (direct memory access) group

The CPU, RAM, and ROM groups are required for the operation of any system, however the remaining groups are optional and need only be present on the board if the application requires it. For the ink jet printing system, all the above blocks were required except the DMA group. The two integrated circuits necessary for DMA were not purchased and are not present on the ink jet printing system's SCCS-85 board. In the case of the ROM group, a single NEC D2716 EPROM, giving 2K of ROM, is being used, leaving I.C. location U5 free for expansion to 4K as required. The board was reconfigured to accept the 2716 EPROM. The 2K of EPROM (2048 bytes) holds the control and monitor programs. The 4K of RAM short term memory holds graphic data downloaded from the host computer and also provides a scratch pad work space for the monitor and control programs. The SERIAL IO group contains the USART (universal synchronous/asynchronous receiver transmitter) through which all communication and data transfer takes place via a standard RS232 serial link. Outputs to the indexer boards and the ink jet head enable circuitry, and inputs for 'handshaking' lines are sent and received in the PARALLEL I/O group. The TIMER group is being used to provide the correct timing to the USART and, in the future, may be used to provide the triggering signals to the ink jet head circuitry.

A complete copy of the SCCS-85 (revision 2) User's Manual is included as Appendix A1. Included in it is both a component list for the board and also the instructions for mounting the components. Additionally, necessary modifications as required by the user are described in detail and complete circuit daigrams are provided. As a means of clarification, however, specific modifications and parts used in this application of the microcomputer board are listed below:

1. A 4 MHz crystal is used so the 8085's clock frequency is 2 MHz.

2. I.C. locations U5, U15, and U16 are not being used.

3. A single NEC D2716 EPROM chip is being used in the ROM group section. It is a single voltage (+5V) I.C. so the modification described on pages 12 and 13 of the User's Manual has been performed.

4. The TRAP interrupt input (pin 6 of the 8085) is being used for enabling the 'joystick'. On the SCCS-85 the TRAP input is normally pulled low so to use this feature the trace between P3-1 and P3-2 (CPU schematic, User's Manual) had to be cut.

5. For proper operation of the RS232 serial data input the following modification was made done (refer to the Serial Group schematic, User's Manual). The trace between C and D was cut so that pin 1 of the 1489 I.C. at location U24 is no longer pulled up to +12 volts.

The connection between pin 3 of the 1489 and pin 22 of the 8251 was broken and pin 3 and pin 22 of the 8251 was jumpered.

6. Two connectors were installed on the circuit board at locations J2 and J6a. Each of these is a 26-pin double row header made by A P Products, Inc. (part # AP 923863-R).

3.2 Downloading from Host Computer

As previously stated, the control and monitor program for the 8085 is contained in the 2716 EPROM on the SCCS-85 circuit board. However, the main operating program and any pattern programs must be downloaded into the SCCS-85 board from a host computer. This is accomplished through the RS232 serial data input which was incorporated into the design of the board. There is a standard 25-pin D connector mounted in the equipment cabinet and it is wired in the following manner to the SCCS-85 board:

Ground	J6a-pin 1
Transmit	J6a-pin 3
Receive	J6a-pin 5

A switch box assembly is used to provide all necessary switching between the host computer, the SCCS-85 board, and a terminal. This makes it possible for the terminal to communicate with either the host computer or with the SCCS-85 board. It also allows the host computer to transmit information directly to the SCCS-85 board, i.e. it allows 'downloading' from the host to the microprocessor. This information is stored in the RAM group on the SCCS-85 board.

3.3 Joystick Control

In initial design stages, the X-Y table was only controllable through the use of a manually operated 'joystick'. In its current state of development, the ink jet printing system uses the SCCS-85 microcomputer board to control the positioning table. After this system was implemented, it was still felt that some means of manually positioning the table should be provided so that the table could be positioned to the required start position prior to a print run.

For this reason, a program was written and made a part of the main operating program for the system. Pressing the JOYSTICK ENABLE button takes the TRAP interrupt on the SCCS-85 board momentarily high, vectoring the microprocessor to the joystick program. Once this has been done, the joystick control operates the table in the manner one would expect with this exception. Only movement parallel to the X or Y axis is possible. This, by the way, is also true in the computer controlled mode. All information for manual joystick control enters the Parallel I/O section of the SCCS-85 board via J2 pins 20, 22, 24, and 26.

4. INVERTER BOARD

The inverter board (Fig. A2) provides two very simple functions in the system. First of all and most obvious, it provides inversion of signals where necessary. There are fourteen inverters on the board, and twelve of these are used to reconfigure signals either coming from or going to the SCCS-85 board. The last two inverters provide inversion between the 'not' XLR-PU outputs on the X and Y axis Superior Electric

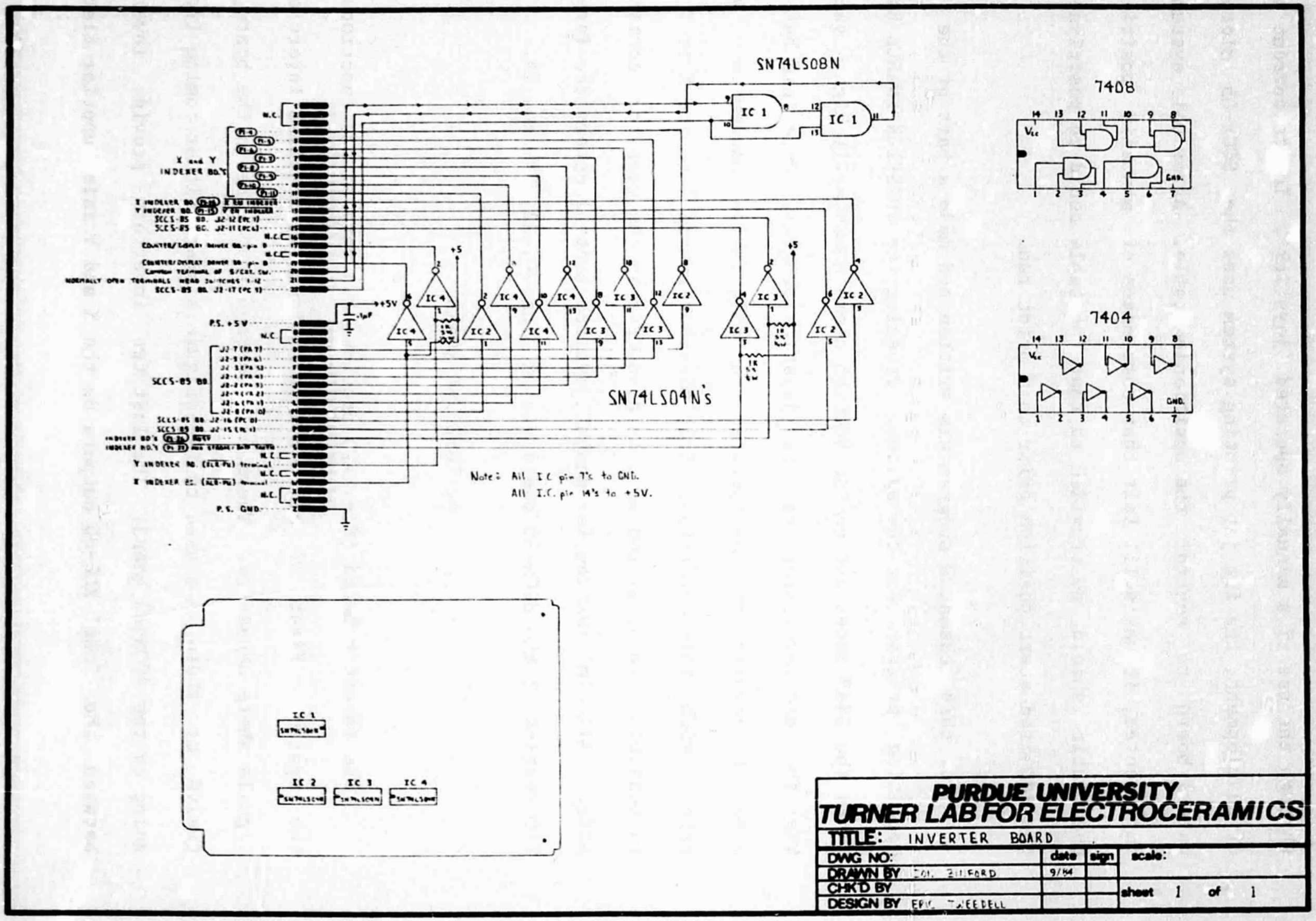


Figure A.2 Inverter Board.

indexer boards and the count inputs on the Counter/Display Driver board. Secondly, the inverter board contains a SN74LS08N quad-AND integrated circuit. Two sections of this IC are used to provide a gating function for the triggering signal for the ink jet drive and control section. This gate allows the SCCS-85 to enable this trigger.

The inverter board is a 'Vector' #3677-2 circuit board (11.4 cm X 16.5 cm) with a 22-pin edge connector. Mounted on the board is one SN74LS08N quad-AND IC and three SN74LS04N hex inverter IC's. Referring to Fig. A2, it is shown that the power supply ground enters the board on pin Z while the +5 volt line is applied at pin A. Pins 20-22 are the input, the output, and the enable lines for the gate IC 1. Two AND gates are used in order to provide proper isolation between input and output. The trigger signal from the trigger select switch MAN/EXT goes to pin 9 of IC 1. If pins 10 and 13 of IC 1 are high due to an enable signal from the SCCS-85 board, then the trigger signal passes on through from pin 9 to pin 8. Since pin 13 is also high, the signal continues from pin 12 to pin 11 and exits the board at pin 21. Eight inverters are used to invert the data line from the SCCS-85 board to the two indexer boards. The output port (PA 0-7) signals enter the board at pins M-D respectively and exit the board at pins 11-4 respectively. Pins 4-11 connect to P1-4 through P1-11 on the Superior Electric indexer boards. Two other signals from PC 0 and PC 1 on the SCCS-85 board are routed to pins N and P. They are inverted and exit at pins 12 and 13 to continue on to P1-16 of the X-axis indexer board and to P1-15 of the Y-axis indexer board respectively. P1-26 ('not' BUSY) lines and P1-37 ('not' AUX STROBE/DATA TAKEN) lines from both indexer boards enter the inverter board at pins R and S. These inverted signals exit at pins 14 and 15 and return information to the SCCS-85 board (PC 7 and PC 6). These two inverters have 1000 ohm pull-up resisters on their inputs. Finally, the 'not' XLR-PU signals from the two indexer boards are inputted at pins U and W and exit at pins 17 and 19 to provide count information to the corresponding channel of the Counter/Display Driver board. These two inverters also have 1000 ohm resistors from their inputs up to the +5 volt supply.

Even though this board is very simple in its function, its input and output pin number designations can be confusing as described above. For that reason, it is suggested that Figs. A5 and A6 be referred to for clarifying signal flow in and out of the board. On these two figures all fourteen inverters are shown. Figure A3 may also be referred to clarify the function of the gate circuit on this board.

5. INK JET DRIVE SECTION

5.1 Overview

The ink jet driver section can be summarized very simply. Figure A3 explains the flow from the triggering source to the Siemens ink jet spray head. The triggering signal, which must be enabled by the SCCS-85 microcomputer board, leaves the inverter board and is routed on to the ink jet head switch assembly. There it is routed to one or more channels of the Pulse Driver board. It should be noted here that most of the testing that has been done was done using only one ink jet nozzle at a time. The Pulse Driver board provides triggering signals of specific

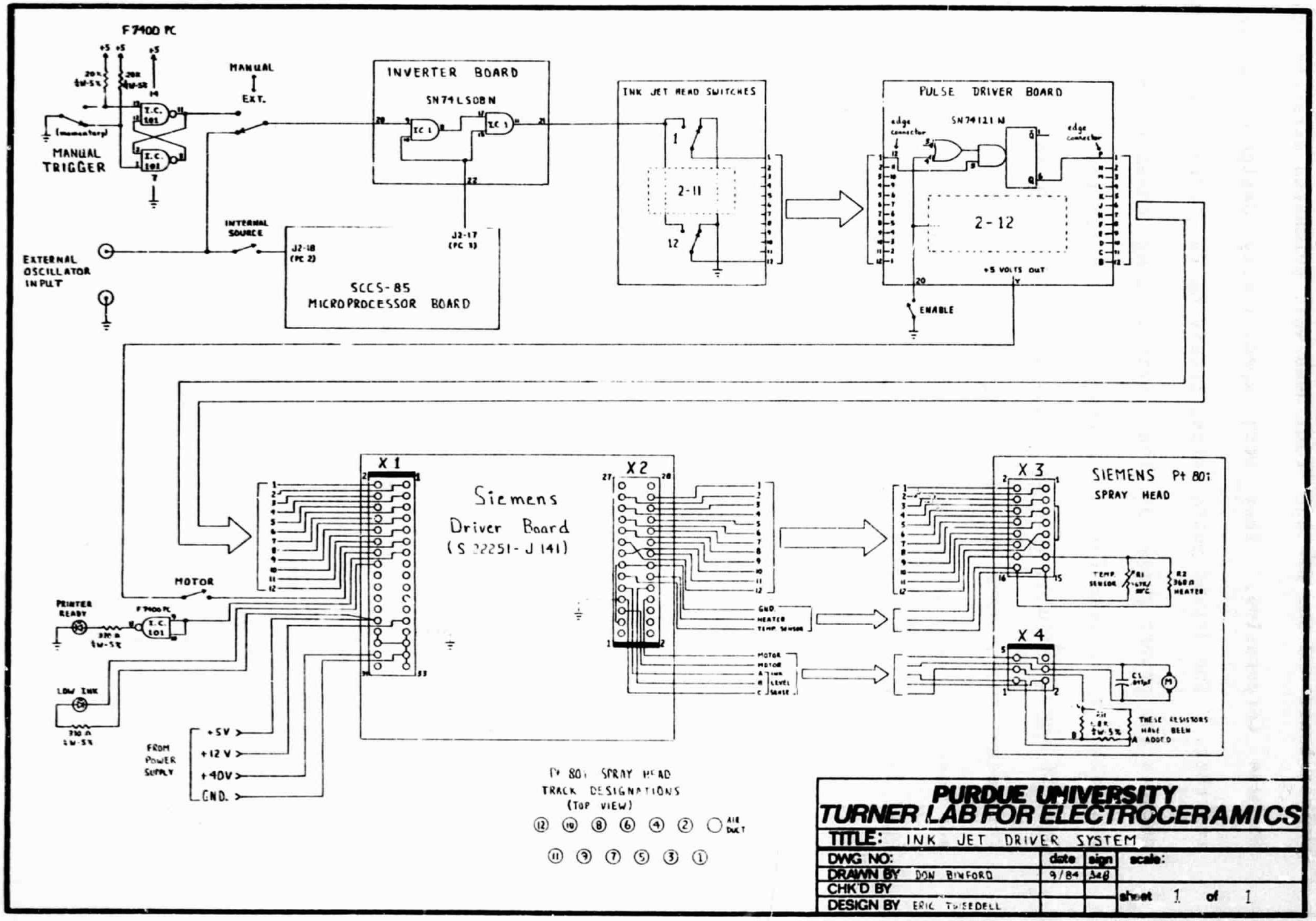


Figure A.3 Ink Jet Driver System.

amplitude and pulse width to the Siemens driver board. The Siemens driver board and the Siemens Pt80i head were purchased directly from the Siemens Corporation. They were specifically designed to be used together. The driver board, after receiving the correct trigger pulse, outputs the proper pulse on the corresponding channel to fire the piezoelectric driver for the corresponding ink jet nozzle. Thus a droplet of ink is ejected onto the substrate. This process continues at a rate equal to the rate of the trigger pulses as long as the microprocessor keeps the gate on the inverter board enabled. Hence, assuming that the X-Y table is moving, a series of ink droplets (forming a line) are printed onto the substrate.

5.2 Triggering and Control Circuits

Triggering is possible from either of two sources. IC 101 (F7400PC) is a quad 2-input NAND gate. Two of these gates have been connected as indicated in Figure A3 to form a 'bounceless' switch. The MANUAL TRIGGER switch is a momentary single-pole double-throw device. Pin 11 of IC 1 is normally low. It goes high at the instant the normally open terminal of the MANUAL TRIGGER switch is taken to ground and stays high until the normally open terminal of the switch goes high again (when the button is released). It is the positive-going portion of this signal which is the actual triggering mechanism. Triggering may also come from an external source connected to front panel banana jacks. A Hewlett Packard model 3310A function generator was used for this purpose. Several different waveforms and amplitudes were investigated but it was found that a squarewave with an amplitude of about 5 volts provided proper triggering. The frequency of the external oscillator had to be adjusted according to the velocity of the positioning table since the relationship of these two variables (and actually several others) affect many characteristics of the line printed. Most tests were conducted using table speeds of .00254, .00508, or .01016 meters per second. These speeds correspond to motor step frequencies of 100, 200, or 400 steps per second respectively. Some experimental work was done concerning the relationship between the trigger frequency and the table velocity. A formula relating the two was found which seemed to produce, in most cases, smooth line patterns on the substrates assuming that a median nozzle to substrate spacing of about 300 μm was maintained. If the table velocity (expressed in meters per second) is divided by .0001 meter per cycle then the value obtained in hertz (cycles per second) is the oscillator frequency which will produce the desired effect. Put more simply, the table motor step frequency divided by four will yield the required oscillator frequency. The MANUAL/EXT trigger select switch was provided to select between the two above trigger sources.

The trigger leaves the common terminal of the MANUAL/EXT switch and flows through the gate on the inverter board as described in the inverter section. From pin 21 of the inverter board the signal travels on to the normally open terminals of the twelve ink jet head switches. These twelve single-pole double-throw switches either ground the corresponding input on the Pulse Driver board or route the trigger signal on to that channel. These switch numbers indicate the track number on the ink jet spray head as indicated in the lower portion of Fig. A3

The INTERNAL SOURCE switch is not presently being used but has been provided for future development. Eventually, it is hoped that the necessary triggering signals will come from the SCCS-85 board. Additional software development is necessary to provide this feature.

5.3 Pulse Driver Board

The pulse driver board's function is to provide the Siemens driver board with triggering pulses of definite pulse duration and amplitude. It assures that the pulses triggering the Siemens board are independent of the amplitude and duration of the triggering pulses coming either from the 'bounceless' switch or the external oscillator.

This board, shown schematically in Fig. A4, is a 12 channel unit built on a 'Vector' #3677-2 circuit board (11.4 cm X 16.5 cm) with a 22-pin edge connector. It is comprised of 12 SN74121N integrated circuits which are monostable multivibrators in 14 pin dual in-line packages. The power supply ground is at pin A and +15 unregulated volts come in at pin Z. There is 7805 +5 volt regulator on the board to provide the necessary supply voltage to the IC's. There is a +5 volt line leaving the board at pin Y which supplies +5 volts to one side of the MOTOR switch. This connection is indicated in Fig. A3. A switched ground (from the front panel ENABLE switch) enters the board at pin 20 to enable the 12 multivibrators. The inputs to the board (from the 12 ink jet head switches) are at pins 1-12 but in an opposite order in terms of track numbers. The pulses exit this board at pins P-B. The output lines from the IC's have been shielded as indicated in Fig. A4 to avoid 'crosstalk' and noise problems.

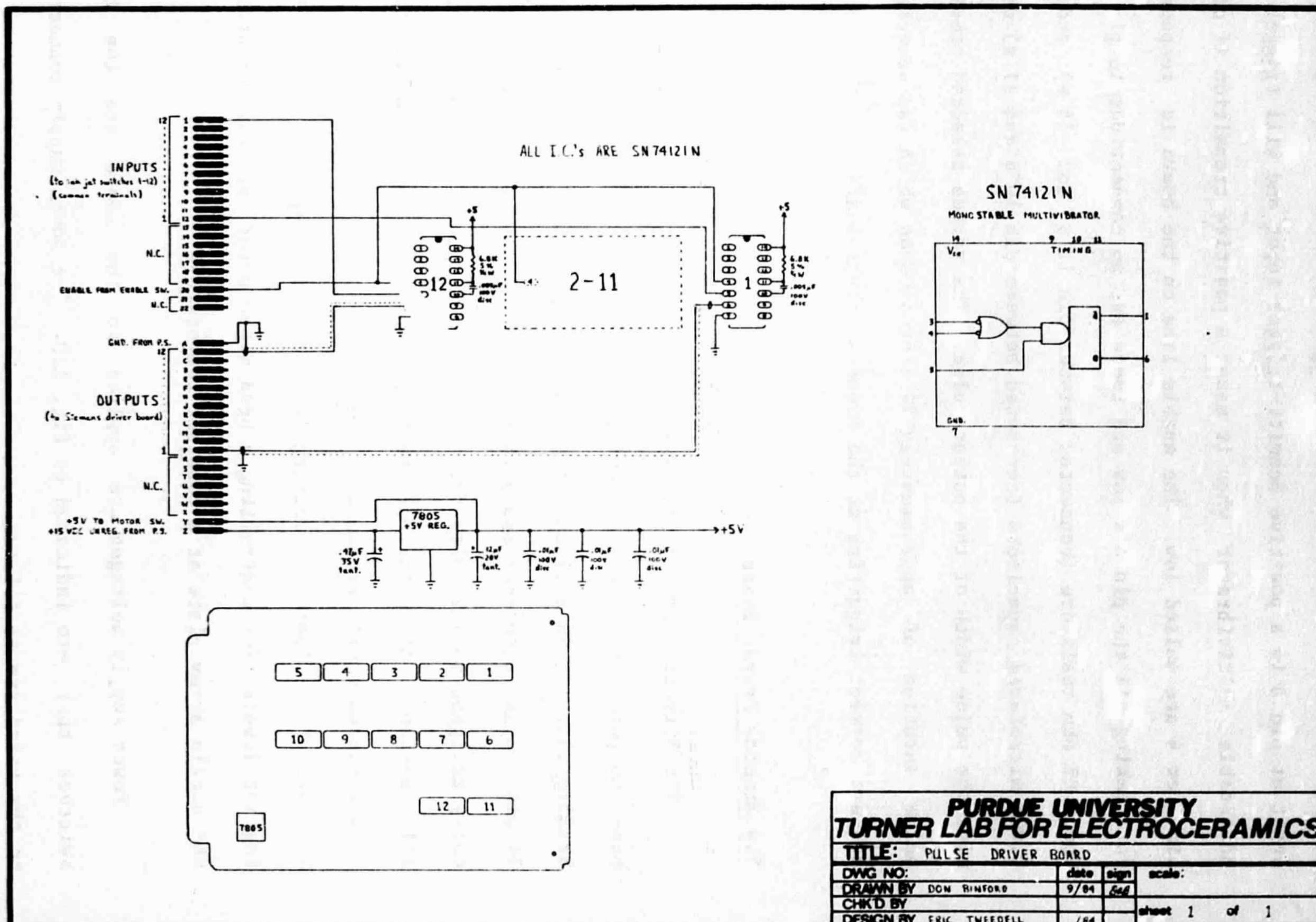


Figure A.4 Pulse Driver Board.

The SN74121N IC´s used on this board are monolithic TTL monostable multivibrators. The internal structure is indicated on Fig. A4. The input at pin 5 is a positive Schmitt-trigger input and will trigger the monostable multivibrator when it makes a positive transition if either pin 3 or 4 are pulled low. The enable line on the board is responsible for taking all the pin 4´s low and there are no connections to pin 3´s. The 6.8K ohm resistors (connected between pin 11´s and 14´s) and the 0.005 microfarad capacitors (connected between pin 10´s and 11´s) determine the pulse width of the output pulse. The values selected provide a pulse duration of approximately 24 microseconds which was selected to achieve correct triggering at the Siemens driver board.

5.4 Siemens Driver Board

The Siemens Driver board was purchased directly from the Siemens Corporation. Their identification number for the board is S22251-J141. It is a circuit board approximately 14 cm square with a 34-pin input connector and a 28-pin output connector. It was specifically designed to work with the Siemens Pt80i spray head that is used in this system. Its major purpose is providing the correct drive signals to the piezoelectric elements in the head. Other circuitry has been included on the board for driving led status display lines, for monitoring ink levels, for controlling a head wiper motor, and for maintaining the nozzle array plate at a constant temperature.

Power supply voltages are applied to the board via the three switches that are indicated on Fig. A10. The power supply connections to the board are as follows:

Ground	X1 pins 27-31 and 33
+5 volts	X1 pin 26
+12 volts	X1 pin 25
+40 volts unreg.	X1 pin 32

The inputs to the board are on pins 1-12 of X1 but note (Fig. A3) that track numbers do not correspond to pin numbers. The same holds true in the case of the outputs which are at X2 pins 15-26. Power supply ground is connected to X2 pins 3, 7, 13, and 14. This ground exits the board at X2 pin 14 to provide the ground reference at the ink jet spray head.

The status display lines are being used to drive the PRINTER READY and LOW INK led's. These lines exit the board at X1 pin 15 and 16 respectively. The PRINTER READY led circuit includes one NAND-gate section of IC 101. It serves to invert the pin 15 signal and drive the led. Three 6.8K ohm resistors have been added on the ink level sense terminals of the ink jet head, as shown on Fig. A3, to simulate the presence of ink to the board. This has been done because the board checks for ink presence before operating. Ink level sense lines leave the Siemens driver board at X2 pins 7, 9, and 10. Due to the addition of these resistors, the LOW INK led is never on except momentarily during power up. The motorized head wiping system which is incorporated into the Siemens units is not currently being used in this application. The actual wiping mechanism was removed but, otherwise, the system has been retained. The switched +5 volts from the MOTOR switch enters the board at X1 pin 14. The motor drive lines are at X2 pins 5 and 8. The MOTOR switch must be on in order to enable the driver channels of the Siemens board. The nozzle array plate heating system consists of a heater resistor and a temperature sensor in the ink jet head and the necessary control and drive electronics on the Siemens driver board. The temperature sensor and the heater resistor are connected to X2 pins 11 and 12. It was found that the array plate heating system tended to dry the xylene ink solution so it was defeated.

As stated before, the major function of the Siemens driver board is to provide the necessary signals to fire the ink jet piezoelectric drivers. There are twelve identical channels on the board for that purpose. Each channel includes an amplitude adjustment potentiometer for fine tuning the output pulses. These twelve controls are located in a line along the side of the board opposite to the X1 input connector and are screwdriver adjustable. They provide an output amplitude range from about 125 to 300 volts peak to peak. It was found through experimentation that output pulses need to be somewhere in the range of 150 to 250 volts to properly drive the piezoelectric transducers. The necessary amplitude will be dependent on the pressure in the ink supply bottle, the ink viscosity, and other related parameters. The pulse width of these output pulses is about 30 microseconds. The exact width is difficult to specify due to the non-square nature of the pulse.

5.5 Siemens Pt80i Spray Head

The Siemens Pt80i ink jet spray head is a drop-on-demand type head with 12 nozzles arranged in two vertical rows. The diameter of each nozzle is 76 μm. Each ink jet channel is concentrically enclosed by a piezoceramic transducer tube. Silver films on the inner and outer surfaces of the tubes serve as electrodes for applying the electric field.

Ink droplets are ejected from the nozzles by momentarily applying an electric field to these transducers. The nozzles that are pulsed are determined by the input signals to the Siemens driver board as described above. The unit, as shipped from Siemens, was designed to print characters using a 12-by-9 dot format. The head in conjunction with its matching driver board was capable of printing up to 300 characters per second.

For this application specific requirements in an ink jet head had to be met which required that the Siemens head be modified in several ways. The unit was designed, originally, such that ink was ejected from the nozzles in the horizontal plane. Also, a collapsible bladder containing the ink was an integral part of the unit. Along with the ink supply were features to monitor the ink level, wipe the nozzle array plate, and maintain the nozzle array plate at a constant temperature. In this application of the ink jet spray head, most of these features were either not needed or simply could not be used due to other considerations. Since the substrate needed to be mounted on a positioning table, the ink droplets from the head had to be ejected in the vertical plane. This, alone, would have eliminated using the existing ink supply since the relative position of the ink jet head to the ink supply is responsible for the static vacuum applied to the ink jet nozzles. It is this static vacuum which is critical to the correct operation of this drop-on-demand system. However, using MOD inks in a xylene solution dictated operating in a closed inert atmosphere which was not possible with the existing head and ink supply system. A new mounting system for the head was devised so that the ink was ejected down onto the substrate. The mounting system includes a means of adjusting the head to substrate distance. Although further studies will be necessary to determine an absolute optimum nozzle to substrate spacing, 300 μm has proven to provide good line definition and so was used for recent testing. A short length of Teflon tubing was attached to the head and connected to a new ink supply arrangement. The specifics of the ink supply system will be discussed in Section 9. The nozzle array plate wiping feature was deemed not necessary and so the actual wiping arm mechanism was removed. The ink level monitoring feature could no longer be used and was overridden. It was also fou.d that the array plate heating system tended to dry the xylene ink solution so it was defeated.

The modified head and the Design Components positioning table were mounted on an aluminum plate having dimensions of approximately 1.3 cm X 35.5 cm X 45.5 cm. Connections to the head from the electronics cabinet is via ribbon cable. The table motors, discussed in Section 6, are connected to the cabinet with 'Jones' style 8-pin connectors. A vertical rod is also mounted on this plate for supporting the new ink supply bottle (Section 9).

6. X-Y POSITIONING TABLE SYSTEM

6.1 Overview

The complete X-Y table positioning system is shown in Fig. A5. The basic system consists of two identical channels, one for movement in the X direction and one for movement in the Y direction, both of which are controlled by the SCCS-85 microcomputer board. A 'joystick' control is

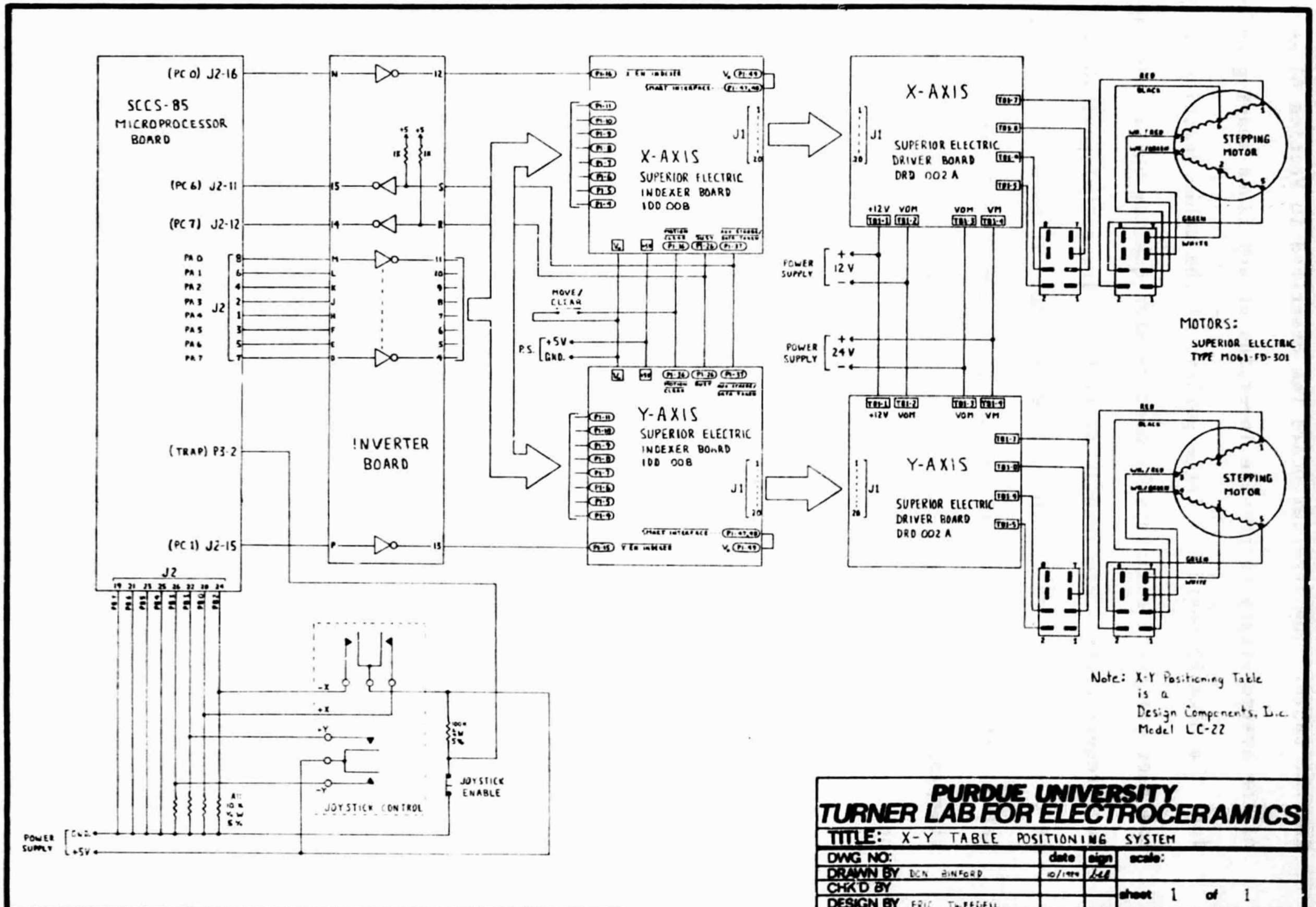


Figure A.5 X-Y Table Positioning System.

also connected to the SCCS-85 board to provide a means of manually moving the table. The inverter board (as described in Section 4) is a part of the system simply to provide inversion of any lines leaving or entering the SCCS-85 board. A large portion of the table positioning system has been purchased commercially because high quality units meeting all our requirements were readily available. The actual positioning table is a model LC-22 from Design Components, Inc. It is a X-Y positioning table with a maximum travel in each direction of 5.08 centimeters. The stepping motors which are part of the positioning table are 200 steps per revolution units made by the Superior Electric Co. Each channel consists of two boards, an IDD008 indexer board and a DRD002A driver board. These boards are part of a line of modules for stepping motor control made by the Superior Electric Company which have the trademark name of ´MODULYNX´.

All move information enters the indexer boards on the 8-bit parallel data line in ASCII format from the SCCS-85 board. The number of steps to be taken and the direction is encoded in the data. Four other lines connect the microprocessor board and the two indexer boards as seen in Fig. A5. These four lines are used in a simple ´handshaking´ routine to correctly sequence signals.

Manual control of table movement is accomplished by pressing the JOYSTICK ENABLE button and then using the ´joystick´ control to move the table as desired. The ´joystick´ function is accomplished in a program that has been appended to the main program as discussed in Section 10. When the JOYSTICK ENABLE button is pressed an interrupt vectors the microprocessor to this ´joystick´ program.

6.2 Superior Electric Indexer Boards

The two indexer boards used in this system are IDD008's made by the Superior Electric Co. They have been specifically designed to be used with the DRD002A driver boards. They are 19.6 cm X 22.4 cm circuit boards having 50 pin edge connectors for input signals and 20 pin edge connectors for connections to the DRD002A driver boards. Robinson Nugent, Inc. edge connectors were used for these connections. They were obtained from Digi-Key Corporation and are type R500. In addition to these two edge connectors there are six spade lug terminals along the same edge of the card as J1. Three of these are for grounds and +5 volt power supply connections and the other three are additional output lines. Figure A6 is a complete circuit schematic for the indexer board.

The indexer boards receive input data in digital format which specifies the number of steps to be taken and the direction. When these commands are received, the indexer boards provide the correct number of properly sequenced phase control signals needed to operate the DRD002A two-phase stepping motor drivers. The indexer boards may be used to operate the motors one step at a time in either direction using a Jog command or to run the motors continuously in either direction using the Run command. The indexer boards can be used in two different basic modes in terms of the way in which move information is inputted. In the Switch Interface mode (P1 pins 47 and 48 left floating) move information is entered on the data bus with external switches which are diode isolated. In this application, which required microprocessor control, the indexer boards are being used in the Smart Interface mode accomplished by jumpering P1-47,48 to P1-49. This allows all move commands to enter

SCHEMATIC DIAGRAM
IDD008 INDEXER CARD

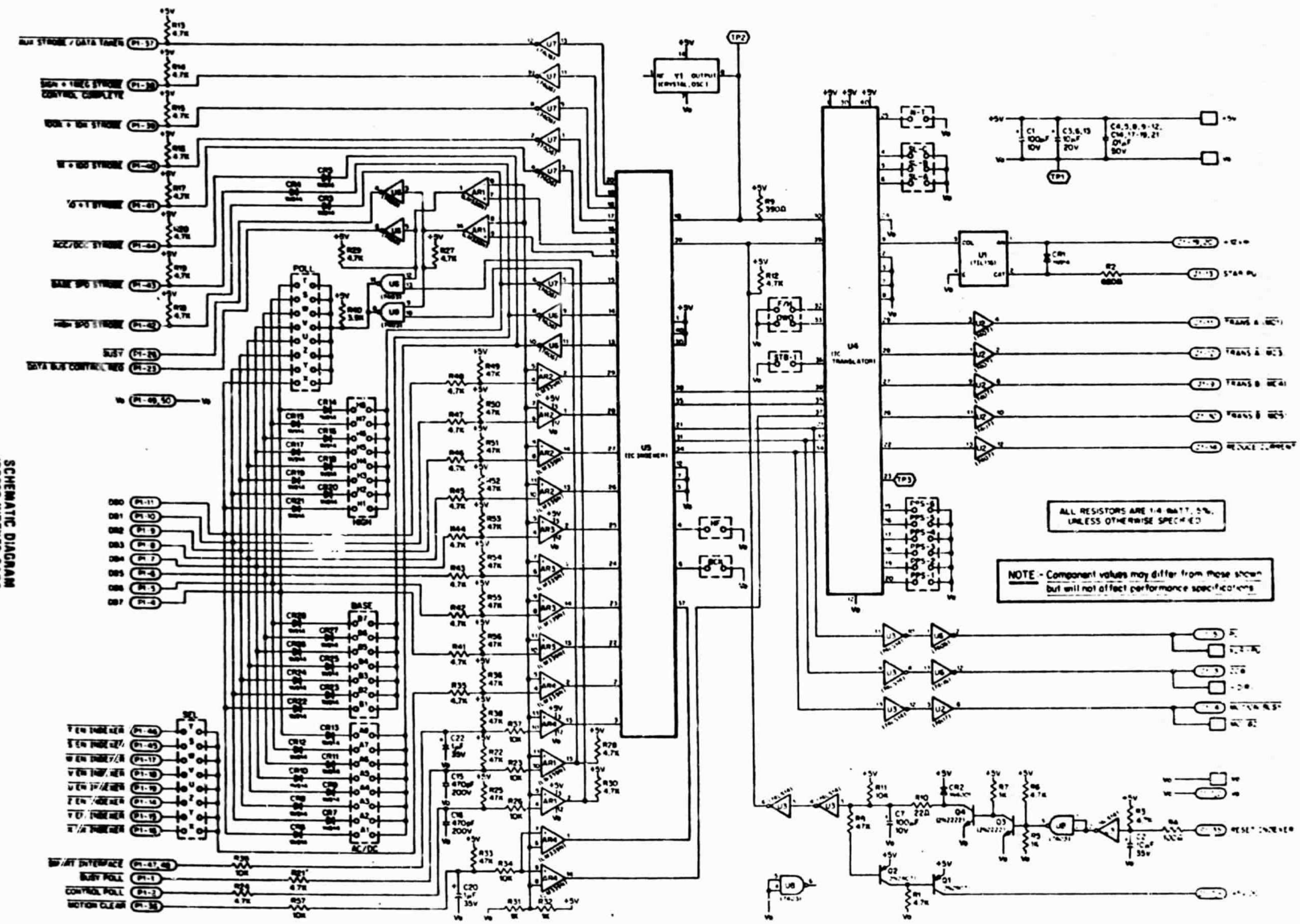


Figure A.6 Schematic Diagram IDD008 Indexer Card.

the indexer boards on the data bus from the SCCS-85 microcomputer board. This mode does require a simple 'handshaking' routine in order to function correctly.

Also contained within the circuitry of the indexer boards are provisions for establishing the base speed, the high speed rate, and the rate of acceleration and deceleration. The values for each of these parameters can be fixed using a series of on-board jumpers or may be entered into the IDD008's memories from the data bus in ASCII format. There are several other operational parameters which can be selected with on-board jumpers. The first of these is unique to each of the two indexer boards in that it basically gives each board an address identification. There are two sets of terminal arrays which are responsible for this address. They are the SEL and the POL terminals. There are axis designations of S-Z for each and so it follows that both sets of X terminals should be jumpered on the X-axis board and that both sets of Y terminals should be jumpered on the Y-axis board. The remaining jumper-selected parameters are set the same for both boards. The base speed, the high speed rate, and the rate of acceleration and deceleration, even though jumper-selected on the board, may be overridden in the Smart Interface mode by inputting the data in ASCII format over the data bus. The following is a listing of the other parameters mentioned and an indication of exactly which terminals are jumpered:

Function	Terminal Pair	Jumpered?	Result
Stepping Mode	F/H OWO	no no	Full-step, two windings on mode
Reduced	STB-1	yes	Reduced standby

Standby Current Option			current option selected
Mid-range Stabilization	N-T	no	Mid-range stabilization feature selected
Mid-range Stability, High Speed Cutout	SL-C SL-B SL-A	yes yes yes	Feedback cut-out frequency of 6134 steps/sec. selected
Half Frequency Option	HF	no	Half-frequency option not selected
Backlash Compensation	BCK	no	No backlash compensation
Resonance Control	PPS-1 PPS-2 PPS-3 PPS-4 PPS-5 PPS-6	no no yes no yes no	Resonance control time of 2000 microseconds selected
Base Speed Programming	B-7 B-6 B-5 B-4 B-3 B-2 B-1	no no no no no no no	With HF not jumpered, base speed of 40 pulses/sec. selected
High Speed Limit	H-8 H-7 H-6 H-5 H-4 H-3 H-2 H-1	no no no no no no no no	With HF not jumpered, high speed of 40 pulses/sec. selected
Acceleration/ Deceleration	A-8 A-7 A-6 A-5 A-4 A-3 A-2 A-1	no no no no no no no no	500 steps/ second/ second selected

In the Smart Interface mode move commands and data are entered into the indexer board through the data bus using a ′handshaking′ scheme as mentioned before. Since individual handshaking response is required before any operation takes place, only one axis address enable input (P1-16 X-axis board or P1-15 Y-axis board) is allowed at a time. The following sequence of events must occur for ASCII characters to be read from the data buss by an indexer board:

1. character put on the data bus.

2. In no less than 62 microseconds the indexer will respond by taking ′not′ DATA TAKEN low thus reading the character on the data bus.

3. The SCCS-85 board sees that ′not′ DATA TAKEN has gone low and responds by taking ′not′ EN INDEXER back high and removing the ASCII character from the data bus.

4. The indexer board senses the low to high transition of This starts the processing of the character read and/or the execution of the move. At the same time that ′not′ DATA TAKEN goes high, ′not′ BUSY goes low indicating to the SCCS-85 board that the indexer is busy and cannot accept data or commands.

5. After the processing and/or execution is completed the indexer board takes ′not′ BUSY back high indicating to the SCCS-85 board that it may again enable the board by taking ′not′ EN INDEXER low again and repeating the above sequence.

By the use of the above sequence, information is transferred from the SCCS-85 board to the indexer board. Not only must this specific sequence be used to input characters but also a definite progression of characters must be entered in order for the indexer to carry out the desired move. One of the first things necessary to specify is the data entry format. The two possible formats are hexadecimal and decimal. An 'O' (the letter O) selects the hexadecimal format and a 'F' selects the decimal format. By default, when power is applied to the indexer board, the hexadecimal format is chosen. It is also chosen whenever the indexer board is given a 'C' command which is a Clear All Registers command. The data entry format remains active until a command specifying the other format is entered.

The indexer card has five registers. Once a register has been selected, data are then entered into the register using a sequence of ASCII characters. Once the data are entered into a register and the register has been closed, another register can be selected by entering the appropriate register select character. The register select characters are as follows:

A	acceleration/deceleration register
B	base speed register
H	high speed register
M	move register
S	delay register

It should be noted that the move register is also selected by power turn-on, a 'not' RESET INDEXER signal, or entry of a 'C' (clear all registers) character. Once a register has been specified by entering one of the five above characters (or by default) then data are entered into the selected register using the following format:

1. Enter the '<' character to clear the previous data and open the addressed register.

2. Enter the data characters.

3. Enter the '>' character to close the register.

All data entered must be within specified limits for the particular register. Also, it should be noted that that data characters are actually only codes for numbers as specified in tables for each register. The required limits and all tables are listed in the Superior Electric indexer board instruction manual and, due to their length, will not be duplicated in this document. In the case of the move register only, the data characters between the '<' and the '>' may be prefixed with a '-' to indicate movement in the negative direction. If '-' does not preceed the data, then movement in the positive direction is assumed by the indexer.

For clarification, some explanation of 'direction of movement' may be in order at this time. In the operating manual for the indexer board, the Superior Electric Co. defines positive movement as that which occurs when the motor is turning in a clockwise direction as viewed from the label end of the motor. If, however, a point directly below the ink jet head is specified as a origin prior to any table movement, then an indexer positive move command moves the table in such a manner that the head is repositioned at a point either on the X or Y negative axis. For that reason, in order to accomplish movement in the positive X or positive Y direction (that is, of the head in respect to the table), a command for movement in the negative direction must be given to the indexer board. In the remainder of this section explaining the operation of the indexer boards, reference to the 'positive direction' refers to the positive direction as defined by the manufacturers of the board. Keep in mind that the actual print direction on the substrate will be just the opposite.

In studies to date, the table has been operated only at a constant base speed. For this reason, the acceleration/deceleration register, the high speed register, and the delay register have not been utilized. As increased substrate speeds are investigated, these registers will have to be used.

There are many ASCII character commands that may be used with the indexer board in addition to the commands already mentioned. Many of these utilize special features of the indexer board which are not needed in this application. The following is a listing of motion commands which are used and a brief description of what each command does:

Command Name	Character	Function
INDEX	G	Initiates a programmed move as specified by previous commands
JOG -	I	Pulses motor one step in negative direction
JOG +	J	Pulses motor one step in positive direction

In this application the G command is the motion command most often used in that it initiates motion for each individual move. When the JOYSTICK mode of operating the table is selected then the two JOG commands, I and J, are utilized.

Referring to Figure A5, it is seen that an external MOVE/CLEAR switch has been provided for the indexer boards. When P1 pin 36 is taken low by pressing this switch then any motion taking place is terminated, without programmed deceleration (if applicable), and the move register is cleared. If this line is held low then all motion except JOG - and JOG + is inhibited.

The Superior Electric Co. indexer board has many features which are not being utilized presently since all the testing done to date has been at fairly low table speeds. The above description of the board and its command structure for the various registers, etc., may seem somewhat complicated. In the future, as an attempt is made to increase print speed, then many of the additional features built into the board will have to be used. However, it is currently being operated using a very simple process with a minimum of commands. The sequence described above for reading ASCII characters from the data bus must be adhered to, but otherwise the following sequence is all that is needed to accomplish table movement:

1. Select the base speed register by inputting a 'B' to the indexer board chosen.

2. Open the register and clear it of previous data by inputting an ASCII '<'.

3. Select the hexadecimal number from the base speed table in the indexer board operating manual which represents the required base speed and input it to the indexer board. This number must be within the range of 00 through FA. (Note: The hexadecimal format

was chosen by default upon application of power to the indexer board.)

4. Input a '>' to close the base speed register.

5. Select the move register with a 'M'.

6. Open the register and clear any previous contents with a '<'.

7. If movement in the negative direction is required then input a '-', otherwise positive direction is assummed.

8. Select the hexadecimal number from the move register table in the indexer board operating manual which represents the required number of steps to be moved. This number must be within the range of 000000 through FFFFFF, although leading zeroes need not be entered.

9. Close the move register with a '>'.

10. Initiate the specified move at the specified base speed by inputting a 'G'.

The two indexer boards in the system, when given the proper commands by the SCCS-85 board, output the proper signals to the associated driver board for each channel in order to accomplish the desired move. A definite format for inputting commands and data to the boards must be followed, but otherwise the process for specifying and accomplishing movement of the X-Y table is fairly simple.

There are three other outputs from the indexer board, two of which are being used. These are buffered translator monitoring signals which exit the board on both the 20-pin edge connector and also on three spade lug terminals. The 'not' MOTION BUSY line is not being used. The -DIR and the 'not' XLR-PU outputs at the quick disconnect terminals are being employed to provide direction and count information to the Counter/Display Driver board. The details of these signals will be discussed later in the L.E.D. Display section.

6.3 Superior Electric Driver Boards

The two driver boards used in this ink jet printing system are part of the MODULYNX trademark line of modules made by the Superior Electric Company. They are type DRD002A cards having dimensions of 19.6 cm X 22.4 cm X 5.7 cm. Inputs to the board are through a 20-pin edge connector and the board outputs and the power supply connections are made via a 10-position terminal board. As stated in the indexer board section above, this board has been specifically designed to be used with the IDD008 indexer board. The DRD002A driver board contains a bipolar chopper stepping motor drive. The indexer board provides the properly sequenced and ramped phase control signals to the driver board to control motor direction, running speed, acceleration, and deceleration. The complete circuit schematic for the driver board is shown in Fig. A7.

The driver board contains two DIP (dual in-line package) switches indicated as S1 and S2 on the circuit diagram. S2 has been provided for matching the board to the various motors that could be connected to it. It adjusts the nominal motor phase current. For the Design Components table that is being used in this system which has Superior Electric Co. type M061-FD-301 motors, S2 needs to be set as follows:

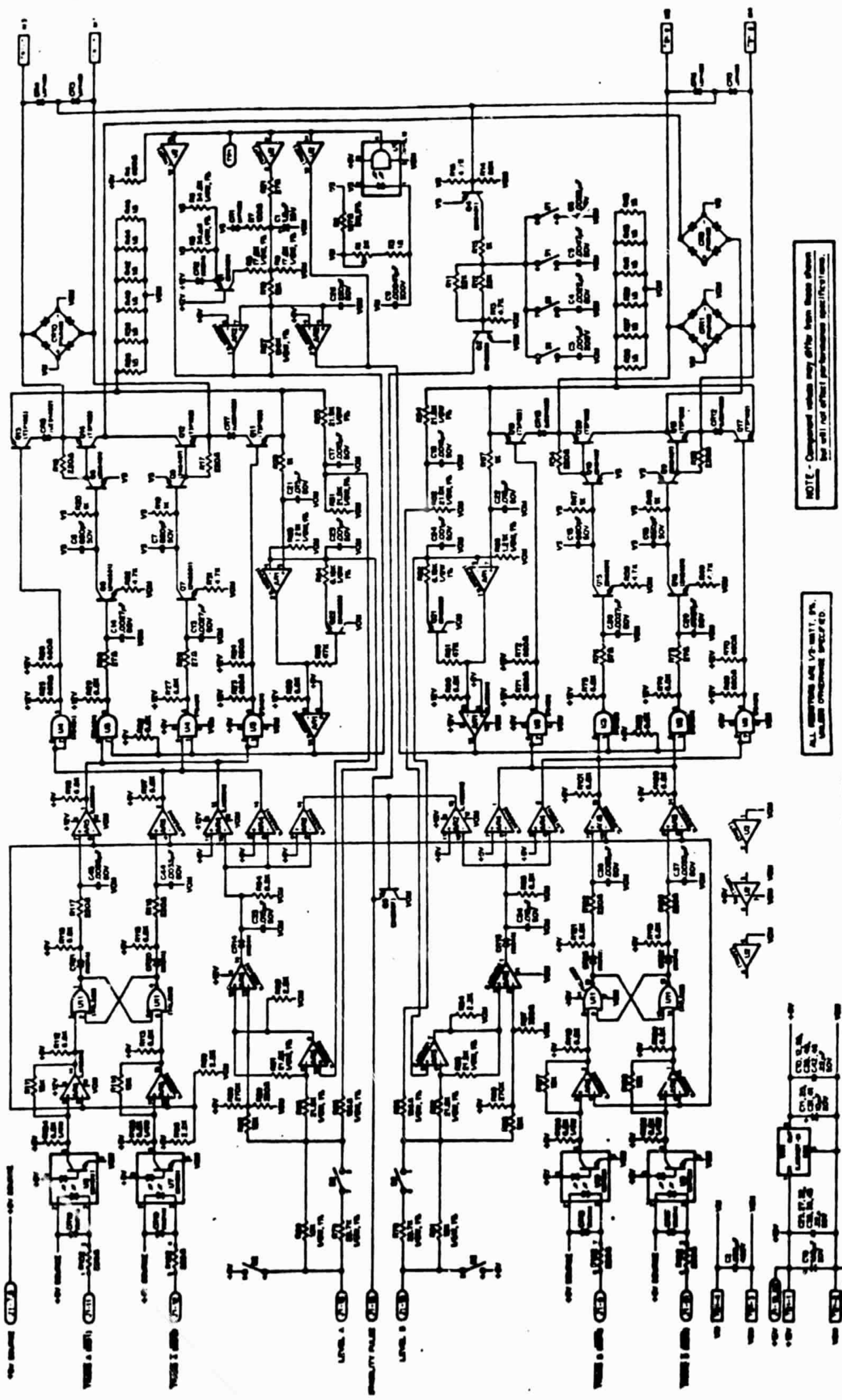


Figure A.7 Schematic Diagram DRD002A Driver Card.

S2 switch position	status
1	off
2	off
3	on
4	on

Switch S1, located near TB1 on the board, is used to adjust the Mid-Range Stabilization feature to a particular motor and power supply voltage used with the board. This feature of the Superior Electric boards improves motor stability at middle range motor speeds. Due to the fact that tests run to date have been at some selected base speed, this feature is not actually used. It is, nevertheless, suggested in the operating manual for the driver board that switch S1 be set as specified for the motor and power supply. The necessary switch settings are given in an appropriate table in the operating manual. The S1 settings for the M061-FD-301 motor and 24 volt power supply combination are as follows:

S1 switch position	status
1	off
2	off
3	off
4	on

Two different power supplies were needed for this board. There is a 12 volt supply which provides power to the logic circuitry on the board and there is a 24 volt supply which is required by the motor circuitry. These two supplies, as discussed in Sections 8.6 and 8.7, are isolated from other power supply sections, that is, the negative sides of these supplies are not tied to the general power supply ground. Also, as recommended, a 1000 ohm, 5 watt, 'bleeder' resistor was added across the output of the Sierracin Power Systems model 3D24 24 volt power supply. It is also seen (Figs. A6 and A7) that a +5 volt line (J1-7,8) leaves the indexer board and is the 5 volt source voltage for the four optocouplers (U7-U10) located on the driver board. Other +5 voltages required for the driver board come from its own on board +5 volt regulator. There is a +12 volt line leaving the driver board at J1-19,20 which is, in turn, a +12 volt source to the indexer board, entering it at its respective J1-19,20. Note that it feeds the anode terminal of an optocoupler on the indexer board (U1) which is part of the Mid Range Stabilization circuitry incorporated into the two boards.

The four main inputs to the driver board are at J1-9 through J1-12. These are labeled as ´not´ MD4, ´not´ MD5, ´not´ MD1, and ´not´ MD3. These four translator inputs are connected to the corresponding translator outputs on the indexer board. The indexer board provides the correct number of properly sequenced phase control signals to the driver board in a manner so that the output signals from the driver board at TB1-5, TB1-7, TB1-8, and TB1-9 move the X or Y axis table motor as required. As mentioned in the indexer board section, this system uses the motors in a Full-Step, Two Windings On mode. Due to the required ´handshaking´ routine between the SCCS-85 board and the indexer board, only one motor at a time is run.

6.4 Design Components X-Y Positioning Table

The X-Y positioning table used in the ink jet printing system is a unit that was purchased from Design Components, Inc. It is a model LC-22 having a maximum travel in each direction of 5.08 centimeters. With the 200 step per revolution motors that are being used with the table, it has a resolution of 25.4 μm per step. The work surface available on the unit as shipped is 10 cm X 10 cm. To this surface has been attached a 10 cm square brass block which is 1.27 cm thick. This block has been machined such that there is a very slightly recessed inner region into which the substrate to be printed is placed. Under the substrate area is a 4.76 mm hole to which a vacuum is applied during the printing operation in order to hold the substrate stationary. A cylindrical heating element is mounted in the block to keep its temperature within the range of approximately 30-35°C. A 120 VAC variable transformer is used to control the temperature.

The two Superior Electric Co. type M061-FD-301 synchronous stepping motors used with this Design Components table are actually part of a Joystick Controller package. It was purchased during initial design stages of the ink jet printing system before the X-Y table was put under microprocessor control. This package was also obtained from Design Components, Inc. and was identified as model JC-103-2. These motors are part of a line of stepping motors manufactured by the Superior Electric Co. under the trademark name of 'SLO-SYN'.

6.5 Joystick Control of Table

As mentioned, during initial design stages of this ink jet printing system the table was only moved manually through the use of a Joystick Controller package purchased from Design Components, Inc. Further development dictated the need for the table to be controlled from a microprocessor so that precise patterns could be printed. After such a system was implemented it was felt that some means of manually moving the table should be retained so that the table could be positioned to the required start position prior to a printing run.

Because of this need, a front panel mounted 'joystick' control may still be used to manually move the X-Y table. This function is now, however, accomplished in a program which is part of the main operating program for the SCCS-85 board. Pressing the JOYSTICK ENABLE button vectors the microprocessor to this joystick program which is appended to the main program. Once this is done, the table is controllable only with the 'joystick' control until the SCCS-85 board is reset. The joystick function uses the Superior Electric indexer boards in their JOG modes.

It can be seen from Fig. A5 that the joystick control information enters the SCCS-85 board on four lines. These lines enter the board at J2 pins 20,22, 24, and 26. They occupy a portion of the I/O port PB and are normally pulled low by the four 10K ohm resistors connected to ground. The other four lines of the PB port are also pulled low as indicated. When the 'joystick' control is moved either one or two of the four control lines are taken high (+5 volts) and in this manner the move information is transferred. Even though it is possible for two of the control lines to be high at one time, table motion can only be in one direction at a time. In other words, only movement parallel to one of the axes is possible. When the 'joystick' control is held in one position, movement occurs in that particular direction until the table reaches the end of its allowable excursion or the control is released to return to its center static position.

7. L.E.D. DISPLAY SECTION

7.1 Overview

The l.e.d. displays which have been designed into the ink jet printing system provide a visual indication of where the ink jet head is in respect to the X-Y positioning table. This is very useful both for 'debugging' pattern data and also for setting up the table for a print sequence. Normally, prior to printing the table is positioned so the ink jet head is located near the lower left-hand corner (as viewed from the front of the cabinet) of the substrate. This then establishes a reference origin. As the printing sequence starts, the numbers read on the l.e.d. displays are the number of steps in the positive X direction or the positive Y direction from this origin.

The entire display section is shown in Fig. A8. Two signal lines are needed for each of the two channels and they come from the corresponding X or Y indexer board. The inverter board is used to invert the two 'not' XLR-IU buffered outputs from the indexer boards since they are negative logic. The Counter/Display Driver board contains two Intersil ICM 7217 IJI integrated circuits which contain all the necessary circuitry to both drive the l.e.d. displays and interpret the count and direction signals from the indexer boards. Clear switches have been provided to reset the two 4-digit l.e.d. displays to zero. The two l.e.d. display modules are 1.27 centimeter high displays having four 7 segment digits each. They are common anode units driven in a multiplexed fashion from the Intersil I.C.'s.

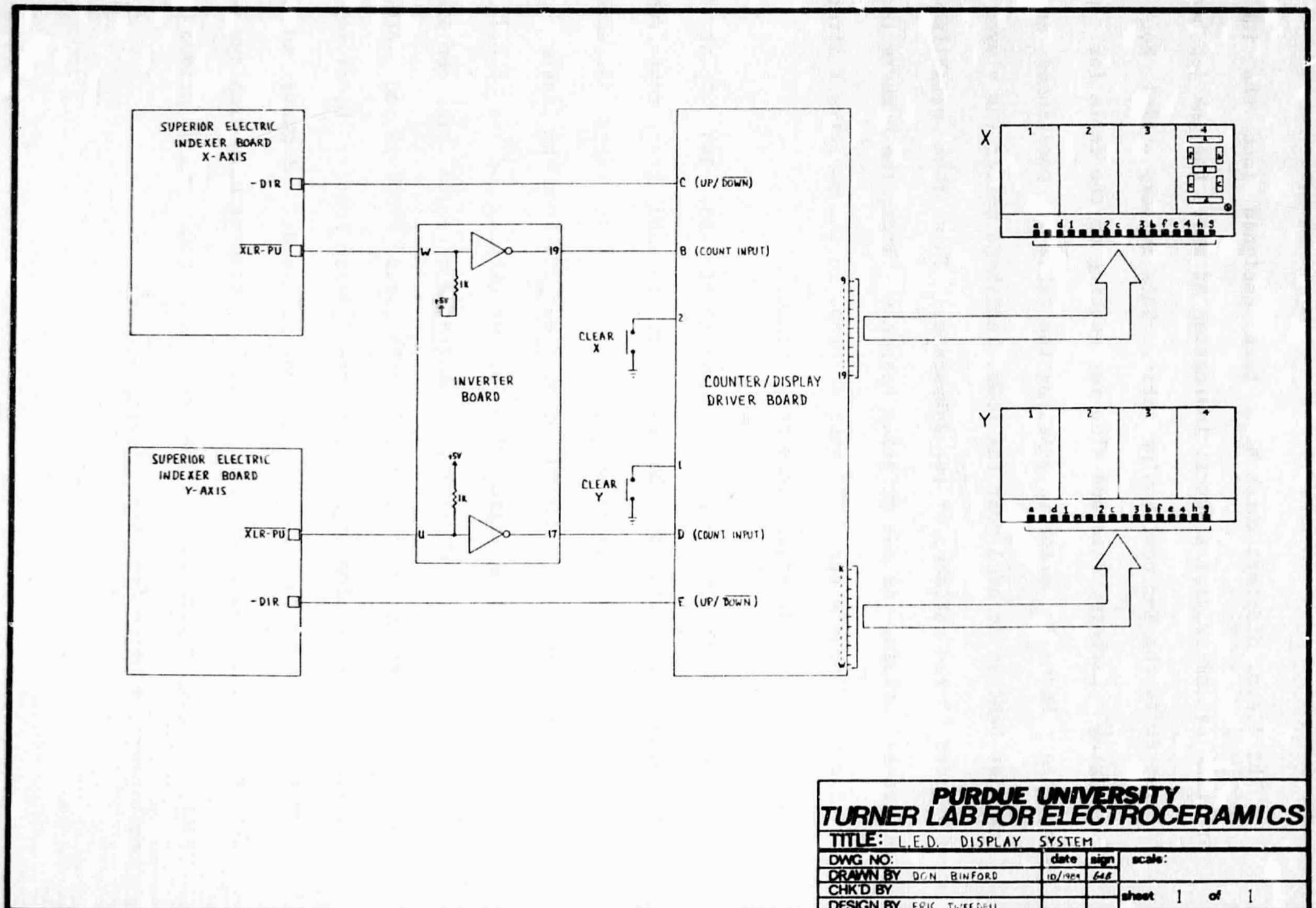


Figure A.8 L.E.D. Display System.

7.2 Counter/Display Driver Board

With the exception of the two inverters being used, the Counter/Display Driver board contains all the circuitry used in the L.E.D. Display section. Except for one on-board +5 volt regulator, all that circuitry is contained within two identical integrated circuits made by Intersil. These and the 7805 regulator are mounted on a 'Vector' #3677-2 circuit board having dimensions of 11.4 cm X 16.5 cm. All inputs and outputs to and from the board are via a 22 pin edge connector. The circuit schematic for the Counter/Display Driver board and its connections to the two displays are shown in Fig. A9.

The power supply ground enters the board at pin A and an unregulated +15 volt line enters at pin Z where it inputs the +5 volt regulator. The X-axis count signal is at pin B and the Y-axis signal is at pin D. Directional signals for the X and Y channels enter the board at pins C and E respectively. When the 'not' XLR-PU output from the indexer board goes from high to low, then back to high, the indexer translator advances one translator sequence in the direction indicated by the -DIR signal, i.e. the motor is moved one step in the specified direction. Remember that the count signal that the Counter/Display Driver board gets is the inverse of the 'not' XLR-PU output from the indexer board. If the -DIR line is in the low state when this occurs then movement is in the negative direction and the counter will count down. If the -DIR line is in the high state when this occurs then movement is in the positive direction and the counter readout will increase. Intuitively, this may seem exactly opposite to what one would expect regarding the status of the -DIR line and the direction of travel. This

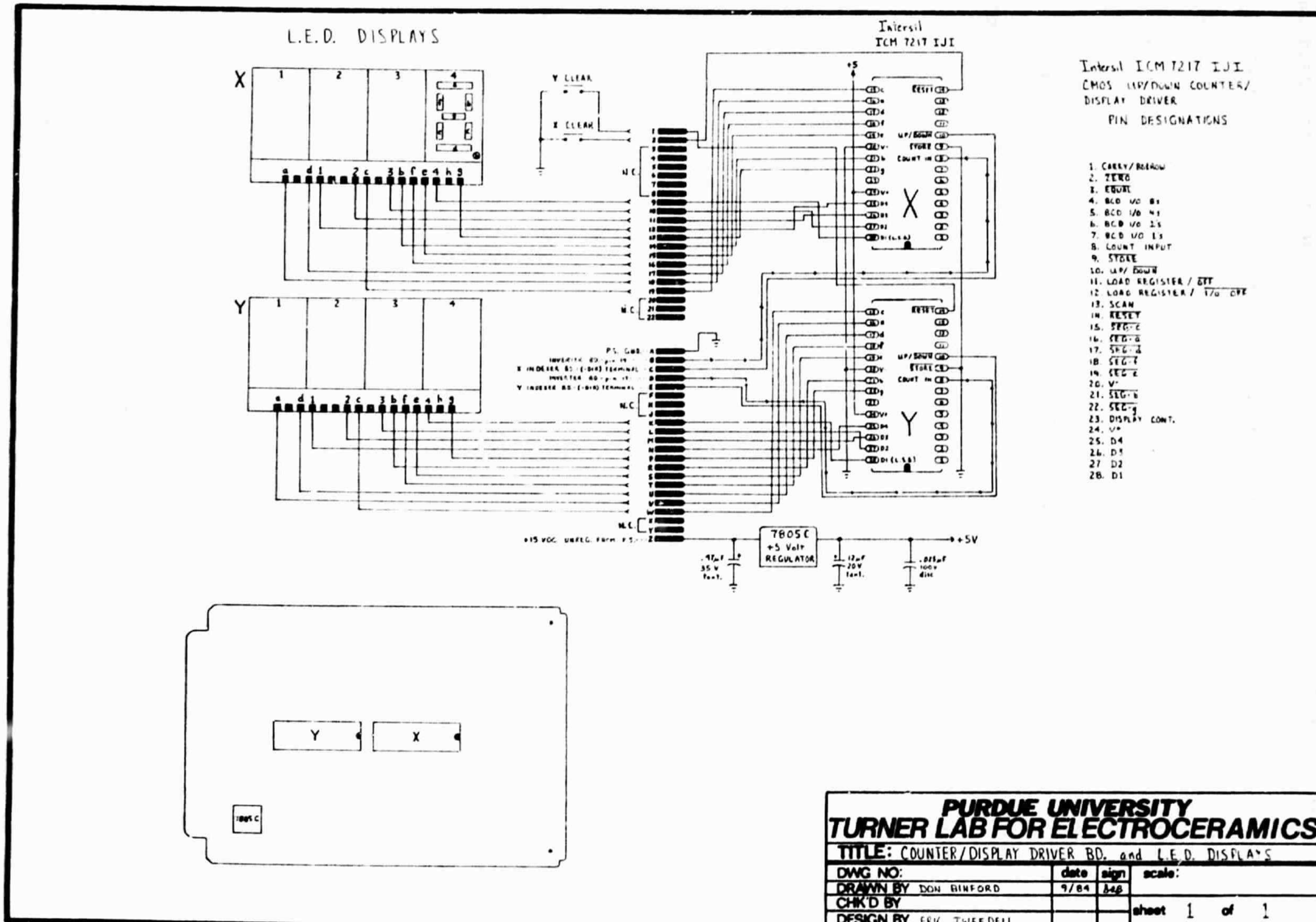


Figure A.9 Counter/Display Driver BD. and L.E.D. Displays.

confusion arises due to the two possible points of reference regarding the movement. In the indexer board operating manual, positive direction is referred to as that which causes clockwise motor rotation when viewed from the label end of the motor. This means that the -DIR line is low for movement in the positive direction. When viewing the X-Y table from in front of the equipment cabinet, CW rotation of the X motor causes table movement to the right and CW rotation of the Y motor causes table movement away from the viewer. However, the direction that the ink jet head is moving in respect to the table is in the negative X or the negative Y direction. For that reason movement in the positive direction from the specified origin on the substrate is accomplished by commanding a negative move within the indexer board. The clear switches connect to pins 1 and 2 of the board and accomplish the clearing function by pulling the reset pins of the I.C.'s to ground momentarily. Outputs to the X-axis l.e.d. display are on pins 9-19 and outputs to the Y-axis l.e.d. display are on pins K-W.

The Intersil ICM 7217 IJI integrated circuit is a 28 pin CMOS device designed to drive four digit common anode l.e.d. displays. The count input at pin 8 is a Schmitt trigger input in order to permit operation in noisy environments. Pin 9 on the I.C. ('not' STORE) has been pulled low in order for the contents of the internal counter to be transferred to the 7 segment outputs. The 7 segment outputs and the four lines D1-D4 operate in a multiplexing arrangement to drive the l.e.d. displays. The reset line to clear the internal counter is at pin 14 of the I.C. The count signal enters the integrated circuit at pin 8 and whether the counter counts up or down depends on the logic level at pin 10. If pin 10 is high then the I.C. functions as a up counter. Conversely, if pin 10 is at a low logic state then the counter counts down. There are several features incorporated into this Intersil I.C. which are not being used in this system, hence there are many unused pins on the I.C.

7.3 L.E.D. Displays

The two l.e.d. display units are identical assemblies made by the National Semiconductor Co. They are common anode, four digit, seven segment character with decimal point modules having part number NSB 5882. The actual displayed characters are 1.27 cm (0.5 inch) high. Connections are made to the units on solder pads along the lower rear edge of the assemblies.

In this application there was no need for decimal points so the indicated ´h´ connection was not used. There are eleven lines feeding each assembly. Pad numbers 1-4 are the common anode connections for each digit. Note, however, that the D1 line from the Intersil ICM 7217 IJI connects to the ´4´ pad on the display. This is because the ´4´ digit is the least significant digit of the display. The other seven connections to the display (pads a-g) are the segment drive lines.

The two displays which are used to indicate table position are mounted in rectangular cutouts in the front panel of the equipment cabinet. The corresponding CLEAR switch is located to the right of each l.e.d. module.

8. POWER SUPPLY SECTION

8.1 Overview

The power supply section for the ink jet printing system is a combination of linear power supplies. The complete power supply section is shown in Fig. A10. Two of the power supplies have been purchased commercially and the remainder were designed and built using off-the-shelf conventional components. There is one MAIN POWER switch which controls the AC input to all the power supply subsections except the +24 volt power supply for the Superior Electric driver boards. It has its own separate switch identified as X-Y POWER. Both of these have front panel mounted indicator lights to show when power is applied. The 2.5 ampere fuse for the +24 volt power supply is front panel mounted, however, the main power fuse is located on the rear panel.

There are three other front panel mounted switches marked 5V, 12V, and 40V which control these three voltages to the Siemens Driver board. There is a l.e.d. associated with each of these switches, all three of which are located on the front panel. One other l.e.d. labeled 15V is provided to indicate whether the +15 unregulated voltage is present.

There are actually three other components which are part of this system which require 115 VAC power. There is a cooling fan for providing ventilation for the equipment cabinet and there is a variable transformer which is used to control the temperature of the brass block which the substrate rests on. Finally, there is a photohelic pressure switch/gage (Fig. A11) which has 115 VAC power applied to it. It is the

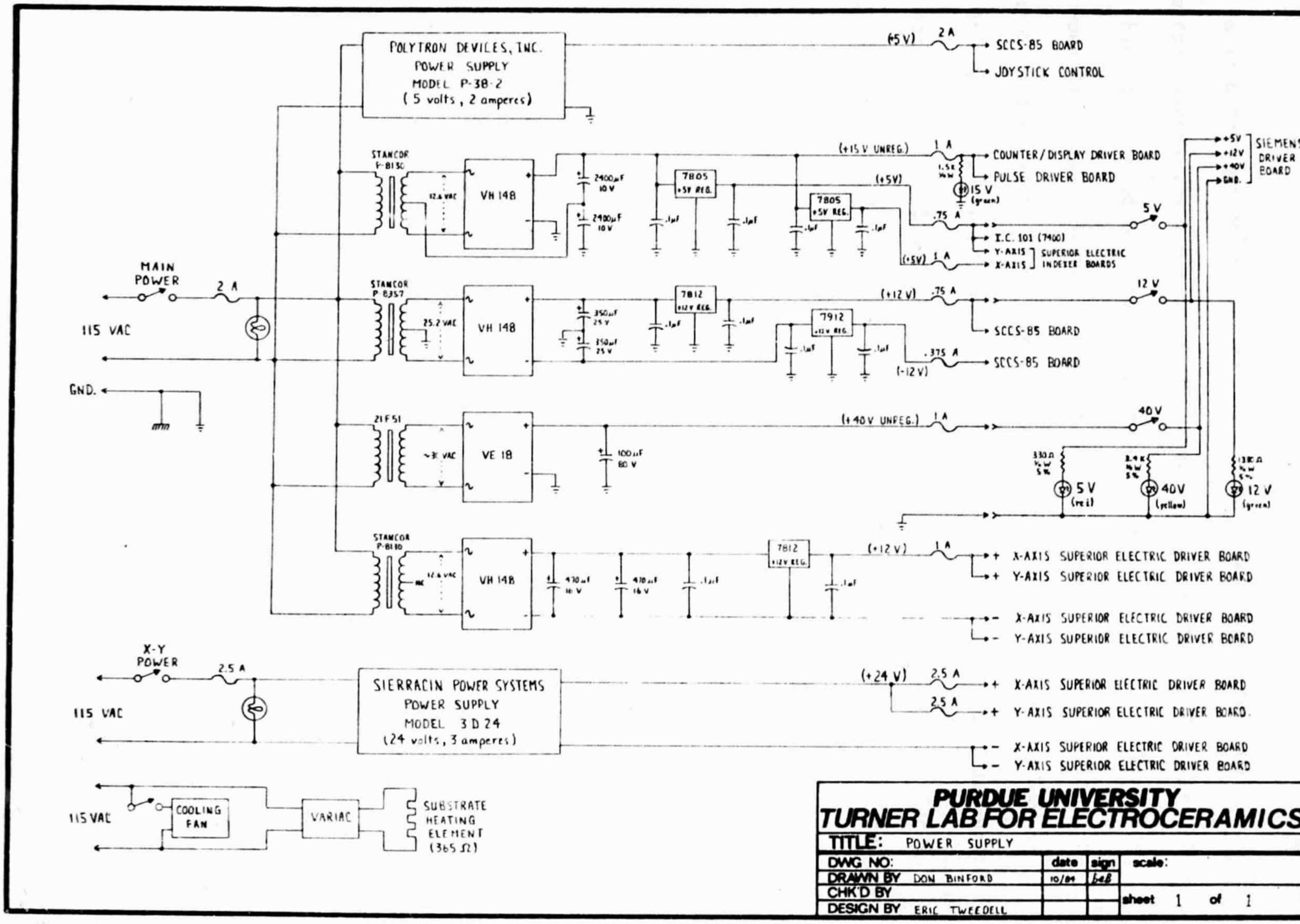


Figure A.10 Power Supply.

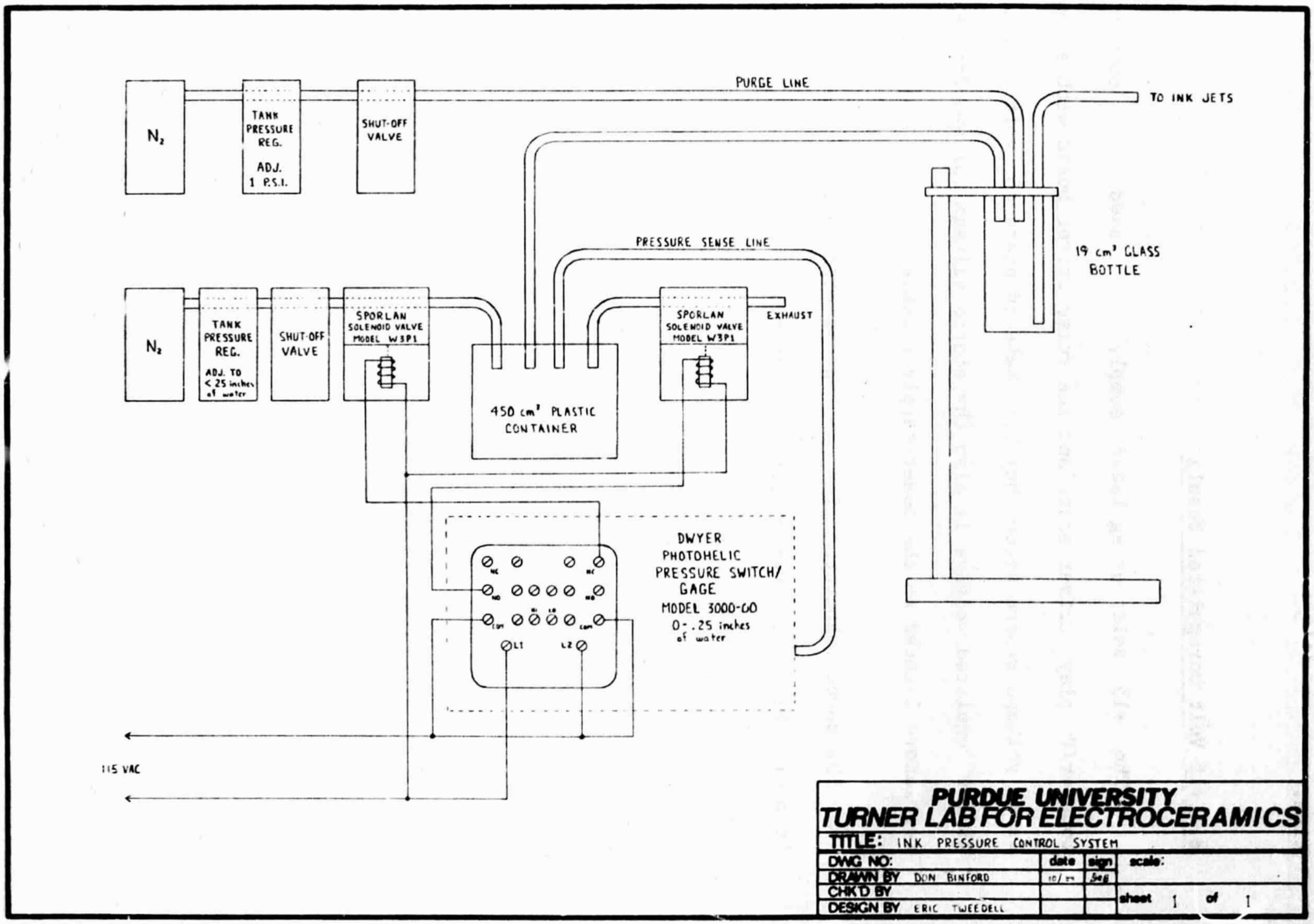


Figure A.11 Ink Pressure Control System.

heart of the ink pressure control system described in the following section.

8.2 +15 Volt Unregulated Supply

The +15 volt unregulated supply is used to supply the Counter/Display Driver board and the Pulse Driver board with an unregulated voltage source since they both have on-board +5 volt regulators. This unregulated voltage is also the source voltage for two 7805 +5 volt regulators located on the power supply chassis.

The power transformer for +15 unregulated voltage section is a Stancor P-8130 which has a 12.6 VAC center tapped secondary. The secondary voltage is applied to the inputs to a Varo brand VH 148 single package rectifier bridge assembly. It has a 6 a rating. The negative output of the bridge is grounded and the positive side is filtered using a series combination of two 2400 μf capacitors. The center tap of the secondary connects to junction point of these two capacitors. This unregulated output is fused at 1 a and, as shown in Fig. A10, there is also an l.e.d. in series with a 1500 ohm resistor to indicate the presence of this voltage. The approximate current draw from this supply is 250 ma.

8.3 +40 Volt Unregulated Supply

The +40 volt unregulated source is used in the ink jet system by the Siemens Driver board. It is used in the output section of the board in order to generate an output pulse of sufficient amplitude to the piezoceramic drivers. As indicated, there is a separate switch for application of this voltage to the board and there is a 1 a fuse to protect the supply.

The secondary winding voltage of the transformer used in this section is approximately 30 VAC. This enters a Varo bridge package (VE 18) rated at 1 a. The AC is rectified and then filtered by the 100 μf, 80 v capacitor from the positive output to ground. The quiescent draw from this section is in the range of 10 ma.

8.4 +5 Volt Supplies

There are three separate +5 volt regulated supplies used in this system. The first is a +5 volt, 2 a rated module made by Polytron Devices, Inc. It is a model P-38-2 designed so that it may be mounted directly to a circuit board. It supplies power to the SCCS-85 microcomputer board and the 'joystick' controller with connects to the SCCS-85. It is fuse protected at 2 a and the quiescent state current flowing from this supply is approximately 700 ma.

The remaining two +5 volt supplies use 7800 series three pin voltage regulators and their input voltage is the +15 volt unregulated supply. Each of these 7805 +5 volt regulators have 0.1 μf capacitors on their inputs and outputs for improved noise rejection and regulation. One of the regulators supplies the logic circuit power to the Superior Electric X-axis indexer board. The other unit supplies +5 volt power to the Y-axis indexer board, the Siemens Driver board (via the 5V switch), and I.C. 101 (debounced switch circuit) shown on Fig. A3. The current supplied by each of these three pin regulators is about 400 ma.

8.5 +12 and -12 Volt Supply

One section of the power supply is used to supply both the positive and negative 12 volts to the SCCS-85 board. Additionally, this same +12 volts goes to the Siemens Driver board. The 115 VAC is reduced to 25.2 VAC by a Stancor transformer P-8357. Its secondary is center tapped and this center tapped connection is tied to the power supply ground. A Varo VH 148 bridge circuit rectifies this AC and a 350 μf capacitor shunts both the positive and negative bridge outputs to ground to provide proper filtering. Both the positive and negative outputs from the bridge are approximately 16 VDC unregulated. These are then the source voltages for the 7812 (positive voltage regulator) and the 7912 (negative voltage regulator) as shown in Fig. A10.

The +12 volt regulated output line is fused at 0.75 a and as mentioned, is the supply for both the SCCS-85 board and the Siemens Driver board (via the 12V switch). The -12 volt regulated output line is fused at .375 ampere and supplies only the SCCS-85 board with power. The Siemens board demands about 70 ma from the +12 volt supply. The SCCS-85 circuit board requires approximately 20 ma from both the +12 volt and -12 volt supplies.

8.6 +12 Volt Superior Electric Driver Board Supply

This portion of the power supply provides the +12 volts needed for both Superior Electric Co. driver boards. It is a separate power supply because it was necessary for it to be isolated, i.e. the negative side of this supply is not tied to the common power supply ground. A Stancor P-8130 provides 12.6 VAC to the VH 148 bridge assembly. This rectified signal is filtered by the two 470 µf capacitors indicated on the schematic. This is the input voltage for a 7812 regulator whose output is fused at 1 a. The two 0.1 µf capacitors on the input and the output of this device are for noise immunity and improved regulation. The input voltage to the 3-pin regulator is approximately 16 VDC. The maximum current required by each of the driver boards is 550 ma.

8.7 +24 Volt Supply

A Sierracin Power Systems model 3D24 power supply was chosen to supply this needed voltage to the Superior Electric driver boards. It is rated at 4.8 a and the lines going to each driver board are fused at 2.5 a. A 1000 ohm, 5 watt resistor was added across the output of the supply as specified by the Superior Electric Co. driver board instruction manual. This serves as a bleeder resistor for the large power supply filter capacitors which were also specified in the instruction manual.

9. INK SUPPLY AND PRESSURE CONTROL SECTION

9.1 Overview

The ink supply and pressure control portion of the ink jet printing system is a critical factor in the correct operation of this system. The system must feed the 'MOD' inks which are in a xylene solution to the Siemens Pt80i spray head in such a manner so that the resultant static vacuum at the ink jet head provides the proper meniscus at the ink jet nozzles. This meniscus is important in determining droplet size and line quality. The meniscus is affected by several factors, the static vacuum at the Pt80i head only one of them. It is also dependent on both the ink solution surface tension and viscosity. Because of this, these two parameters must also be given careful consideration.

The components of the system are shown in Fig. A11. The ink to be printed is contained in the 19 ml glass bottle and is maintained at a level above the bottom of the line feeding the head. The three tubes which enter the bottle are actually fitted and sealed into a cap which is permanently affixed to the stand holder. This makes the process of changing inks and purging the system with xylene simple. The stand and holder arrangement makes it possible to adjust the height of the ink bottle. The bottle height is adjusted to some point below the level of the ink jet head so that a static vacuum occurs at the head. The line going from the 450 ml plastic bottle to the ink bottle is nitrogen gas under a pressure of about 0.1 inch of water column. The nitrogen gas is necessary to provide an atmosphere in the ink bottle to which the ink will not react and the slight pressure is necessary to offset a portion of the static vacuum at the ink jet nozzles due to the lower level of the ink. The Dwyer Photohelic Pressure Switch/Gage and the two Sporlan solenoid valves provide a means of maintaining a reasonably consistent pressure in the ink bottle. The approximate static vacuum at the ink jet nozzles is achieved by adjusting the ink bottle to a point where the ink level is roughly 2.5 cm below the nozzle orifices. Fine control of this required static vacuum is accomplished by the pressure controller. The plastic container serves as a buffer to cushion the on-off operation of the two valves.

The desired effect of this mechanism is to provide a consistent, slightly concave meniscus at the ink orifices in the ink jet head. This consistency is essential so that an ink droplet is ejected only when the piezoceramic driver for a particular orifice is pulsed. If the static vacuum is too great, then ink is never ejected from the head or if the static vacuum too little then ink may drip from the openings without the piezoelectric drivers being energized.

In normal printing operation the purge line nitrogen gas source is closed off. The purge system has been provided as a means of flushing the ink jet head with xylene after printing to avoid the clogging of the ink jet orifices. To purge the system the ink bottle is replaced with a bottle containing pure xylene. The pressure line from the plastic container to the ink bottle is closed with a pinch valve and a catch container is placed under the head. The shut-off valve for the purge line is then momentarily turned on, forcing the xylene through the ink supply line and the ink jet head openings.

9.2 Siemens Pt801 Head Modifications

The Siemens Pt 801 ink jet head is a 'drop-on-demand' style head purchased directly from the Siemens Corporation. It has been designed with 12 nozzles arranged in two vertical rows, each nozzle having a diameter of 76 μm. Its original intended purpose was the printing of alphanumeric characters onto paper using a 12-by-9 dot format. The Siemens driver board that is being used with this system has been specifically designed to be used with this print head.

For this application, certain changes had to made to the print head. First, since ink had to be ejected down onto a substrate resting on a horizontal table, a head mounting mechanism had to be designed for the head so it could accomplish this. It was originally designed to be mounted so that ink droplets were ejected along a horizontal line rather than downward in a vertical line as needed for this system. Along with this was incorporated a means of adjusting the head to substrate distance.

The original ink container could no longer be used in this new position so the original ink well orifice was enlarged to allow the attachment of a 2.5 mm I.D. teflon tube. This, then, was sealed into the ink bottle cap as described earlier. The Teflon tube was essential here because it resists deterioration due to the presence of xylene. Other tubing used in the ink supply and pressure system is 3.175 mm I.D. tygon tubing. The original Pt80i head also had provisions for monitoring the ink level, wiping the nozzle array plate, and maintaining the nozzle array plate at a constant temperature. The required new method of supplying ink to the head made it impossible to utilize the ink level sensing system and the head wiping mechanism was simply not required in this application. Testing determined that the array plate heating system tended to dry the xylene-ink solution so it was disabled.

In studies to date, only one of the twelve ink head tracks has been u at any one time. This was done so that an attempt could be made to optimize the various parameters associated with the printing of well defined lines. Testing seemed to indicate that a head to substrate spacing of approximately 300 μm printed well defined lines down to widths of 150 μm. It is hoped that by reducing the nozzle orifice diameters and by determining optimums for other parameters, that it may be possible to produce high quality lines in the range of 50 μm.

9.3 Ink Pressure Control System

The Dwyer model 3000-00 Photohelic Pressure Switch/Gage compares atmospheric pressure to the pressure in the sense line from the 450 ml container. Accordingly, then, it controls the two solenoid valves to maintain a relative constant pressure in the container. Using this method a pressure somewhere in the range of 0.05 to 0.1 inches of water is established in the polyethylene container.

The Dwyer unit, mounted on the right side of the equipment cabinet, has a range from 0 to 0.25 inches of water. The device is adjusted by setting an upper and lower pointer on the 0 to 0.25 scale. For this application, the lower point has been set at 0.05 inches of water and the lower at 0.15 inches of water. The actual meter movement pointer fluctuates with the slightly varying pressure. If the pressure gets below the lower set point then the intake solenoid is energized, opening the valve and allowing nitrogen to flow into the container. This causes an increase in pressure and as soon as the pressure is just slightly above the lower set point, then the solenoid is deenergized and the valve closed. The exhaust valve, during this time, has, of course, been closed. Should the pressure in the container increase beyond the upper pressure set point, then the exhaust valve would be opened and remain in the open position until the pressure dropped slightly below the upper set point. It is in this fashion that the Dwyer controller maintains a fairly constant pressure in order to offset a small portion of the static vacuum at the ink jet nozzles.

The Sporlan brand solenoid valves are model W3P1. They are 115VAC devices which are normally closed in the deenergized state. As shown on Fig. A11, the pressure regulator between the nitrogen gas source and its shut-off valve must be adjusted to a point below 25 inches of water for the system to work properly.

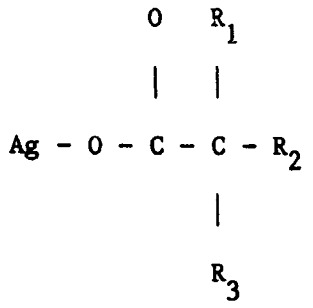
9.4 Purge System

The purge system has been provided so that the ink head nozzles and the supply tube can be flushed with xylene after printing is finished, when changing from one ink to another, or whenever the nozzle orifices should clog for any reason. To purge the system, first close off the pressure line running from the 450 cubic centimeter container to the ink bottle. Replace the ink bottle with a bottle containing xylene and place some kind of catch container under the ink jet head. Verify that the nitrogen gas source regulator for the purge system is set to a point no greater then 1 p.s.i. Now, opening the purge line shut-off valve briefly two or three times forces the xylene through the ink supply line and the nozzles. This purging opertion should be done any time that the system is going to set idle for any more than a few hours.

9.5 Ink Compositions

The viscosity and surface tension of the water based Siemens ink were measured at 25°C and found to be 18 mPa.s and 47 mN/m, respectively. These values were used as starting points for ink development although it was appreciated that both the viscosity and surface tension of the Siemens ink at the nozzles would be lower due to the heater used in the head. The basic ink chemistry selected was metallo-organic compounds with oxygen as the hetero atom bridge because of the background developed during earlier studies in the Turner Laboratory. Xylene was selected as the solvent because the desired compounds have a high solubility, and because it has a low viscosity and a high vapor pressure. A silver conductor was chosen for evaluation of ink jet printing partly for economic reasons, and partly because preliminary experiments showed that good adhesion to ceramic substrates could be achieved without the addition of base metal compounds.

The silver compound selected was Ag neodecanoate with formula



where $R_1 + R_2 + R_3 = C_8H_{19}$. The tertiary ligand gives enhanced solubility in xylene over secondary or primary ligands, and the ten carbon atoms is a good compromise between solubility and metal content. The $C_9H_{19}COOAg$ contains 38.6 w/o Ag, and the solubility in xylene is such that solutions containing more than 20 w/o Ag are stable. Upon heating in air at a rate of $10^{o}C$/minute, silver neodecanoate begins to decompose at $175^{o}C$, has its maximum decomposition rate at $230^{o}C$, and all carbon is gone at $250^{o}C$.

Several procedures are available for synthesizing silver neodecanoate, and the one selected was a double displacement reaction following the equations:

$$C_9H_{19}COOH + NH_4OH \rightarrow C_9H_{19}COONH_4 + H_2O$$

$$C_9H_{19}COONH_4 + AgNO_3 \rightarrow C_9H_{19}COOAG + NH_4NO_3$$

This synthesis route eliminated the high temperatures required in some procedures, and eliminated the possibility of introducing any inorganic constituents other than silver. The NH_4OH was added to the neodecanoic acid while stirring at room temperature to produce the ammonium soap. An aqueous solution of $AgNO_3$ was then added while stirring, and the Ag neodecanoate precipitate repeatedly washed with hot ($80^{o}C$) water. The precipitate would typically analyze 35 w/o Ag, which indicated that some water was retained. Equal volumes of water and xlyene were then added, the 2 phase liquid agitated for 2 hours, and the organic layer removed and filtered after separation. The silver concentration in solution was then increased to the desired value by vacuum distillation of some of the solvent at room temperature. The three inks used for printer evaluation had the properties given in Table 9.1.

Table 9.1 Properties of Experimental Silver Inks.

Ink No	Viscosity (mPa.s)	Surface Tension (mN/m)	Density (g/ml)	Ag (w/o)	Ag neo-decanoate (w/o)
1	3.6	35.4	1.058	16.0	41.5
2	5.5	39.3	1.072	17.1	44.3
3	10.2	40.0	1.088	18.3	47.4

10. SOFTWARE

The program controls two Superior Elctric indexer and stepper motor driver boards. Written in 8080 assembly language, it is designed for a minicomputer with parallel in/output ports. The stepper motors move an x-y table in 25 μm increments for a 5 cm maximum travel. This program requires a data file which contains information for a pattern. Data are produced by reducing an existing pattern to lines and rectangles or constructing any pattern out of lines and rectangles. A flow chart for the main program is shown in Fig. A12, and the main program itself is given in Appendix A2.

The indexer and driver boards are part of the Superiod Electric 'Modulynx' (tm) system. The indexer board (PN.IDD008) contains logic for a 'smart' interface. There are two versions of the interface, the simpler uses two handshaking lines and enable lines for the X and Y axes. The indexer board has a parallel 8 bit data buss. The buss and the handshaking lines are connected to two parallel ports (see Fig. A5). The addresses of the two ports are 10h and 12h (h = hexidecimal). The indexer data buss is connected to the 10h port. Indexer commands and index (move) data are passed over the buss in ASCII code. The indexer

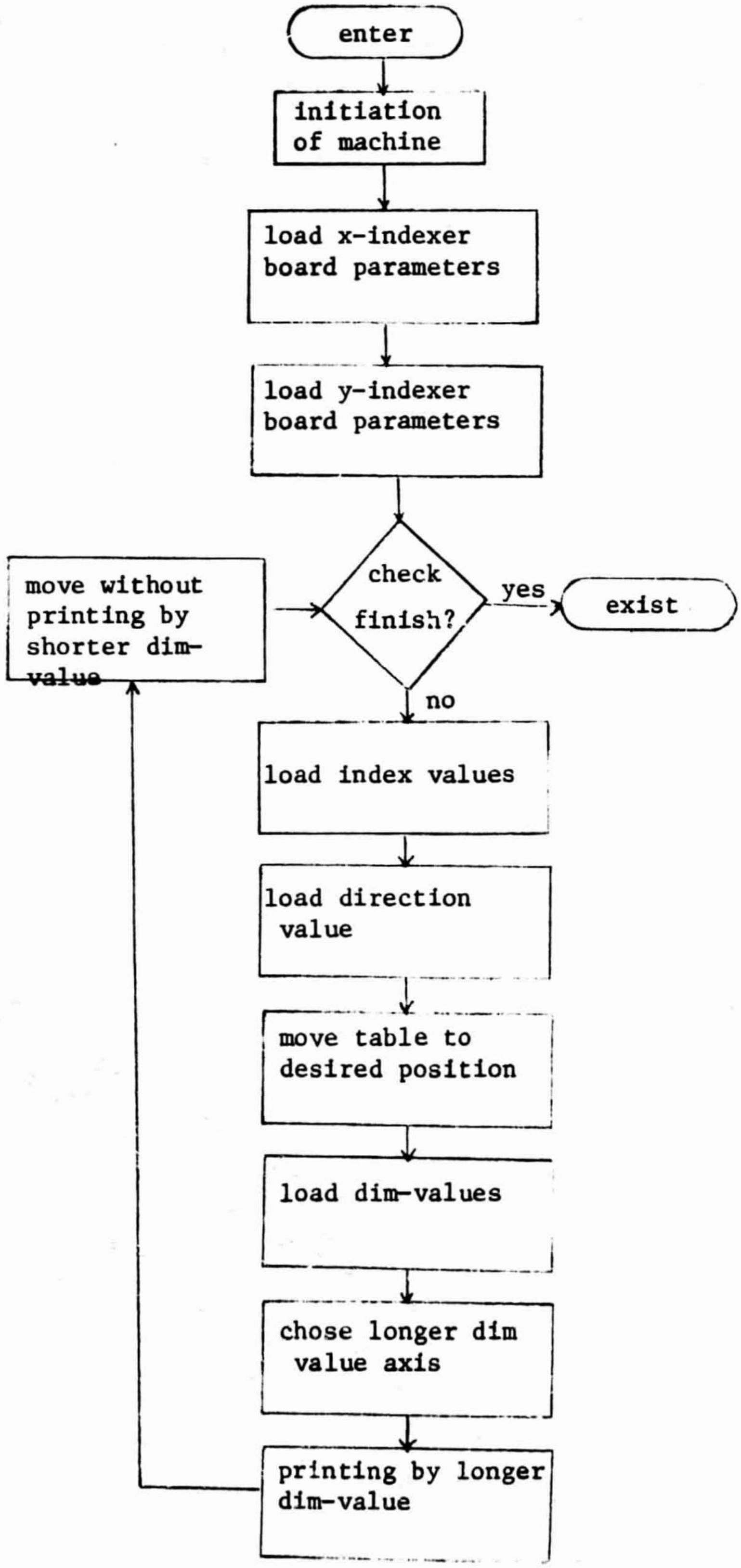


Figure A.12 Flow Chart for Main Program.

board receives input data which specifies the number of steps to be taken and the direction. The port at 12h is split so half the bits are inputs and half are outputs. Below is a diagram of the connections to the 12h port.

busy	data taken	not used	not used	head o/off	not used	Y en.	X en.
80h	40h	20h	10h	08h	04h	02h	01h
inputs				outputs			

The diagram indicates the function of each bit in the 12h port. Parallel port input/output is done by means of the ´in´ or ´out´ commands in conjuntion with the ´a´ register. Each bit can be considered part of a eight bit binary number that is addressed by the hexidecimal number below it. The far right bits are assigned to the enables so that a one in the 01h position enables the x axis and a one in the 02h position enables the y axis. The program sections boxy and boxx control the enables. The enable command (01h or 02h) is passed through subroutines move or movel to subroutine hand which controls the ports. The 08h bit is used for the ink jet head on/off control. This bit (output line) is manipulated by subroutine movel. The 80h bit is the busy line and the 40h bit is the datataken line. These are handshaking lines which regulate the flow of data to the indexer boards and moniter progress of table moves. Note that subroutine hand is used all through the main program. It along with movel (a modified form of hand) control both ports. Often the "a" register is loaded with a "mvi" command immediatly before hand is called. Whatever is in the "a" register is put on the indexer buss through the 10h port. Hand also controls the 12h port,and so outputs the enables and calls the handshaking routines dataken,busy and wait. Listed below are data that provide board initialization and coordinates for a rectangle and a line.

DATA

```
one   org 1000h
two   db    01h,'B','<','0','A','>'
three db    02h,'B','<','0','A','>'
four  dw    0000,0000,0000,02dch,0023h
five  dw    02dch,0000h,2d00h,0000h,0270h ; line
six   dw    2e2e,2e2e,2e2e
```

The org statement directs the assembler program to begin at line 1000. Data starts at memory location 1000h when loaded. The db and dw directives define data to be stored. The next two lines initialize the indexer boards. The 01h in line two enables the x axis, and similarly the 02h enables the y axis in line three. The indexer command B is the address for the base speed register on the indexer board. The brackets open and close the register and the hex number 0A sets the base speed to 400 steps per second. The base speed can be set to many different speeds and the indexer manual lists the various settings. Quotes direct the assembler to leave ASCII code. Subroutine bdset passes the initialization data directly to the indexer. This subroutine can be easily changed to pass the acceleration and high speed commands if they are needed. Line six contains ASCII code for the end of file symbol,a period. If the program reaches end of file it jumps to the microcomputers moniter program. Lines four and five are pattern data for a rectangle and a line

FORMAT

four	0000	0000	0000	02dch	0023h
	x index	y index	ydir,xdir	xdim	ydim

Each ten byte line represents a printed rectangle or line. The x and y indices are the distance to the lower left corner of the rectangle. In this example that point is the origin for the pattern so the index values are zero. The index direction is specified by ydir and xdir,00 for positive,2d(hex) for negative table moves. When printing a line,rectangle,or indexing,the table will move in a negative direction while the printing progresses in a positive direction on the substrates. For this reason, index directions across the print are 2d(hex) for positive,00 for negative moves. The xdim and ydim values are the dimensions of the form to be printed. Subroutine load moves the x and y index into the registers for Goto Subroutine goto, which moves ydir and xdir into temporary storage. Goto then indexes the table to the desired location using subroutines move,doit and add3. Then Load puts the xdim and ydim in registers for testing. The printing action starts in a sweep of the long axis, jogs up the short axis, sweeps back, then jogs up again untill the jog value is decremented to zero. No printing occurs in jogs. The printing sweep starts in the lower left corner of the rectangle. Often that point for the first rectangle is the origin for the entire pattern. The printing sweeps will end in the far upper right corner of the rectangle if the jog axis value divided by five (jog amount) is odd. If it is even,then the sweeps will end above the starting point

When a line is desired, the perpendicular axis coordinate is set to zero. The table position will be left at the end of the line. Index values to the next form should be calculated from that point. Lines cannot be printed in a negative print direction. Indexes to a new line must go towards the pattern origin and print moving away. The dimensions are tested to see what axis has the longer dimension then the axis with the shorter dimension becomes the jog axis. The program jumps to xjog or yjog which are nearly identical except for the axis the jog motion takes place on. Jog values have been set to 125 μm due to physical limitations of the ink jet head. Jog values could be set to any value by changing the number in subroutine jog. The shorter dimension of the form to be printed must be a multiple of the jog amount or zero.

The heart of rectangle printing is the program sections boxy or boxx. They are the same except that the jog axis is changed as mentioned before. Once in boxx or boxy the printing action will occur untill the jog axis value is decremented to zero. For this reason it is important that the smaller dimension of the rectangle be a multiple of the jog amount (5). If the program ends up somewhere totally unexpected the smaller dimension amounts should be examined. Another possible problem area is that all data are expected to be hexidecimal. This means all dimensions of a pattern to be printed must be converted to hexidecimal. The last major subroutine is Add3. The primary function of Add3 is to convert the hexidecimal data to ASCII code. The indexer board expects six place numbers for each index value. Each digit is passed as two in conformance with ASCII code. Essentially ASCII code requires a '3' prefix for numbers up to nine and a '4' prefix for hexidecimal letter-numbers. Add3 outputs the sign of the index, then two zero places (that application doe not require indices of great magnitude), and the next four places are taken from the de registers and passed in ASCII. The subroutines letck adj4 and rart are part of add3.

A microcomputer controlled switch for the the ink jet head trigger signal is located on the inverter board (Fig. A3). For it, an output line was needed that goes to a high state at the same time a printing pass was started. When completed, the output line had to return to a low state. The output line is connected to the inputs of two 'AND' gates. The external oscillator is connected to the other input. Due to bleed through of the signal,two gates were used to do the job of one, as shown in Fig. A3.

The main program had to be modified several times to find a satisfactory method of controlling the line along with the index board. The result was subroutine movel,a modified form of subroutine hand. As mentioned in the program description,the 12h port was used for handshake lines. The primary subroutine for handshaking and passing data is Hand. The indexer board accepts data when the low (refer to the timing diagram). The indexer swings the 'not' data taken line low when it has received the data. Then the microcomputer turns off the enable at which time the indexer board does a move. A minor change in the command that does that allowed two output lines to be controlled. The commands are as follows in subroutine hand.

```
mvi    a,00    ;load a register with zero
out    12h     ;reset enable
```

The change was to load 'a' with 08h instead of 00. Then the enable line is cleared while the 08h line is set to one. Thus the command t that initiates motion is the same command that turns on the head. After the move is completed, the indexer board switches the 'not' busy line high. The microcomputer turns off the head when it detects this low. Remember that all lines between the SCCS-85 board and the indexer boards are inverted (Fig. A5). A summary of the operating procedure is given in Table 10.1.

The "joysk" subroutine at the end of the main program is a new subroutine just developed. This subroutine enables joystick manual movement of the table to a desired position. The TRAP interrupt of microprocessor 8085 is being used in this system. After the interrupt button has been pushed, the microprocessor jumps to location 1024h where a jump to "joysk "command is stored. The "joysk" detects joystick movement direction data from input port B (see Fig. A5), and outputs appropriate commands to the indexer boards.

Table 10.1 Operation Procedure: for Ink Jet Printing

START MACHINE

1. Turn on main power switch.
2. Turn on +5V, +12V, +40V switches for Siemens driver board power.
3. Turn on "MOTOR" and "ENABLE" switches and set MAN/EXT switch to EXT.
4. Load substrate into brass block recess.
5. Turn on vacuum pump.
6. Turn variac to 40% for heater.
7. Wait 10-15 minutes for substrate to stabilize in temperature.
8. Turn off "internal source switch".
9. Add the MOD ink to the ink supply system and check for proper operation.
10. Switch on one of the twelve nozzle control switches.
11. Turn on X-Y Power switch.
12. Connect frequency oscillator to "external oscillator input".
13. Turn on external frequency oscilator.

14. Set frequency to 50-150 Hz.

15. Adjust the amplitude of the oscillator output to 5 volts peak-to-peak.

LOAD PROGRAM:

1. Reset the SCCS-85 board.

2. Initialize the microprocessor and instruct it to prepare to load by entering first a ´d´ and then a ´l´.

3. Download the main program from the hose computer to the SCCS-85 board.

4. Again send a ´l´ to the microprocessor.

5. Download the pattern data program from the host computer to the SCCS-85 board.

6. Instruct the microprocessor to process the pattern data and start the printing operation by typing in first ´g 1900´ and then a ´return´.

7. When the printing operation is completed, the microprocessor will send a prompt to the terminal.

APPENDIX A1

SCCS-85
USER'S MANUAL

SCCS-85

user's manual

rev 2

TABLE OF CONTENTS

PREFACE

To use the SCCS-85 to its fullest, especially when hardware reconfiguration is being made, it is strongly recommended that the user obtain a copy of the "8080/8085 User's Manual" from Intel. This manual contains detailed information on all chips used on the SCCS-85 (except standard TTL parts). Also on the recommended reading list is the "8080/8085 Assembly Language Progamming Manual" which will be useful when writing assembly language routines and programs.

1.0 INTRODUCTION

The SCCS-85 is a versatile 8085-based microcomputer system residing on a single 4.5 by 7.0 inch PC board. It may be used by itself as a powerful control computer in a variety of control applications such as peripheral I/O controllers, programmable device controllers, and the like. Or, with the addition of a keyboard, video interface and monitor it may serve as a complete microcomputer for the hobbyist, with capabilities for easy expansion as the need dictates.

By extending the SCCS-85's bus to additional cards, memory capacity can be extended to a full 65K bytes, additional I/O devices may be added as needed, the bus may be buffered to permit additional bus loading, etc.

Flexibility was the prime design goal of the SCCS-85. The card is designed so that for a particular application only those chips required need be installed. For example, in applications which do not require the 4-channel DMA controller, the two chips comprising the DMA GROUP are simply omitted from the board. If, at a later date, it is decided to add the DMA capability one need only cut two traces on the underside of the board and install the chips.

Furthermore, extensive provisions have been made in the PC board to permit configuring the I/O devices in the way which best suits an application. For example, a group of wrap-posts are provided which allow any combination of two interrupts from the SERIAL GROUP and the PIO GROUP to be combined into a single interrupt line to the 8085, thus allowing these devices to operate in various modes of interrupt-driven and/or polled I/O.

While the SCCS-85 has extensive provisions for reconfiguring the components to suit an application the PC board has also been etched so that with no modifications at all, the board is already configured to operate as a small computer communicating via RS-232 with a terminal. With no modifications, the board provides the necessary hardware to implement a real-time clock, programmable signal generator, and the like.

Lastly, the SCCS-85 has been designed with the user's pocketbook in mind. All chips used on the SCCS-85 are easily obtainable and at modest costs. I/O as well as the system bus connectors are inexpensive, easily-obtained A P Products jumper headers. These are available in either straight or right-angle configurations, and either male or female. This allows connections to be made with ribbon cables, by plugging into other boards, or even wire-wrapping to individual pins. And only those connectors actually needed have to be installed.

FEATURES

The SCCS-85 CPU is divided into seven functional groups: the CPU group, ROM group, RAM group, SERIAL I/O group, PARALLEL I/O group, TIMER group, and DMA group. While some groups, like the CPU, RAM, and ROM group are mandatory and must be present on any SCCS-85, the remaining groups are optional and need only be present on the board if the application requires it.

Following is a description of the features of each group, suggested applications, and user-definable options for that group.

** CPU GROUP. Ths SCCS-85 is based on the Intel 8085 CPU chip. This MPU was chosen because of its extensive provisions for interrupts, MPU support, and ease of interfacing with a very low chip count, plus its widespread software support at present time. The address bus is fully demultiplexed and the control signals have been decoded into individual memory-read, memory-write, I/O-read, I/O-write, and interrupt-acknowledge signals to make expansion easy.

Very flexible interrupt facilities are available. The normal 8060-type interrupt/interrupt-acknowledge protocol can be used by external peripherals which can place the RESTART instruction on the bus. Another, easier to use facility includes three separate RST lines, each of which can directly interrupt the CPU causing the CPU to jump to one of three different locations in memory, with NO other hardware required. One of these lines may be used by the TIMER group to interrupt the CPU at equal time intervals, after programmable delays, etc. Another is available for using the serial and/or parallel interfaces in interrupt mode. The third is left available on the bus for user purposes. A final interrupt line, similar to the RST lines, is the TRAP line, which causes an immediate "panic" jump to a location in memory for such purposes as power-fail sequences, hardware errors, etc.

The 8085 also provides a 1-bit intput and 1-bit output port. Two 8085 instructions allow the bits to read, and set or reset.

The 8085 may run at speeds up to 3 MHz, but if the TIMER group is present a maximum of 2 MHz is recommended, allowing the timer clock to be derived from the CPU clock. Alternativly, the timer clock may be supplied independently, allowing the 8085 to run at 3MHz, so long as sufficiently fast RAM and ROM are used. The maximum allowable memory access time is 1.5 times the CPU clock period. For example, for an 8085 running at 3MHz the CPU clock period is 333 ns, so the memory access time is 500ns. RAMs used in a 3 MHz system, then, must have an access time less that 500ns.

When power is applied to the SCCS-85 the CPU is automatically reset, and begins execution at memory location 0000h. Space is avialable on the PC board for a manual reset pushbutton, plus a signal on the bus may be pulled low to reset the CPU. A RESET line coming FROM the CPU is available for resetting other devices.

** ROM group. The SCCS-85 has room for 2K or 4K of PROM. As etched, the board accepts two 2708 EPROMs, giving 2K of ROM. If more ROM is desired pads and traces on the board are organized so that it may be reconfigured to accept two 2716 EPROMs, giving 4K of ROM.

Memory locations 0000h through 0FFFh are reserved for the on-board ROM. This constitutes the first 4K of memory. At least one ROM is required for operation of the SCCS-85, located at 0000 where execution begins when the system is powered-up.

If 2708 ROMs are used, a +12 and -5 volt supply will be needed. If the serial interface group is used along with the RS-232 drivers +12 and -12 volts will be required for them, so a 5V zener diode and resistor are provided on the board to derive the -5V from the -12V supply. If the new +5V-only 2716s are used instead, no supplies will be required for the ROMs except the normal +5V supply.

** RAM group. Up to 4K of RAM may be intstalled on the SCCS-85, in increments of 1K bytes. Each 1K block of RAM consists of a pair of 2114 1K x 4 static rams; one for the upper and one for the lower four bits of each byte. RAM addresses range from 1000h to 1FFFh, the second 4K block of memory.

Other memory-related points: ROM and RAM together occupy the first 8K of memory. The memory is fully decoded, meaning that this 8K block does not "appear" at any other memory locations within the 65K byte address space. This makes it possible to expand the SCCS-85 memory on additional boards. To make external memory decoding easy the SCCS-85 generates a MEMSEL signal on the expansion bus which indicates if a memory location currently being read or written is within the first 8K block of memory.

** TIMER group. In many control applications events occuring in real time must be monitored and/or controlled. A simple real-time clock required that the CPU be interrupted at a constant rate. Other applications may require that the time interval between two events be measured, pulses of a particular length or signals of a particular frequency be generated. The TIMER group, using an Intel 8253 peripheral timer chip, provides a flexible means of performing these and other tasks.

The 8253 contains three separate, identical 16-bit counters. Each counter may be programmed by software to operate in one of six different modes. Mode 0 allows the CPU to command the timer to interrupt the CPU after a programmed delay. Mode 1 is a programmable one-shot for generating pulses of programmable length. Mode 2 allows pulses to be generated at a programmable rate, while mode 3 generates a square wave of programmable frequency. Mode 4 produces a single pulse after a programmable delay. Mode 5 allows a hardware trigger input to initiate a programmable delay after which a single pulse is produced.

Each counter may be read at any time to ascertain the contents of the counter. Thus, if the timer's clock input is supplied externally the result is a hardware event counter which the software can read and modify at any time. Furthermore, even if the timer's clock inputs are supplied on board, an individual enable input for each is available to allow external hardware to enable counting.

As configured, one timer's output is connected to the RST7.5 input of the 8085. Thus, the output of the timer is used to interrupt the CPU at a programmable rate, after a programmable delay, etc. Another timer's output serves as the baud rate generator for the serial interface group, thus allowing a fully-programmable baud rate for serial I/O. The third timer's output is available for external use at connector J3.

If other configurations are desired jumpers in P1 may be changed to allow each timer's clock and enable input and each timer's output to be connected as desired.

** PIO group. An Intel 8255 parallel I/O interface chip provides 24 lines of parallel I/O for user applications. The chip is programmable in several different modes including 24 lines of basic input/output, one or two strobed 8-bit I/O ports with handshaking and interrupt control lines, strobed bi-directional 8-bit bus with 5 control lines and interrupts, or combinations of the above. The SCCS-85 allows the PIO group to be handled under programmed control, interrupt control, or a combination of the two.

All 24 I/O lines plus GND and +5V are available at connector J2.

** SERIAL I/O group. The Intel 8251 USART chip provides a programmable choice of synchronous or asynchronous I/O. Synchronous serial I/O is useful for such applications as using the SCCS-85 as an intelligent tape controller where data is recorded as a combination of both clock and data. The 8251 can be commanded to search for the sync byte/s which precede the data, as in IBM's bi-sync format.

A more common application is using the 8251 as a serial I/O port connected to a terminal or another computer. For these applications RS-232 drivers and receivers are provided on the SCCS-85 for the transmit and receive data lines, plus the modem-control lines $\overline{CTS}$, $\overline{RTS}$, $\overline{DTR}$, and $\overline{DSR}$. As configured, the modem control inputs $\overline{DSR}$ and $\overline{CTS}$ inputs are disabled, as most terminals do not support them. They may be easily enabled for use on modems by simply cutting two option traces on the board.

Using the timer group as the baud rate generator, baud rates from less than one bit per second to 9600 may be programed. If a 3.5795 MHz crystal (color-burst) is used on the SCCS-85 baud rates of 19,200 and 38,400 are also available.

Since the RS-232 lines usually use a standard 25-pin D (delta) connector, special provisions have been made on the SCCS-85 PC board for mounting one of these connectors very easily. The most readily available 25-pin D connectors are the type with the solder-cup pins. By apparent coincidence the two rows of pins on these connectors are 1/16th inch apart, just the thickness of the SCCS-85 PC board. Taking advantage of this fact, wide PC traces extend to the extreme edge of the board in a pattern that exactly aligns with the pins on a 25-pin D connector. Thus, a connector may be slipped onto the edge of the PC board and its pins soldered directly to the PC traces, thereby wiring the connector with the standard pinout and rigidly mounting it.

If the D connector must be mounted external to the PC board, connector J6a is provided. Since D connectors are available which crimp directly to a 25-pin ribbon cable which in turn crimps to a standard female ribbon cable connector, the pinout of J6a is such that the result is a correctly wired D connector.

** DMA group. Some applications require that blocks of data be rapidly transferred directly from memory to a peripheral and vice-versa. Examples include disks and CRT controller chips such as the Intel 8275. For these applications the DMA group with its 8257 provides four separate channels of DMA (direct memory access) for supporting up to four DMA peripherals. Having the DMA controller on the SCCS-85 board makes it as easy to add a DMA peripheral device to the system as a non-DMA peripheral.

All DMA request and grant lines, as well as the terminal count line (indicating when a DMA transfer is complete), are available at connector J5.

OTHER FEATURES:

** The PC board is designed to be mounted in one of two methods. As supplied, the SCCS-85 board is 4.5 x 7.0 inches. Four mounting holes are provided near the corners for mounting the board, adding rubber feet, or mounting a protective plexiglas cover above and/or below the board for protection. In addition, four large pads are provided for connecting a power supply cable directly to the board.

In applications where the system is to be expanded with a mother board, 1/4-inch is sheared from each end of the board. Now, a right-angle AP connector is installed in the bus connector location and the board can be perpendicularly-plugged into a mating AP female connector on a mother board, which now supplies the power to the board through the bus.

** Consideration has been given to making the board as easy to assemble as possible. The PC board is rather dense, yet nearly all conductors-between-IC-pins are on the top side of the board to minimize the possibility of solder bridges. Pin 1 of each IC is identified on the top and bottom sides of the board, and all holes are plated-through. Both sides of the board have a solder mask, and the top is silk-screened with a component placement "road-map".

** For making more extensive modifications to the SCCS-85 a spare 16-pin IC pattern is provided. This allows an additional IC to be mounted on the board and used for any desired purpose.

2. ASSEMBLY

PARTS LIST

CPU GROUP & MISC. (mandatory)

U1	8085 CPU	
U2	74LS373	8-bit tri-state latch
U3	74LS257	demultiplexor
U4,U19	74LS138	decoder
U17	74LS54	quad AOI gate
U18	74LS02	quad nor gate
R1	1K	resistor
R2	75 ohm 1	watt (see ROM GROUP assembly directions)
R3	1K	resistor
C2	20pF	ceramic disk capacitor
C3,C4	100uF	tantalum capacitor
C5-C8, C17, C18	1uF 35V	tantalum capacitor
D1	1N4733	5V zener diode, 1 watt
D2	1N4001	diode
X1		crystal 3.57 (color burst) or 4 MHz (see text)
SW1	SPST N.O.	pushbutton switch (optional) reset switch
J1		50-pin connector J1 (see text)
C9-C16, C19, C20	0.1uF	ceramic disc or monolithic capacitor

ROM GROUP (mandatory)

U5,U6 2708 (or 2716 if board altered)
NOTE: U5 is optional

RAM GROUP (1K mandatory, other 3K optional)

U7-U14 2114,2114L, etc. 1K x 4 static ram, 450ns or less
- 300ns 2114L chips highly-recommended
U10,U14 - 1st 1K of RAM
U9, U13 - 2nd 1K of RAM
U8, U12 - 3rd 1K of RAM
U7, U11 - 4th 1K of RAM

TIMER GROUP (optional)

U20 8253 Intel 3-channel timer chip
J3 or J4 10-pin connector J3 or 40-pin connector J4

PIO GROUP (optional)

U21 8255 Intel programmable peripheral interface
J2 or J4 26-pin connector for J2 or 40-pin connector for J4

POWER SUPPLY

Board requires +5V regulated plus or minus 5%; if 3-voltage EPROMS or RS-232 options are used, also requires plus and minus 12V at 150 ma regulated to 10%. Five-volt supply current is typically 1.25 A for fully-populated SCCS-85.

SERIAL I/O GROUP (optional)

U22	8251	Intel USART
U23	1488	quad RS-232 line driv
U24	1489	quad RS-232 line receiver
J6a or J6b		26-pin connector J6a or 25-pin delta connector for J6b

DMA GROUP (optional)

U16	8257	Intel 4-channel DMA controller
U15	74LS373	8-bit tri-state latch
J5		10-pin connector J5

NOTE: Because of the high component density of the SCCS-85 PC board successful assembly requires some degree of expertise in the soldering of many very small connections. A low-wattage pencil-type soldering iron and fine rosin-core solder is a must! If you feel that your expertise or equipment are not up to the task and cannot enlist the help of a friend, please do us both a favor and return the board for a refund.

PRELIMINARY COMMENTS ON ASSEMBLY

It is advised when assembling and bringing up an SCCS-85 system for the first time that the board be assembled in the minimum configuration with no hardware reconfigurations made. This means installing the CPU group, one 2708 with the SCCS-85 Monitor, 1K bytes of RAM, the Timer Group, and Serial Group with RS-232 interface. This will simplify debugging the system should it become necessary.

Note that by simply installing the above components and making no other changes the SCCS-85 can be connected to a terminal and be operated with its operating system.

Before beginning assembly it is a good idea to inspect the board for any flaws in manufacturing. They are much easier to find now than after the board is assembled, and IF NO SOLDERING HAS BEEN DONE ON THE BOARD a defective board may be returned for replacement. Look for shorted or broken traces by holding the board up to a bright light.

It is generally recommended that sockets be used for all ICs on the board to facilitate replacement should it ever become necessary. However, some applications requiring very high tolerance to vibration or corrosive and/or dirty environments may be best served by first testing and burning-in chips, then soldering them directly to the board. In all cases sockets will probably be used for the proms, and all sockets should be of a high quality. A recommended type (sold by James Electronics) have pins which contact the FLAT SIDE of the IC pins over a broad surface area, and plugging & unplugging the IC seems to result in less damage to the IC socket than the TI low-profile sockets which contact the ragged EDGES of the IC pins. Gold-plated IC sockets are probably a good idea in hostile environments though not absolutely necessary in most other applications.

When installing IC sockets, be sure to note that the pin-1 designation on the socket (most have them) is oriented properly on the PC board.

In other critical applications where the board may be subject to repeated flexing (such as plugging into and out of a mother board) a further precaution is sometimes taken to insure the integrity of through-plated holes. After all components have been soldered in place (IC sockets, etc.) all remaining feed-through holes are filled with a small amount of solder. This can be done from the bottom side of the board, although a bit of practice is recommended to judge how much solder to put in each hole where you can't see the other side. In general, filling the feed-throughs is not needed, although it doesn't take very long to do and may enhance one's peace of mind.

It is probably best that sockets not be installed where options are being omitted (e.g. U15 and U16 when the DMA option is not installed). If the option is later installed a new socket installed then would be preferable to one which has been accumulating dirt and corrosion for a period of time.

After the board has been assembled but before the ICs are installed, you may want to de-flux the bottom side of the board. This is probably only a cosmetic improvement and is not recommended unless ALL feed-through holes have been filled. Even then, extreme caution must be exercised to prevent the defluxing solution from getting into the IC sockets on the top side, a virtual disaster since when it evaporates it will leave a small film of flux on the pins.

The remainder of the assembly is categorized into functional groups.

CPU GROUP

The CPU group is mandatory.

STEP 1:

Install:

- ☐ C3,C4 — 100uF tantalum capacitor. OBSERVE POLARITY!
- ☐ R1,R3 — 1K ohm, 1/4 watt resistor
- ☐ D2 — 1N4001 diode or equiv. OBSERVE POLARITY
- ☐ SW1 — momentary contact N.O. pushbutton switch for RESET

STEP 2:

It is recommended that the 8085 clock be crystal controlled. It is mandatory that a crystal be used IF:

1. The 8253 timer chip (U20) is installed and requires that it's input clock (which comes from the CPU) be accurate.
2. The 8251 USART chip (U22) is installed and its clock is derived from the 8253 timer chip (standard) or derived directly from the CPU clk signal CLK (see option under SERIAL I/O GROUP).

The 8085 itself may be run at speeds up to 3MHz (using a 6MHz crystal) but care must be taken that at speeds higher than 2MHz the other components on the board will also be able to run that fast. Standard Intel parts will meet specs up to 3MHz with 300ns RAMs and EPROMs, but one may have little information on parts from second-source manufacturers, so it may prove less of a problem to limit yourself to 2MHz. Furthermore, while the 8253 timer chip can handle bus accesses at 3MHz its clock input (which comes from the CPU clock on the SCCS-85) is limited to 2MHz, so running the CPU faster than that would require that the timer chip be supplied with its own clock.

It is therefore recommended that the CPU be run at 2MHz or slower.

To provide a crystal controlled clock, install:

- ☐ X1 — 3.5795 (color burst) or 4 MHz crystal
- ☐ C2 — 20pF ceramic capacitor (do not install C1)

If maintaining a precise CPU clock frequency is not required, and substantial drift of frequency is not objectionable, the expense of the crystal may be eliminated by installing

- ☐ install 10K resistor in place of X1
- ☐ install 20pF capacitor in C1
- ☐ omit C2

This information is from Intel literature and has not been tested as yet. The above values will cause the 8085 to run at approx. 3 MHz; somewhat faster that the 2 MHz rate obtained with a 4 MHz crystal. Note that using this technique of driving the 8085 clock will required that any serial communications chips have their own crystal-controlled clock, hence it is felt that this option has little to recommend it.

STEP 3:

Install:

- ☐ 40-pin IC socket for U1'
- ☐ 20-pin IC socket for U2
- ☐ 16-pin sockets for U3, U19 respectively
- ☐ 14-pin sockets for U17, U18 respectively

--->> NOTE: DO NOT INSTALL ICs IN SOCKETS AT THIS TIME <<---

Install:

- ☐ C5, C6, C7, C8, C17, C18 — 1uF 35V tantalum capacitors. Do not substitute non-tantalum capacitors! OBSERVE POLARITY!
- ☐ P3 — 6-pin double-row header (optional - see section on INTERRUPTS under HARDWARE ENGINEERING)
- ☐ C9-C16, C19, C20 — 0.1 uF ceramic disc or monolithic capacitor

This completes assembly of the CPU group.

ROM GROUP

One 2708 or 2716 EPROM is required (U6). The second (U5) is optional. BOTH ROMs must be either 2708s or 2716s; no mixing possible.

As etched, the SCCS-85 board will accept one or two 2708 EPROMs. Alternatively, one of two options may be chosen: one or two 5V only (Intel) 2716 EPROMs; or one or two three-voltage 2716 EPROMs.

2708 EPROM installation

To utilize 2708 EPROMs, for which the board has been etched, do the following:

STEP 1:

Install:

- ☐ 24-pin IC socket for U6 (mandatory).
- ☐ 24-pin IC socket for U5 (optional - install if two proms will be used)

☐ 16-pin IC socket for U4

--->> DO NOT INSTALL ICS IN SOCKETS AT THIS TIME <<---

NOTE: It is recommended that high-reliability sockets be used for U6 and U5 since they will be inserted and removed often. Better still would be zero insertion force sockets.

Step 2:

Install:

☐ 75 ohm 1 watt resistor (see note below)
☐ 5.1 volt 1 watt zener diode (1N4733)

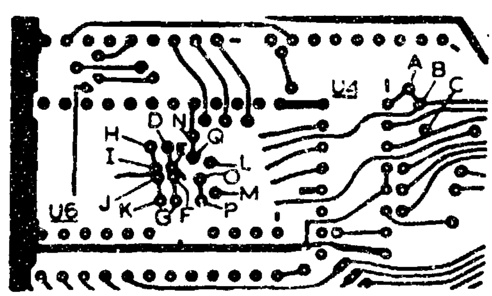
NOTE: If you will never use more than one EPROM you may substitute a 120 ohm 1/2 watt resistor for R2. Alternatively, if you can guarantee that the total current draw from both EPROMs on the -5V supply is 60 ma or less then you may substitute a 100 ohm 1/2 watt resistor for R2.

This completes assembly of the 2708 ROM GROUP.

Three-voltage 2716 EPROM installation

To utilize 2716 EPROMs which require +5, -5, and +12V supplies (e.g. TMS2716, Motorola 2716, etc.) perform the following modifications and assembly. (Refer to dwng. no. 2 of schematics.)

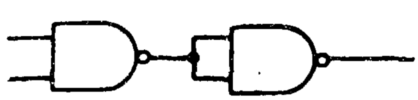
Step 1 On the bottom side of the board find the area of the PC pattern near U4 and U6 shown at right. In this figure, three pads near pin 1 of U4 have been labelled A, B, and C. Cut the trace between pads A and B.



Step 2 Install a 74LS00 IC in the SPARE IC pattern on the board. Be sure to connect +5V and GND to the chip using the conveniently located traces on the bottom side of the board.

Step 3 Using short lengths of wire-wrap wire use two of the gates in the 74LS00 chip to add the following circuit to the board:

from pad B
from pad C



to pad A

Step 4 On the bottom side of the PC board find the area under U6 shown above. In the figure above pads have been labelled D through Q. Cut the trace between pads O and P. Also cut the trace between pads N and Q. Jumper pads N, L, and P together.

Step 5 Cut the trace between pads H and I. Also cut the trace between pads J and K. Jumper pad H to Q. Also jumper pad K to M.

Step 6 Install IC sockets for U4, U5, and U6 exactly as in Step 1 under "2708 EPROM installation" above. DO NOT INSTALL IC CHIPS YET.

Step 7 Install D1 and R2 exactly as in Step 2 under "2708 EPROM installation" above.

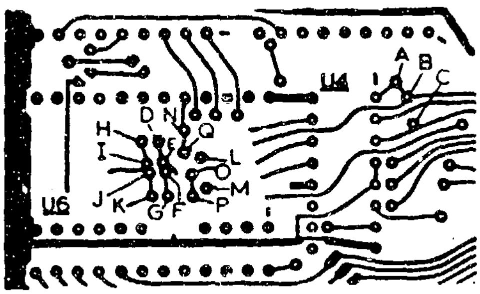
This completes the assembly of the three-voltage 2716 EPROM option.

Single-voltage 2716 EPROM installation

To utilize single-voltage (5V-only) 2716 EPROMs such as the Intel 2716 perform the following modifications and assembly:

Step 1 Perform steps 1 through 3 under "Three-voltage 2716 EPROM installation" above.

Step 2 The figure at right shows the PC board area under IC U6. In the figure pads have been labelled D through Q. Cut the trace between pads D and E. Also cut the trace between pads F and G. Jumper pads D, G, and L together.



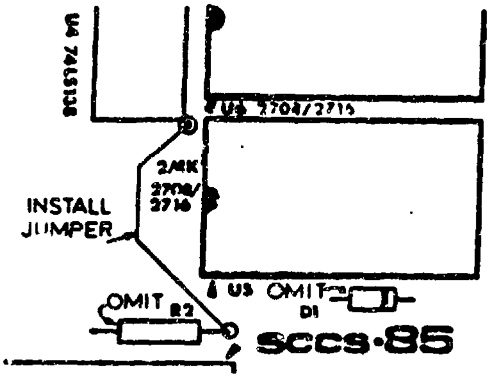
Step 3

Cut the trace between pads O and P. Jumper pads P and M together.

Step 4 Install IC sockets for ICs U4, U5, and U6 exactly as in step 1 under "2708 EPROM installation" above. DO NOT INSTALL CHIPS YET.

Step 5 DO NOT install R2 or D1. Instead, on the top side of the PC board install a jumper from the pad for R2 which is closest to pin 1 of J1, to the pad circled in white near pin 9 of IC U4. See figure at right.

This completes installation of the single-voltage EPROM option.



RAM GROUP

The SCCS-85 board can support up to 4K bytes of on-board RAM. The RAM is made up of 2114 static RAMs which are organized as 1K by 4 bits wide, thus allowing RAM to be expanded in increments of 1K bytes.

For an SCCS-85 running at 2 MHz (with 4 MHz crystal) RAMs with 450ns access times are satisfactory. For faster CPU clock rates, e.g. 3MHz, 300ns RAMs will be needed, and are recommended for good reliability margins. In all cases low-power RAMs are not absolutely necessary, but are highly recommended, as they not only reduce power consumption of the board, but also generate less noise.

RAM GROUP INSTALLATION:

For K bytes of RAM, install

- ☐ 18-pin IC socket for U10
- ☐ 18-pin IC socket for U14

--->> NOTE: DO NOT INSTALL ICs IN SOCKETS AT THIS TIME <<---

For 2K bytes of RAM, also install

- ☐ 18-pin IC socket for U9
- ☐ 18-pin IC socket for U13

For 3K bytes of RAM, also install

- ☐ 18-pin IC socket for U8
- ☐ 18-pin IC socket for U12

For 4K bytes of RAM, also install

- ☐ 18-pin IC socket for U7
- ☐ 18-pin IC socket for U11

This completes assembly of the RAM group.

TIMER GROUP

The timer group is optional.

As wired the 8253 timer receives its clock signal from the CPU clock. Since the maximum clock frequency for the 8253 is 2MHz using the 8253 will require that the CPU clock not be faster than 2MHz. If it is then the timer group will have to be slightly reconfigured to allow input of a separate clock signal. For details on this see the section on Hardware Engineering.

Timer Group Installation

Install:

- ☐ 24-pin IC socket for U20

If extensive reconfiguration is anticipated (see Hardware Engineering) install:

- ☐ 16-pin double-row header in P1

If connections to the timer group must be made from off-board, install:

- ☐ 10-pin double-row header in J3 or, alternatively, 40-pin double-row header in J4 (see section on Connectors)

This completes assembly of the timer group. See Hardware Engineering section for suggestions on reconfiguring timer group.

PARALLEL I/O GROUP

The parallel I/O group is optional.

To add the parallel I/O group install:

- ☐ 40-pin IC socket for U21

---->> WARNING <<----

When installing the socket for U21, the PIO chip, and when installing the chip itself, note that pin 1 of this chip will be oriented in the opposite direction from nearly all other chips. Plugging the PIO chip in backwards would likely destroy it!

Install:

☐ 26-pin double-row header in J2 or, alternatively,
40-pin double-row header in J4 (see section on Connectors)

This completes assembly of the Parallel I/O Group.

SERIAL I/O GROUP

This group is optional.

The serial I/O group consists of the 8251 serial communications IC, and optional RS-232 driver and receiver chips.

As configured, the 8251 USART receives its baud rate clock from the timer group. This allows the baud rates to be fully software-controllable, as well as minimizing chip count. Thus, as configured, installing the serial I/O group will necessitate that the timer group be present. If desired, however, the 8251 USART may be supplied separately. See Hardware Engineering.

The 8251 USART may also operate in a synchronous rather than asynchronous mode. This would be used, for example, if the SCCS-85 were a dedicated digital mag. tape drive controller. The serial data to and from the USART would be interfaced to the tape head, requiring different driver chips than the 1488/1489 duo used here. Most likely then, U23 and U24 would be omitted and connections made directly to appropriate pads. Since the vast majority of users will use the serial port for normal RS-232 communications with terminals and the like, a detailed discussion of other configurations is beyond the scope of this manual.

To add the serial I/O group install:

☐ 28-pin socket for U22
☐ 14-pin sockets for U23, U24 (if being used)

If the RS-232 signals will be taken off-board through a ribbon cable, install:

☐ 26-pin double-row header for J6a

If the RS-232 cable is to plug directly onto the PC board, install:

☐ 25-pin female Delta, or "D" connector. (See mounting instructions in section on Connectors.)

This completes assembly of the Serial I/O Group.

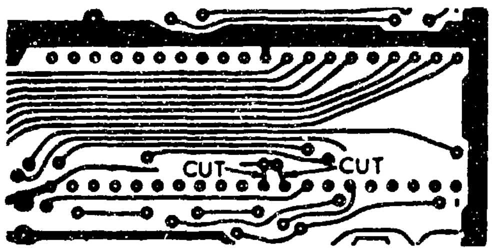
DMA GROUP

The DMA group is optional.

The on-board DMA group, when installed, allows external I/O peripherals to be as easily designed for DMA operation as the more common program-controlled technique.

To add the DMA group, perform the following modifications and assembly:

Step 1 On the bottom side of the PC board find the area under U16 shown at right. Extending from pins 9 and 10 of U16 are two short traces connecting both to another pair of connected pads. Cut these two short traces as shown to enable the DMA group.



NOTE: If the DMA option is later removed (chips U15 and U16 removed) these traces MUST be re-installed or the SCCS-85 will not operate.

Step 2

Install:

- ☐ 40-pin IC socket for U16
- ☐ 20-pin IC socket for U15

DO NOT INSTALL CHIPS YET

WARNING: Some 8085 chips with early date codes have a bug in them which prevents correct DMA operation. The symptoms are that a DMA cycle may affect the flags in the 8085 flag register. The DMA transfer itself operates correctly, but the program being executed at the time will have unpredictable results.

This completes assembly of the DMA Group.

FINAL ASSEMBLY

The assembly of the SCCS-85 is now nearly finished.

If interrupts from either the 8251 USART or the 8255 parallel I/O chip are to be used, install:

- ☐ 12-pin double-row header in P2

For directions on configuring P2 for interrupts see section on Interrupts in Hardware Engineering.

If the SCCS-85 bus is to be extended to other cards, install:

☐ 50-pin double-row header in J1

For suggestions on different connector options, see section on Connectors.

If the SCCS-85 will be receiving its power through the provided pads near the lower edge of the board, install:

☐ 4-conductor power cable for +5, +12, -12, and GND. Connect to indicated pads.

Before installing the ICs apply power to the board. With a VOM meter check for the following voltages:

☐ 0V at pin 20 of U1
☐ 5V at pin 40 of U1

If 2708 or 3-voltage 2716 EPROMS are being used, check for
☐ -5V at pin 21 of U6
☐ +12V at pin 19 of U6

If 5V-only EPROMs are being used, check for
☐ +5V at pin 21 of U6

If the RS-232 driver is being used, check for
☐ +12V at pin 14 of U23
☐ -12V at pin 1 of U23

IF THE ABOVE VOLTAGES ARE NOT ALL CORRECT DO NOT PROCEED UNTIL THE SOURCE OF THE PROBLEM IS FOUND AND CORRECTED!

With all other assembly completed and any reconfigurations made, install the ICs in their proper sockets. Be absolutely sure that the IC is properly oriented with respect to pin 1, ESPECIALLY IC U21, which is reversed relative to the others. Pin 1 is indicated by a white arrow on the silkscreened top of the board. It is also indicated by a "half-moon" in each silkscreened IC locator box.

This completes assembly of the SCCS-85 board! If an EPROM containing the SCCS-85 monitor is being used refer to the manual for the monitor for a simple program which can be entered into the SCCS-85 for demonstration purposes.

3. HARDWARE ENGINEERING

This chapter contains suggestions for reconfiguring various component groups on the SCCS-85 to fit your particular application.

CPU CLOCK RATE

The 8085 CPU may be operated at clock rates up to 3MHz, although other restrictions may make 2MHz a more practical upper limit. Since this selection must be made at time of assembly it is covered fully in the CPU GROUP section of chapter 2, ASSEMBLY.

ROM SELECTION

The SCCS-85 is etched to accept 2708 EPROMs without alteration. However, with a minimum of patching either single-supply or triple-supply 2716 EPROMs may be used, thereby doubling the ROM capacity of the board.

Like the CPU CLOCK RATE, the ROM SELECTION is best done at time of assembly. However, the change to 2716 EPROMs can be made even after all parts have been installed. For directions on making the reconfiguration see the ROM GROUP section of chapter 2, ASSEMBLY.

RAM OPTIONS

In order for the 8085 to have access to a stack at least 1K bytes of RAM will have to be installed on the SCCS-85. Since it is most likely that users expanding their RAM will want to do so into successively higher memory locations, then the following table for RAM expansion should be followed:

For 1K bytes of RAM, locations 1000H to 13FFH, use RAMs U10 and U14.

For 2K bytes of RAM, locations 1000H to 17FFH, use RAMs U9, U10, U13, and U14.

For 3K bytes of RAM, locations 1000H to 1BFFH, use RAMs U8, U9, U10, U12, U13, and U14.

For 4K bytes of RAM, locations 1000H to 1FFFH, use RAMs U7 through U14.

TIMER GROUP OPTIONS

The TIMER GROUP is probably one of the most versatile components of the SCCS-85. Much of the SCCS-85's flexibility in adapting to dedicated control applications stems from the power of the TIMER GROUP. Since the 8253 timer contains three completely independent and identical timer/counters, and each can be individually programmed to operate in one of six modes of pulse generation, square-wave generation, delay timing, event counting, and the like, nearly endless applications can be easily accommodated.

Each of the three timer/counters in the 8253 has its own clock input, gate input, and output line. The clock input provides the events (level transitions) which the chip's counters count, while the gate input allows the clock input to be enabled or disabled. Depending on the mode the timer has been programmed with, the output will then provide the appropriate signal such as a continuous square wave of programmable frequency, a pulse train of programmable rate, a single pulse of programmable length, a single pulse at the end of a programmable delay, and so on.

The flexibility of the 8253 itself is enhanced by the SCCS-85's provisions for supplying the clock and gate inputs from different sources, as well routing the outputs to various places for use.

Nearly all of the clock, gate, and output signals for the 8253 are routed through the double-row connector pattern P1. Throughout this discussion refer to dwng. no. 4 of the schematics (Timer Group, I/O address decoder). Here it can be seen that nearly all of the 8253 clock, gate, and output pins are connected to the side of P1 closest to the 8253 chip itself (both on the schematic and on the PC board as well). The various signal sources and output destinations are connected to the other side of P1. Thus, most all hardware configurations for the three timers can be made by proper connections between these two rows of pins.

In keeping with the design philosophy of the SCCS-85 the timer group comes pre-configured in a logical structure so that with no modifications the timer group will function in a way that will suit many applications. Looking on the bottom side of the PC board between the two rows of pins of P1 can be seen the seven traces which define this configuration, and which can easily be cut later if reconfiguration is desired. Following is a discussion of that configuration along with suggested applications.

Timer 0

Looking at the schematic it can be seen that timer 0 is configured to receive its clock from the 2MHz CPU clock. This signal passes from pin 15 to pin 16 of P1 as shown by the dotted line on the schematic. If a different source for CLK0 is desired the trace between pins 15 and 16 would be cut, and the new clock connected to pin 16. The gate for timer 0 passes through pins 11 and 12 to Vcc, hence it is enabled all the time. The output for timer 0 passes through pins 13 and 14 to pad B, which is connected by cuttable traces to pads A and C. Pad A leads to connector J3 pin 3, while pad C is the RST7.5 interrupt input to the 8085. Thus, as configured, timer 0 may serve two different purposes. In the first, the 8085 program would enable the RST7.5 interrupt input and the output of timer 0 would then interrupt the 8085 on each rising edge of the output. Hence, with the six possible modes timer 0 can be used in, the CPU can be interrupted: 1. at a constant rate (real time clock); 2. after a programmed delay; 3. at the end of a programmed delay initiated by either a software or hardware trigger. The second use of timer 0 would be to simply provide its output at connector J3 pin 3. RST7.5 interrupts should then be masked in the 8085.

Timer 1

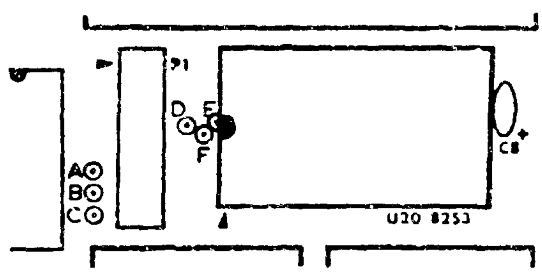
As configured timer 1 also gets the 2MHz CPU clock, though it comes through pads D-E rather than through P1. The gate for timer 1 is permanently tied TRUE. The output of timer 1 passes through pins 7-8 of P1 then on to the SERIAL I/O group where it serves as the baud rate clock for the USART chip. This allows the baud rate to be fully programmable by software. For suggestions on progamming timer 1 for various standard baud rates see chapter 4 on Software Engineering.

Timer 2

As configured timer 2 receives the 2MHz CPU clock through pins 1-2 of P1. Its gate is tied TRUE through pins 5-6. The output of timer 2 is connected, through pins 3-4 of P1, to connector J3 pin 2 for whatever use is desired. An example might be to buffer the output through a transistor to drive a small loudspeaker for generating beeps, tunes, and the like.

RECONFIGURATION

Reconfiguration will generally involve cutting one or more traces under P1, or possibly at pad groups A-B-C or D-E-F. (Shown at right.) If changes will be made at P1 it is suggested that a 16-pin double-row header be installed at P1. This will allow connections to be easily and reliably made by wire-wrapping to the pins on this header.



At this point possible reconfigurations should be apparent. If it is desired to supply any of the timers with clocks other than the CPU clock, one need only cut the appropriate trace under P1 (or trace D-E in the case of timer 1) and connect the desired clock. Note that if the clock is originating off the board, pin 1 of J3 has been conveniently routed to pin 9 of P1 for the user to connect to whatever he pleases.

If very long timing periods are desired, a seperate low-frequency clock can be supplied to a timer. Alternatively, two timers may be cascaded by connecting the output of one to the clock of another. The first timer would then be programmed to operate as a rate generator (a divide by N circuit) to supply a programmable frequency to the next.

It should be noted that the clocks need not be continuous square wave signals. The timers themselves merely count falling transitions on the clock inputs. Thus, if a clock is supplied externally from a device which produces a pulse in coincidence with some event, then the timer can serve as an event counter. The 8253 will allow the count register to be read by the program at any time to determine the current count of events. Or, with the various modes the timer may be programmed into, such things as "interrupt after N events", "interrupt after N events following a hardware strobe on the gate input", "interrupt every N events", etc. are easy to implement.

Since the use of the timers can involve interrupts, see also the discussion of interrupts in chapters 3 and 4.

INTERRUPTS

The 8085 has extensive provisions for using interrupts. As on the 8080A the 8085 has an INTR line which may be pulled high to initiate an interrupt sequence. On the first machine cycle of the next instruction the $\overline{INTA}$ (interrupt acknowledge) signal will be sent, informing the interrupting device that it should place its interrupt vector on the bus, after which the 8085 will call to one of eight memory locations.

For purposes of using interrupts from peripherals on the SCCS-85 board, (e.g.the 8251, 8253, or 8255) the 8085 also provides another mechanism for generating interrupts. Three inputs to the 8085 chip, the RST5.5, RST6.5, and RST7.5 inputs, will each cause an interrupt vector to a specific memory location when pulled high, WITH NO OTHER HARDWARE NECESSARY. Furthermore, two of these inputs, the RST5.5 and RST6.5 are LEVEL SENSITIVE, meaning that an interrupt will be maintained as long as the input is held high. This is used for things like a USART which provides a RECEIVE BUFFER FULL signal which can be used to interrupt the 8085 when a character is received. Here, the interrupt is held until it is serviced by the software which reads the received character, thereby resetting the flag and the interrupt request.

The other input, the RST7.5 input, is EDGE sensitive meaning that if the input is pulled high and held there indefinitely, only ONE interrupt will be recognized - it is the low-to-high transition which generates the interrupt. To generate another interrupt the input must go low then go high again. This input is useful for things like making a real time clock. A square wave of desired frequency is simply tied to the RST7.5 input, then on each rising edge an interrupt will be generated.

Provisions have been made on the SCCS-85 for configuring RST interrupts in any desired fashion. As manufactured two of the interrupt inputs, the RST7.5 and RST6.5 inputs have been pre-configured in a way which should be adequate in most applications. This configuration is described below, along with suggestions for re-configuring for other applications.

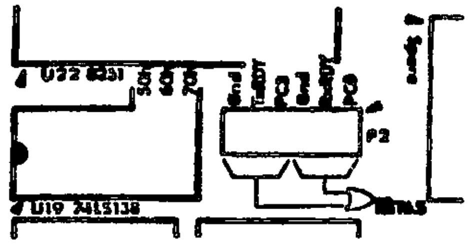
RST7.5

As described above under TIMER GROUP the RST7.5 input is connected to the output of timer 0. Thus with no alterations this timer can be used for such things as "interrupt at an N Hz frequency (real time clock)", "interrupt after N clock cycles (programmable delay)", or in general, interrupt according to some programmable time function.

If it is desired to use the RST7.5 interrupt input for some other purpose this can be done in different ways. For example, if the interrupt will be supplied from some other point on the PC board, cut the trace between pads B and C shown above, and connect the signal to pad C. (Refer also to dwng. no. 4 of the schematics.)

If the interrupt signal will be supplied from off the board, cutting the trace between pins 13 and 14 of P1 will leave the RST7.5 input connected to connector J3 pin 3.

RST6.5



Referring to dwng. no. 5 of the schematics it can be seen that an INTERRUPT SELECTOR GROUP has been provided to allow two interrupt signals to be OR-ed together to generate the RST6.5 interrupt input. Selecting two of four possible signals can be done at the double-row header P2. (See figure at right.) The four signals are RxRDY and TxRDY from the 8251 USART and PC0 and PC3 from the 8255 PIO chip. Also present at P2 are two grounded pins, allowing one or both of the interrupt inputs to be tied inactive. Note on the schematic the dotted lines indicating that the PC board is etched with both interrupt inputs grounded. Hence, to allow one or two of the four interrupt signals it will be necessary to cut one or both of the traces 5-6 or 11-12 under P2 and connect the desired interrupt signal to pads 2-4-6 and/or pads 8-10-12.

Of course it is also possible to use interrupt signals other than the four mentioned above. Just cut trace 5-6 or 11-12 and connect the desired interrupt signal to pads 2-4-6 or 8-10-12 as above.

OTHER INTERRUPTS

Three other interrupt inputs to the 8085 are available for user purposes. These are RST5.5, INTR, and TRAP. All three of these signals pass through double-row header P3 where, as pre-configured, these inputs are all tied low by short traces on the bottom side of the PC board under P3. (See dwng. no. 1 of the schematics.) These three inputs are available on the bus at J1 for connection to other boards in the system, but this by no means rules out using any of them on the board. Just cut the trace under P3 to enable the desired input, and connect the interrupt signal to pad 2, 4, or 6, depending on the interrupt desired.

Note when using the INTR input that the bus on the SCCS-85 DOES NOT float to the high level. This means that the "trick" used on some systems of letting the floating bus put a RST 7 instruction on the bus will not work.

SERIAL I/O

In nearly all applications the SERIAL I/O group, when used, will be used for RS-232 communications with other devices.

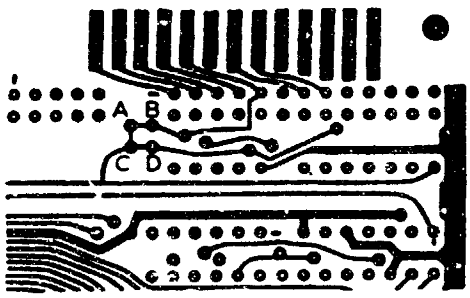
Some devices, modems in particular, make use of a number of device-control and handshaking signals in the RS-232 definition, so the SCCS-85 has been designed to support these signals. Such signals include REQUEST-TO-SEND, DATA-TERMINAL-READY, DATA-SET-READY, and CLEAR-TO-SEND. Two of these are inputs while two are outputs.

The two outputs, $\overline{DTR}$ and $\overline{RTS}$ are GENERAL PURPOSE 1-BIT OUTPUT PORTs and as such can be used for any purpose the user desires, such as powering-up a printer, etc.

Of the two inputs, one is a GENERAL PURPOSE 1-BIT INPUT PORT, the $\overline{DSR}$. Its status can be read at any time by the CPU and has no effect on the transmission or reception of data.

The other input, $\overline{CTS}$ is a clear-to-send input which must be low (TRUE) to enable the 8251 transmitter. Transmission will stop on the next character if this pin goes high.

Since not all devices will support the $\overline{DSR}$ and $\overline{CTS}$ inputs, the SCCS-85 has these two inputs tied TRUE on the RS-232 side of the 1489 receiver chip. (See dwng. no. 6 of the schematics.) If your device supports either of these signals be sure to cut the appropriate traces on the bottom side of the board to enable the input. The trace between pads A-B (shown at right) must be cut to enable $\overline{CTS}$, while C-D must be cut to enable $\overline{DSR}$.



Two options exist for connecting the external device to the PC board. These options are covered in the section on CONNECTORS below.

It is possible to supply the 8251 with a baud rate clock other that from the timer group. This is most easily done by cutting the trace between pins 7 and 8 of P1, then connecting the new clock signal to pin 7 of P1.

No provisions have been made for supporting a current-loop interface on the SCCS-85.

SOD, SID LINES ON 8085

The 8085 chip itself provides a 1-bit input and 1-bit output port, called SID (serial-in data) and SOD (serial-out data) respectively.

These lines are present on connector J3, pins 7 and 8 for user purposes. Refer to the 8080/8085 Assembly Language Reference Manual and 8085 User's Manual for information on reading and setting these lines.

BUS EXTENSION

The SCCS-85 on-card bus is available at connector J1 for expansion to other boards.

Chapter 5 contains a Bus Signal Definition table which describes each of the signals on J1.

For the most part, the signals on the bus are unbuffered, direct connections to the 8085 and other chips. Hence, care must be taken when expanding the system to other cards that the bus loading be kept below minimums.

The Connector Signal Definition table indicates which lines are buffered.

With all address and data lines on the bus being driven by MOS chips, and also being listened to by MOS chips, two aspects of bus-loading must be considered when extending the bus.

The first aspect is current-drive capability. The 8085 address and data lines can sink up to 2ma of current and still maintain an output-low voltage less than 0.45V. Connecting other MOS chips to the address and data lines add an insignificant current load (less than 10uA) to the bus, hence they need not be considered when checking bus current loading.

On the other hand, connecting TTL to the bus (e.g. for I/O address decoding) adds a significant current load to the bus. Since all TTL on the SCCS-85 is LS, the current load per TTL input is a max. of about .36 ma for a low input. Hence, an unbuffered MOS address or data line can support at most five LS TTL loads.

The other aspect of bus loading which must be considered is capacitance-loading. The timing specs for the 8085 chip are given assuming a 150pF load on the signal outputs. For loads between 150pF and 300pF timing specs are to be derated by +0.30ns per pF. In other words, if we load the bus to 300pF we must derate the timing specs by 45ns. In high bus-load systems, the advantages of running the CPU at 2MHz rather than 3MHz is apparent.

The Intel specs for the MOS peripheral chips (e.g. 8251, 8253, etc.) quote that an input to the chip has a max. capacitance of 10pF, while a bi-directional data pin has a max. capacitance of 20pF. LS TTL inputs also have an input capacitance of about 5 pF, but since more than five LS TTL chips will overload the current drive of the bus, the total capacitance of the TTL loads on the bus can be overlooked in most cases.

Based on the above and armed with the data sheets for the chips on your particular SCCS-85 you can add up the bus loading for the various signals and establish how much more any bus extension dare load the bus. To give some idea of what you might come up with, the following table shows the loading on the upper 8 address lines, the lower 8 address lines, and the data bus for a FULLY POPULATED SCCS-85:

A8-A15	A0-A7	D0-D7
65pF, 1 TTL	110pF, 1 TTL	134pF, 1 TTL

Above it can be seen that for the user with some options missing on his system, quite a bit of "headroom" exists for expanding the bus without overdoing the loading.

CONNECTORS

The SCCS-85 provides a convenient and flexible system for making connections between the board and the outside world. Connectors J1 through J6a are each a double-row of plated-through holes spaced on 0.100 inch centers in patterns of from 10 to 50 holes.

These patterns can support a wide variety of connector hardware which is not only versatile, but inexpensive as well. Some of the possibilities are outlined below.

No connections at all

The connector system used has advantages even where no connections at all are made to a connector. The connector pattern is very inexpensive to produce on a PC board, and requires no special tooling, gold-plating, and the like. You don't have to pay extra for something you may not even use!

Just one or two connections

If only one or two connections need to be made, a piece of wire can simply be soldered directly to the proper hole in the pattern without damaging its future use with a regular connector.

With ribbon cables

When a ribbon cable is to be connected to a pattern, it is recommended that a "double-row jumper header" made by AP Products be installed in the connector. These are double rows of pins bound together in the proper spacing by a plastic header. The header is simply inserted into the pattern and soldered on the bottom. Then a ribbon cable with a female connector on the end may be simply plugged onto the header.

These double-row headers come in two varieties; straight and right-angle. The straight variety will serve best on the I/O connectors J2 through J6a where a ribbon cable extending to a single destination will be plugged.

The right-angle type are very useful for "daisy-chaining" the bus connector J1 to several boards. For example, to make an economical "mother-board" for extending the bus to three other cards, 50-pin right-angle headers would be installed in connector J1 of all cards. Then, four 50-pin female ribbon cable connectors would be installed equidistantly along a length of 50-conductor ribbon cable. Finally, each of the four boards would be plugged onto this cable.

Plugging boards directly into other boards

AP Products also makes double-row FEMALE headers which will solder directly into the connector patterns on the board. This provides two ways in which boards may be plugged directly into other boards.

For example if a right-angle header is installed in connector J1 and a female header installed in another PC board, then the SCCS-85 can be plugged perpendicularly into the "mother-board".

Or, if it is known that the SCCS-85 will extended to only one other board then one may solder a female header in the connector on the TOP side of the peripheral board, and a male header installed on the BOTTOM side of the SCCS-85. Then the two boards may be easily plugged together in a "sandwitch" configuration. This technique will be recommended for the up-coming CRT interface card, when using the SCCS-85 as a dedicated smart terminal.

CONNECTOR J2, J3, and J4

According to the silk-screen legend on the board, J4 is the combination of connectors J2 and J3.

J3 is mainly associated with inputs and outputs for the Timer Group, while J2 provides access to the PIO group. If the signals from J2 and those from J3 are destined to different places, then a 26-pin header should be installed in J2 and a 10-pin header installed in J3. Then, separate ribbon cables may be plugged into these connectors.

If, on the other hand, both the timer and PIO signals will be cabled to the same destination, then a single 40-pin header may be installed in J4. A single 40-conductor ribbon cable will then be sufficient for the interconnection.

The Connector Pin Assignment table in chapter 5 gives the signal assignments for connectors J2 through J4.

SERIAL CONNECTOR J6b

One last connector deserves mentioning, that being the RS-232 connector J6b. Provisions have been made for installing an easy-to-get DB-25S "D" connector directly on the SCCS-85 board. Once installed, this connector is already wired in the RS-232 standard configuration.

On the PC board note that connector J6b appears to be a set of edge-finger contacts. In fact, the positioning of these fingers on the top and bottom of the board exactly coincide with that of the pins on a DB-25S connector. Further, the two rows of pins on the connector are separated by 1/16th inch, just the thickness of the SCCS-85 board. This means that the connector may be slipped onto the edge of the board with the row of 13 pins on the bottom side and the row of 12 pins on the top side. Sliding the connector along the edge of the board, eventually the pins will line-up with the edge fingers. Once positioned, simply solder each of the pins on the connector to the finger directly beneath it. Not only is the connector now correctly wired, but rigidly mounted as well!

ADDITIONAL I/O SELECT LINES

If the user is adding additional peripheral chips on another board he may find it convenient to use one or more of the unused outputs of the ON-BOARD I/O ADDRESS DECODER to perform the chip-select function. (Refer to dwng. no. 4 of the schematics.)

These pads are located between U19 and U22 on the PC board labelled 50h, 60h, and 40h.

These pads each go low anytime the lower 8 bits of the address bus equal the indicated value through the next 15 higher addresses. For example, the pad labelled 50h will go low anytime the address bus contains XX50h through XX5Fh where XX indicates that the upper 8 bits are arbitrary. (This makes no difference during I/O cycles since the 8085 places the I/O address on BOTH the upper AND lower 8 bits of the address bus.)

It is important to note that the peripheral using these I/O address select signals must qualify them with either the $\overline{IOR}$ or $\overline{IOW}$ control signal. This is because these outputs will go low when MEMORY locations with addresses in the proper range are being accessed. It is a convention that Intel peripheral chips qualify their chip-select inputs with the $\overline{IOR}$ and $\overline{IOW}$ signals.

4. SOFTWARE ENGINEERING

This chapter contains hardware-related software information; such things as the I/O addresses of the various peripheral chips, interrupt vector addresses, programming the timer chip to provide standard baud rates for the 8251 USART, etc.

POWER-UP INFORMATION

When power is first applied to the 8085 the reset circuitry will reset the 8085 automatically, after which the 8085 will immediately begin executing at loc. 0000 in memory. Therefore, there must be some program there to execute. That is why at least one EPROM must be installed in U6. At loc. 0000 in your software should be the initialization routine needed for your particular hardware. For example, if the timer chip is used to provide the baudrate clock for the USART, then the timer chip must be initialized to operate timer 1 in the correct mode and divide by the proper number for the desired baud rate. Then the USART must be initialized for the desired mode of operation.

If interrupts are being used, the 8085 interrupt mask must be initialized to enable the desired interrupts.

The stack pointer must be set to point to the top of your available RAM. Etc., etc.

Note also that this same reset sequence takes place any time the manual reset pushbutton SW1 is closed.

RAM

As mentioned under hardware engineering, RAM starts at 1000H and goes up to 13FF for 1K bytes, 17FF for 2K bytes, 1BFF for 3K bytes, and 1FFF for 4K bytes.

PERIPHERAL CHIP I/O ADDRESSES

There are four peripherals on a fully-populated SCCS-85. These are the 8253 timer, 8255 PIO, 8251 USART, and 8257 DMA. The following table gives the I/O addresses for each of the registers within the chip:

DEVICE	ADDRESS	READ OPRATION	WRITE OPERATION
8251 USART	00h	Rec. Data Reg.	Trans. Data Reg.
	01h	Status Reg.	Control Req.
8255 PIO	10h	Read Port A	Write Port A
	11h	Read Port B	Write Port B
	12h	Read Port C	Write Port C
	13h	ILLEGAL	Write Control Reg.

8253 TIMER	20h	Rd. Counter 0	Wrt. Counter 0
	21h	Rd. Counter 1	Wrt. Counter 1
	22h	Rd. Counter 2	Wrt. Counter 2
	23h	ILLEGAL	Write Mode reg.
8257 DMA	30h	Read/write chan. 0 DMA address	
	31h	Read/write chan. 0 Terminal Count reg.	
	32h	Read/write chan. 1 DMA address	
	33h	Read/write chan. 1 Terminal Count reg.	
	34h	Read/write chan. 2 DMA address	
	35h	Read/write chan. 2 Terminal Count reg.	
	36h	Read/write chan. 3 DMA address	
	37h	Read/write chan. 3 Terminal Count reg.	
	38h	Read Status Reg.	Write Mode Reg.

I/O Addresses 80h and higher are available for user definition

INITIALIZATION OF TIMER AND USART FOR SERIAL I/O

If you are using the 8253 timer and 8251 USART in the configuration the board is manufactured in, then the following 8085 assembly code may be used to initialize both for serial I/O at the buad rate of your choice:

```
;*************************************************************
; code to initialize 8253 timer and 8251 USART for serial I/O on
; SCCS-85 board.

; first initialize timer chip to generate 16X baudrate for USART
        mvi     a,76h   ;program timer 1 for mode 3, expect 2 bytes
        out     23h
        mvi     a,lobaud ;send lower byte of baudrate divisor to timer
        out     21h
        mvi     a,hibaud ;send upper byte of baudrate divisor
        out     21h

;next initialize USART
        mvi     a,82h   ;force usart to expect command word
        out     01h
        mvi     a,40h   ;now make usart expect mode word
        out     01h
        mvi     a,4eh   ;mode byte - baud clock is 16X
        out     01h
        mvi     a,27h   ;command byte
        out     01h

; initialization complete!
;*************************************************************
```

In the code above there are two bytes, lobaud and hibaud, which the user must determine to select the desired baudrate. The table below gives the proper divisor value for each of the standard baudrates, and for two different CPU clock crystals. To use this divisor value, convert it into hexadecimal. Then the upper two hex digits are "hibaud" and the lower two digits are "lobaud".

Baud Rate	Divisor, with 3.58MHz crystal, (1.79MHz CPU clk)	Divisor, with 4.00MHz crystal (2MHz CPU clk)
38,400	3	not possible
19,200	6	not possible
9600	12	13
4800	23	26
2400	47	52
1200	93	104
600	186	208
300	373	417
150	746	833
110	1017	1136
75	1491	1667

INTERRUPTS

As configured the SCCS-85 makes use of the RST7.5 and RST6.5 interrupt inputs on the 8085.

When a RST7.5 interrupt occurs, the equivalent of a CALL instruction to loc. 003Ch is executed. At this point the user should store a JMP instruction to the service routine for that particular interrupt. Don't forget to preserve the contents of the registers and re-enable interrupts before returning to the interrupted program.

Similarly, when a RST6.5 interrupt occurs, the equivalent of a CALL to loc. 0034h is executed.

5. Hardware Reference

CONNECTOR PIN ASSIGNMENTS

J1 - Expansion bus

1	GND	2	GND
3	A1	4	A0
5	A3	6	A2
7	A5	8	A4
9	A7	10	A6
11	A9	12	A8
13	A15	14	A14
15	A13	16	A12
17	A11	18	A10
19	MEMSEL	20	(not used)
21	+12V	22	-12V
23	$\overline{IOR}$	24	$\overline{MEMR}$
25	$\overline{MEMW}$	26	$\overline{IOW}$
27	D6	28	D7
29	D4	30	D5
31	D2	32	D3
33	D0	34	D1
35	$\overline{INTA}$	36	AEN
37	S1	38	S0
39	INTR	40	RST5.5
41	IO/$\overline{M}$	42	TRAP
43	(user defined)	44	ALE
45	READY	46	CLK
47	RESET OUT	48	$\overline{RESET\ IN}$
49	+5V	50	+5V

J4 - PIO/TIMER

1	PA4	2	PA3
3	PA5	4	PA2
5	PA6	6	PA1
7	PA7	8	PA0
9	+5V	10	GND
11	PC6	12	PC7
13	PC4	14	PC5
15	PC1	16	PC0
17	PC3	18	PC2
19	FB7	20	PB0
21	PB6	22	PB1
23	PB5	24	PB2
25	PB4	26	PB3
27	(not used)	28	(not used)
29	(not used)	30	(not used)
31	user defined	32	OUT2
33	RST7.5	34	$\overline{CLK}$
35	user defined	36	user defined
37	SID	38	SOD
39	user defined	40	user defined

J2 - PIO

1	PA4	2	PA3
3	PA5	4	PA2
5	PA6	6	PA1
7	PA7	8	PA0
9	+5V	10	GND
11	PC6	12	PC7
13	PC4	14	PC5
15	PC1	16	PC0
17	PC3	18	PC2
19	PB7	20	PB0
21	PB6	22	PB1
23	PB5	24	PB2
25	PB4	26	PB3

J3 - TIMER

1	user defined	2	OUT2
3	RST7.5	4	$\overline{CLK}$
5	user defined	6	user defined
7	SID	8	SOD
9	user defined	10	user defined

J5 - DMA

1	DRQ1	2	DRQ0
3	DRQ3	4	DRQ2
5	$\overline{DACK0}$	6	$\overline{DACK1}$
7	$\overline{DACK2}$	8	$\overline{DACK3}$
9	GND	10	TC

J6b - RS-232 Delta connector

1 GND
2 $\overline{\text{Transmit data}}$
3 $\overline{\text{Receive data}}$
4 Request to send (output)
5 Clear to Send (input)
7 GND
8 Data Set Ready (input)
20 Data Terminal Ready (output)

J6a - RS-232 double-row header

1 GND
3 $\overline{\text{Transmit data}}$
5 $\overline{\text{Receive data}}$
7 Request to send
9 Clear to send
13 GND
15 Data Set Ready
14 Data Terminal Ready

NOTE: Connectors J2 and J3 are positioned such that together they may be considered a single 40-pin connector J4, or used individually.

BUS SIGNAL DEFINITIONS

GND — Logic ground for SCCS-85

A0 - A15 — (Output) Address lines 0 through 15. These are positive true signals. Lower eight bits are valid from the falling edge of ALE to end of machine cycle, and are buffered EXCEPT DURING A DMA CYCLE. (See data sheets on 8257 DMA controller. Upper eight bits are also valid from the falling edge of ALE to end of cycle but are not buffered.

D0 - D7 — Bi-directional positive-true data bus. During write cycles data on bus is valid during trailing edge of $\overline{\text{MEMW}}$ or $\overline{\text{IOW}}$ pulse. During read cycles, data must be valid on trailing edge of $\overline{\text{MEMR}}$ or $\overline{\text{IOR}}$ pulse. The data bus is not buffered.

$\overline{\text{IOR}}$ — (Output) $\overline{\text{I/O READ}}$ control signal. Low-going pulse during which selected peripheral should enable its tri-state bus drivers and place data on bus. Data must be valid on the trailing (rising) edge of pulse. Buffered by LS TTL gate.

$\overline{\text{IOW}}$ — (Output) $\overline{\text{I/O WRITE}}$ control signal. Low-going pulse used by peripherals to strobe data on bus into peripheral register. Latching should occur on trailing (rising) edge of pulse. Buffered by LS TTL gate.

$\overline{\text{MEMR}}$ — (Output) $\overline{\text{MEMORY READ}}$ control signal. Low-going pulse during which selected memory device should place its data on the bus. Data must be valid on the trailing (rising) esge of pulse. Buffered by LS TTL gate.

$\overline{\text{MEMW}}$ — (Output) $\overline{\text{MEMORY WRITE}}$ control signal. Low-going pulse used by memory devices to enable the writing of the data currently on the bus into the selected memory location. Buffered by LS TTL gate.

AEN — (Output) ADDRESS ENABLE signal used only during DMA cycles. Otherwise remains low. Positive-true signal which indicates that the address currently on the address bus is that provided by the DMA controller during a DMA transfer cycle. Unbuffered, has same timing and specs as pin by same name on 8257 DMA controller.

S0, S1 — (Output) Machine-cycle status bits output by 8085. Same timing specs and definitions as pins by same name on 8085. Not buffered.

IO/$\overline{M}$ — (Output) Indicates if the current READ or WRITE is to memory or I/O. Same timing and specs as pin by same name on 8085. Not buffered.

ALE — (Output) Positive going pulse used to latch the lower eight address bits and the status bits S0 and S1. Latches should be a level-triggered type. Not buffered.

CLK — (Output) Square wave of half the frequency of the crystal used to clock the 8085. Same timing as the pin by same name on 8085. Not buffered.

READY — (Input) This input is used by a slow peripheral or memory to insert wait states into a machine cycle. Has pullup resistor, so may be left unconnected, or several devices may drive the input through open-collector gates.

$\overline{\text{RESET IN}}$ — (Input) When pulled low the program counter is reset to 0 and the INTE flip-flop and HLDA flip-flop are reset. May be momentarily grounded with pushbutton switch to effect a manual reset.

RESET OUT — (Output) Positive-true signal that indicates the CPU is being reset. May be used as a system reset. Not buffered.

INTR — (Input) Positive-true input which initiates an interrupt to the 8085. Same definition and restrictions as pin by same name on 8085.

$\overline{\text{INTA}}$ — (Output) Is used instead of (and has the same timing as) the $\overline{\text{MEMR}}$ during the next instruction cycle after the INTR has been accepted. Has same timing and specs as pin by same name on 8085. Not buffered.

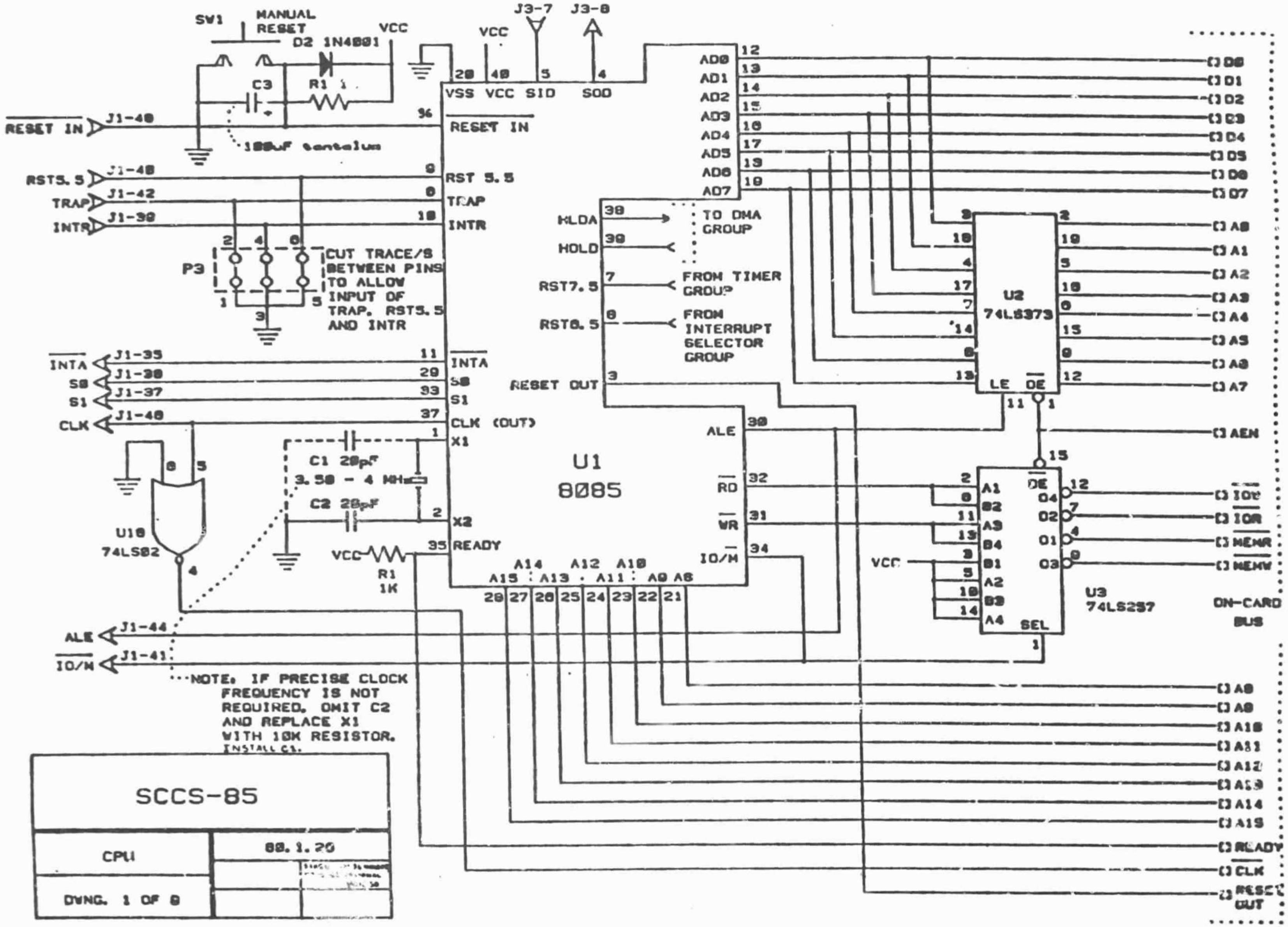
TRAP — (Input) Input which causes a non-maskable interrupt. Control transfers to location 0024h in memory. Same timing and specs as pin by same name on 8085.

RST5.5 — (Input) Has same input timing as INTR but causes a RESTART to automatically be inserted. Control is transferred to loc. 002Ch. Maskable.

MEMSEL — (Output) Positive-true signal which indicates that the address currently on the bus is within the first 8K bytes of memory space. Is useful in systems with memory expanded onto additional cards to determine if the memory selected is on the SCCS-85 card. Buffered by LS TTL gate.

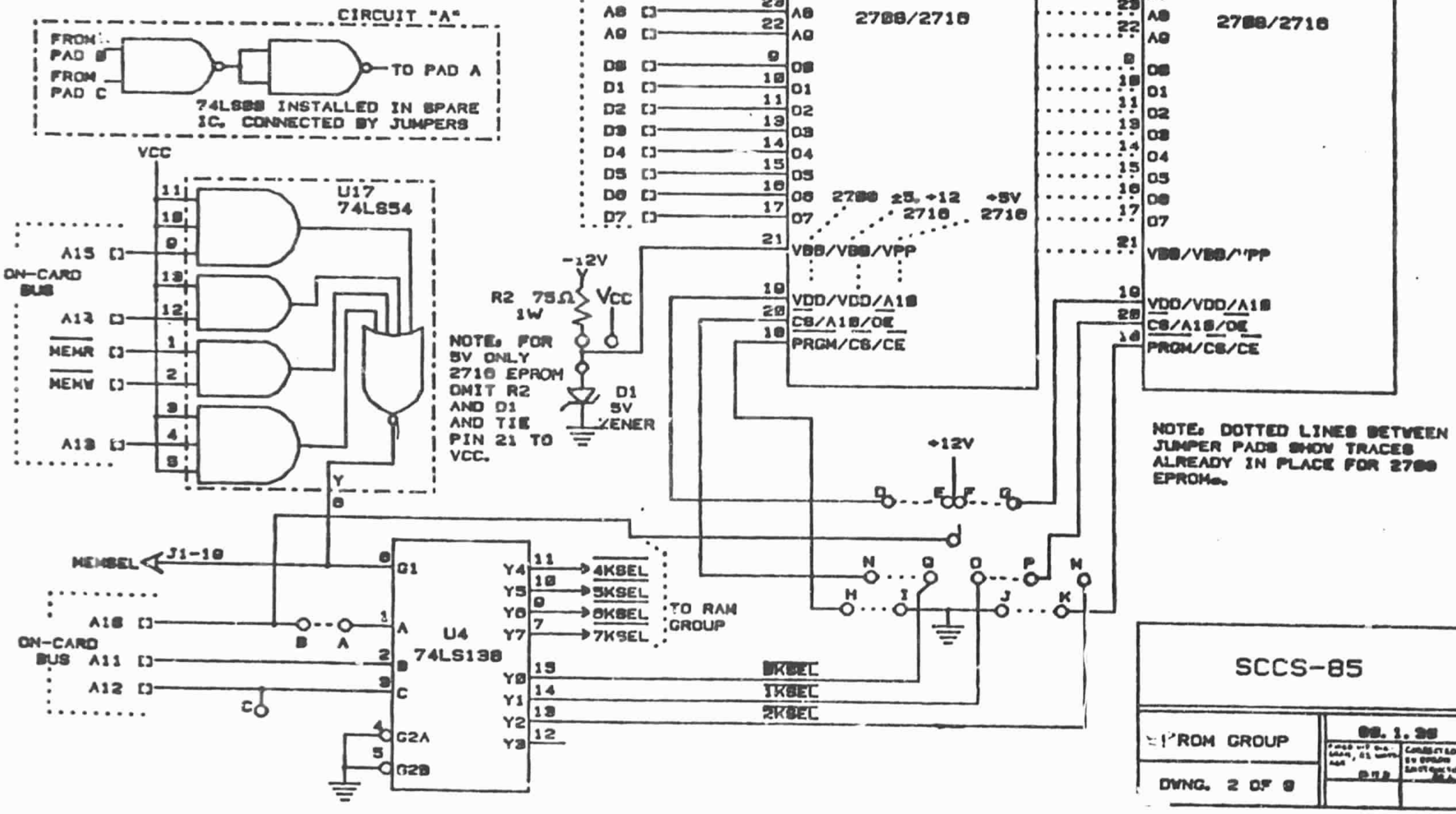
+12V, -12V — Power supply inputs to SCCS-85 board. Requirements are regulation to plus or minus 10% with currents up to 150 ma. If the SCCS-85 uses 5V-only EPROMS and the RS-232 driver chip is not used, these supplies may be omitted.

+5V Logic supply to all chips. Must be regulated to plus or minus 5% and capable of 1.5 A for a fully-populated SCCS-85.



SCCS-85	
CPU	80.1.26
DWNG. 1 OF 8	

	CUT TRACES:	JUMPER PADS
2708	NONE	NONE
+5, -5, +12V 2716	N-Q, O-P, H-I, J-K, A-B	N-L-P, H-G, K-M INSTALL CIRCUIT "A"
+5V ONLY 2716	D-E, F-G, O-P, A-B	D-L-G, P-M INSTALL CIRCUIT "A"

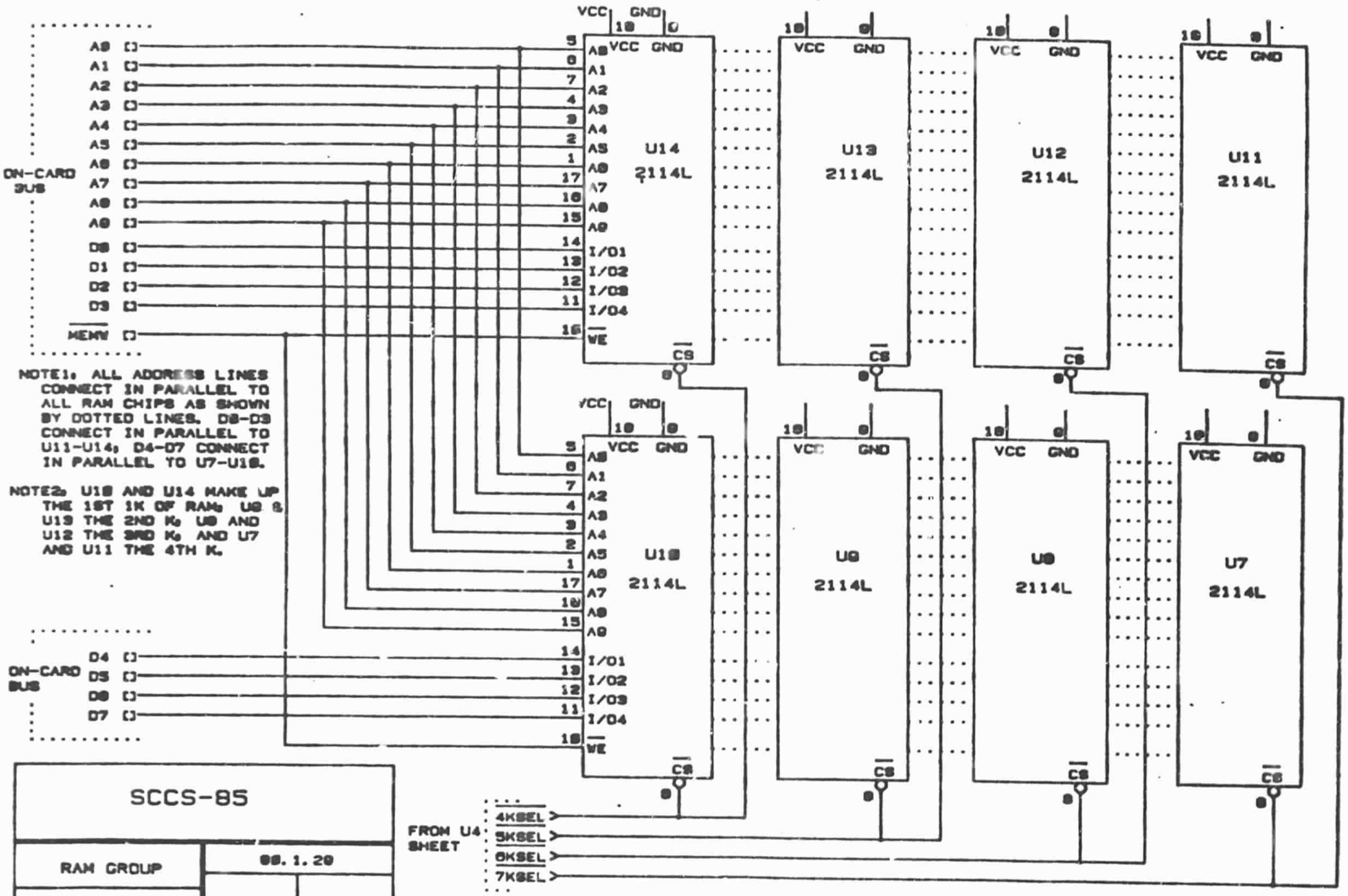


NOTE: DOTTED LINES BETWEEN JUMPER PADS SHOW TRACES ALREADY IN PLACE FOR 2708 EPROMs.

SCCS-85

EPROM GROUP

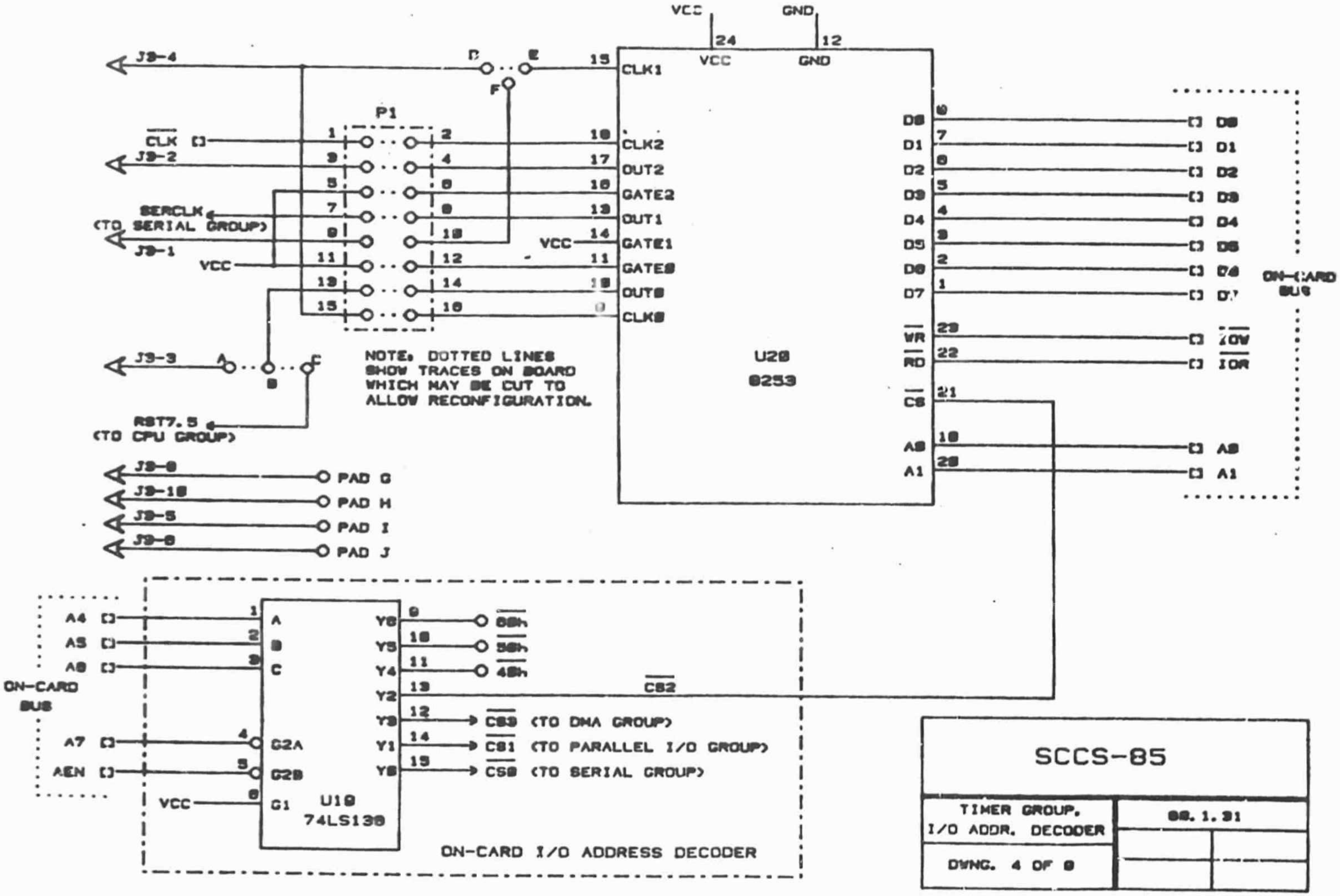
DWNG. 2 OF 8

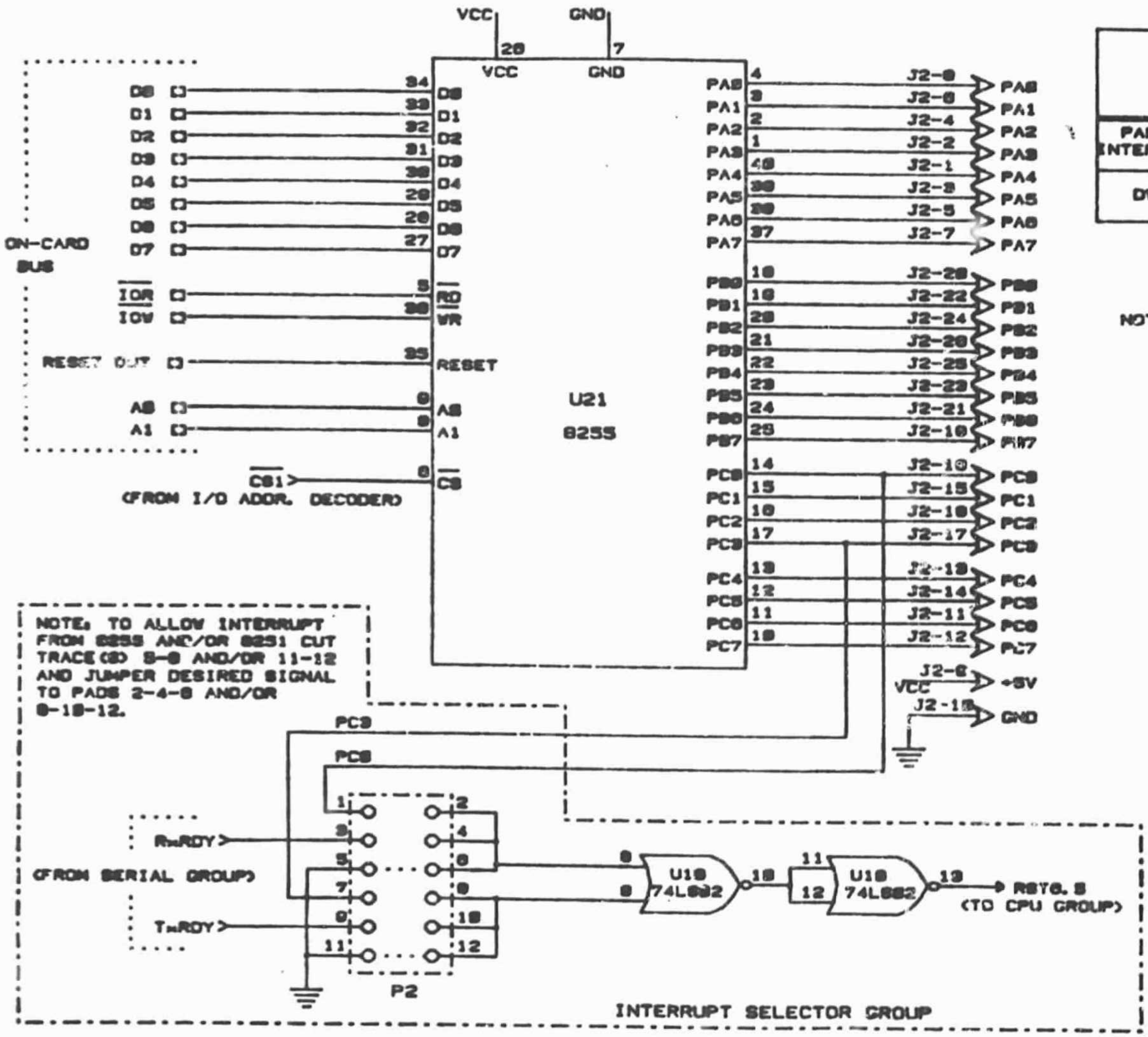


NOTE1. ALL ADDRESS LINES CONNECT IN PARALLEL TO ALL RAM CHIPS AS SHOWN BY DOTTED LINES. D0-D3 CONNECT IN PARALLEL TO U11-U14, D4-D7 CONNECT IN PARALLEL TO U7-U10.

NOTE2. U10 AND U14 MAKE UP THE 1ST 1K OF RAM, U9 & U13 THE 2ND K, U8 AND U12 THE 3RD K, AND U7 AND U11 THE 4TH K.

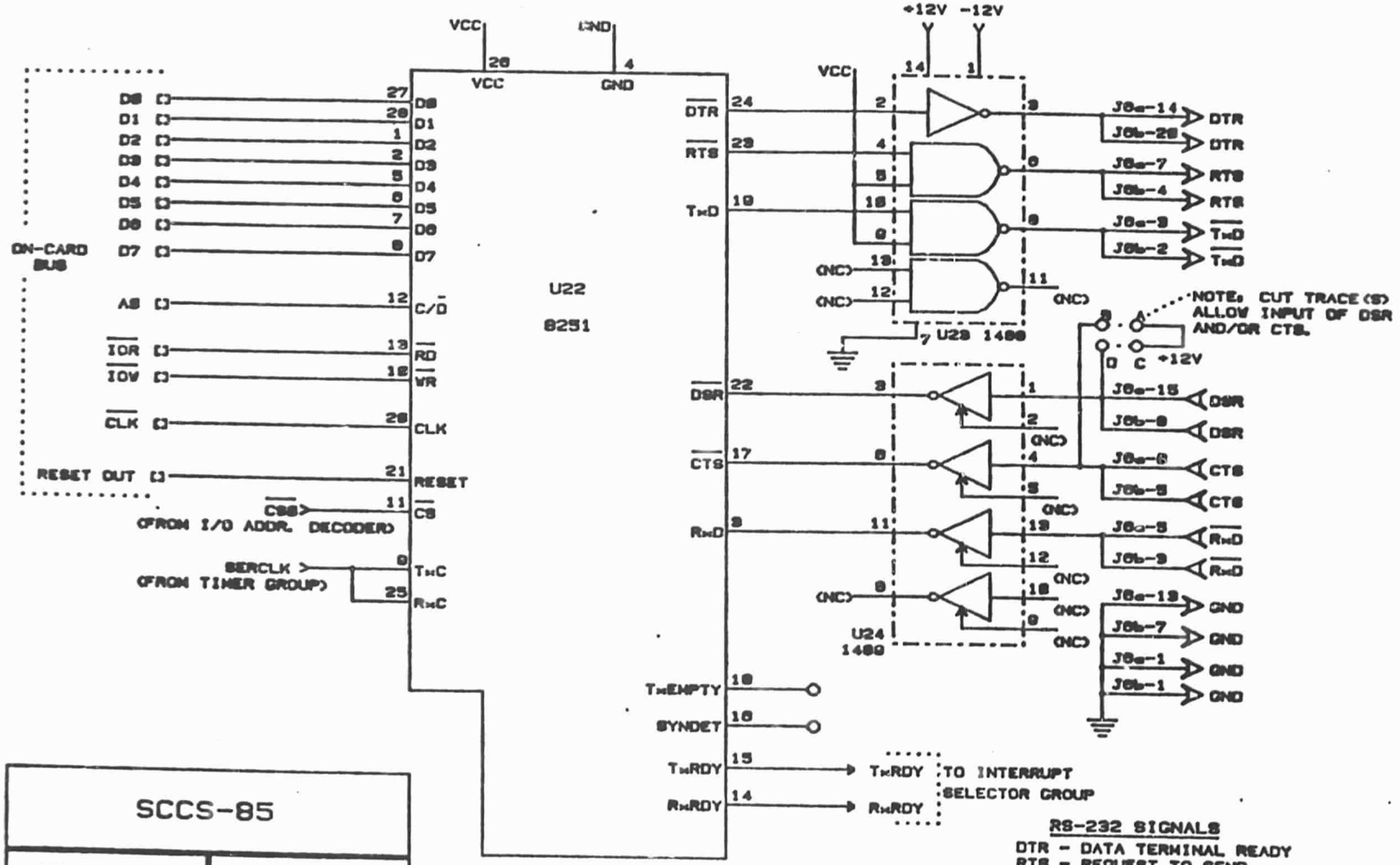
SCCS-85		
RAM GROUP	88. 1. 20	
DWNG. 3 OF 8		


SCCS-85
TIMER GROUP,
I/O ADDR. DECODER
88. 1. 31
DWNG. 4 OF 8
U20
8253
VCC
GND
CLK1
CLK2
OUT2
GATE2
OUT1
GATE1
GATE0
OUT0
CLK0
D0 D1 D2 D3 D4 D5 D6 D7
WR
RD
CS
A0
A1
P1
J3-4
CLK
J3-2
SERCLK
(TO SERIAL GROUP)
J3-1
J3-3
RST7.5
(TO CPU GROUP)
NOTE: DOTTED LINES SHOW TRACES ON BOARD WHICH MAY BE CUT TO ALLOW RECONFIGURATION.
ON-CARD BUS
IOW
IOR
PAD G
PAD H
PAD I
PAD J
U19
74LS138
A4 A5 A6 A7 AEN
G2A
G2B
G1
60h
50h
40h
CS2
CS3 (TO DMA GROUP)
CS1 (TO PARALLEL I/O GROUP)
CS0 (TO SERIAL GROUP)
ON-CARD I/O ADDRESS DECODER



SCCS-85		
PARALLEL GROUP, INTERRUPT SELECTOR GROUP	[illegible]	
DWNG. 5 OF [illegible]		

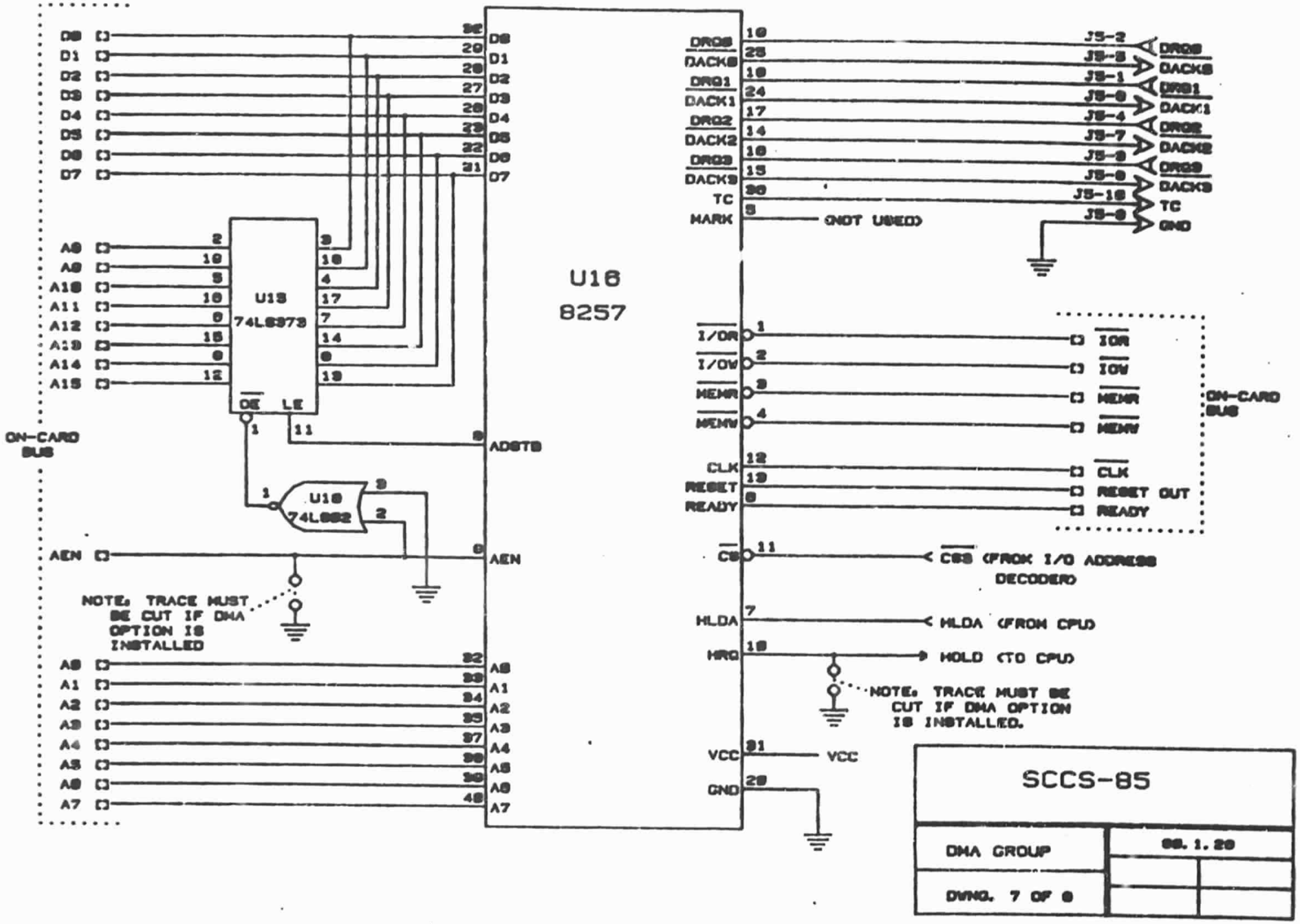
NOTE: PIN NUMBERS (SHOWN FOR J2) ALSO APPLY TO J4.



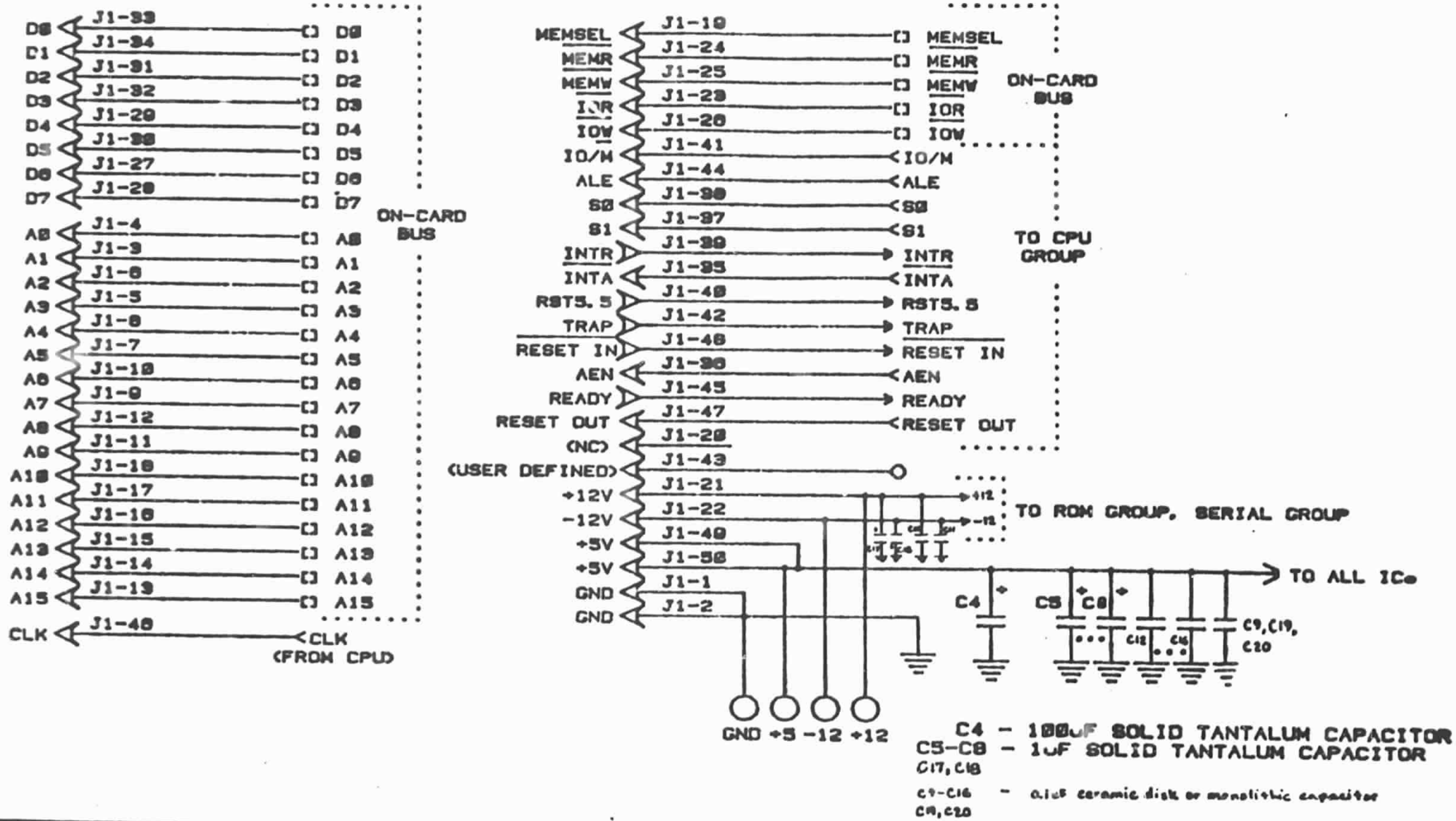
SCCS-85	
SERIAL GROUP	86. 1. 31
DWG. 8 OF 8	

RS-232 SIGNALS

DTR - DATA TERMINAL READY
RTS - REQUEST TO SEND
TxD - TRANSMIT DATA
DSR - DATA SET READY
CTS - CLEAR TO SEND
RxD - RECEIVED DATA



D0 D1 D2 D3 D4 D5 D6 D7
A8 A9 A10 A11 A12 A13 A14 A15
U13
OE LE
ON-CARD BUS
U18
AEN
NOTE: TRACE MUST BE CUT IF DMA OPTION IS INSTALLED
A0 A1 A2 A3 A4 A5 A6 A7
U16
8257
ADSTB
DRQ0 DACK0 DRQ1 DACK1 DRQ2 DACK2 DRQ3 DACK3 TC MARK
(NOT USED)
TC
GND
I/OR I/OW MEMR MEMW
IOR IOW MEMR MEMW
CLK RESET READY
CLK RESET OUT READY
ON-CARD BUS
CS
(FROM I/O ADDRESS DECODER)
HLDA
HLDA (FROM CPU)
HRQ
HOLD (TO CPU)
NOTE: TRACE MUST BE CUT IF DMA OPTION IS INSTALLED.
VCC
VCC
GND
SCCS-85
DMA GROUP
DWG. 7 OF 8



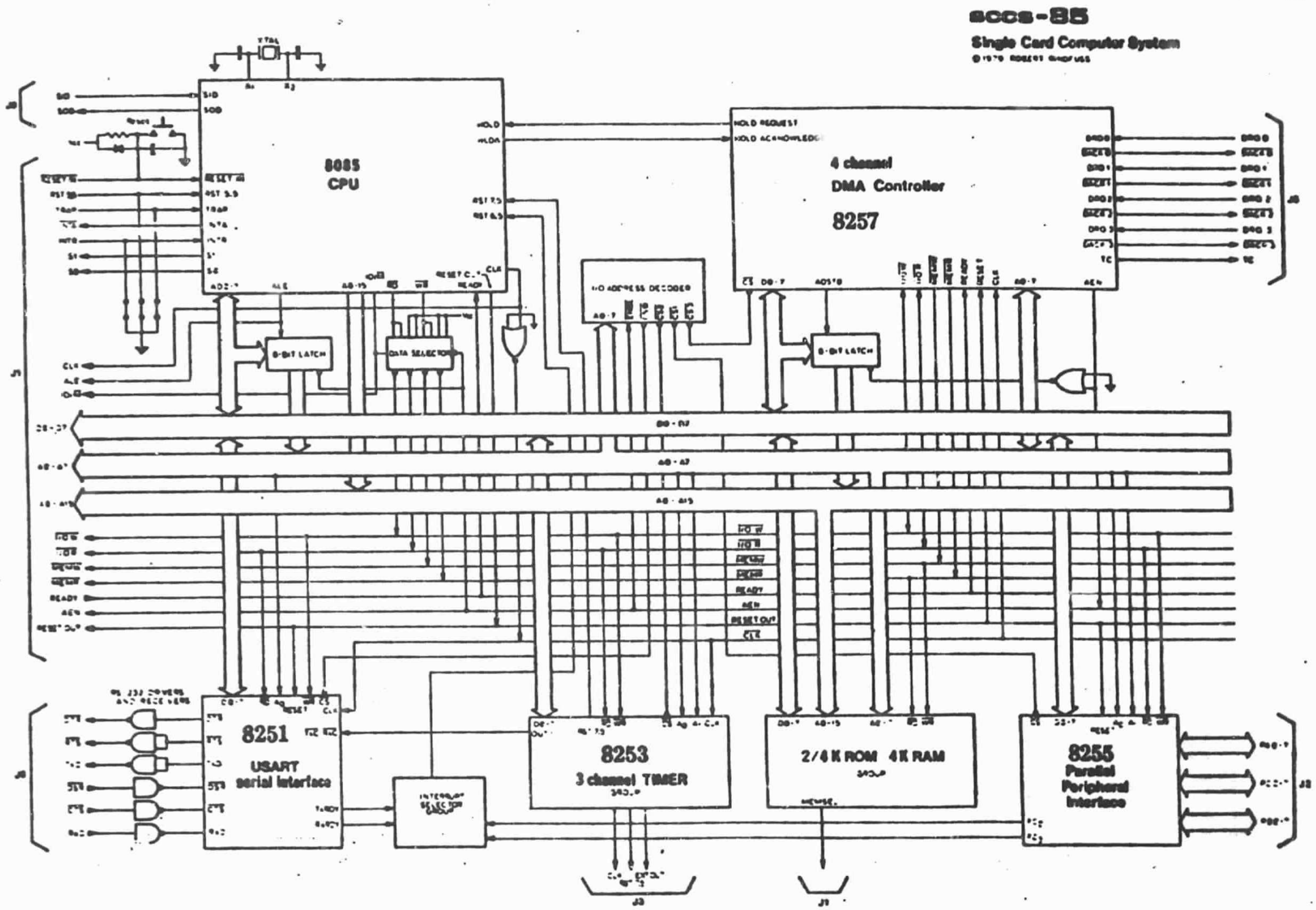
SCCS-85

BUS CONNECTOR, POWER SUPPLY

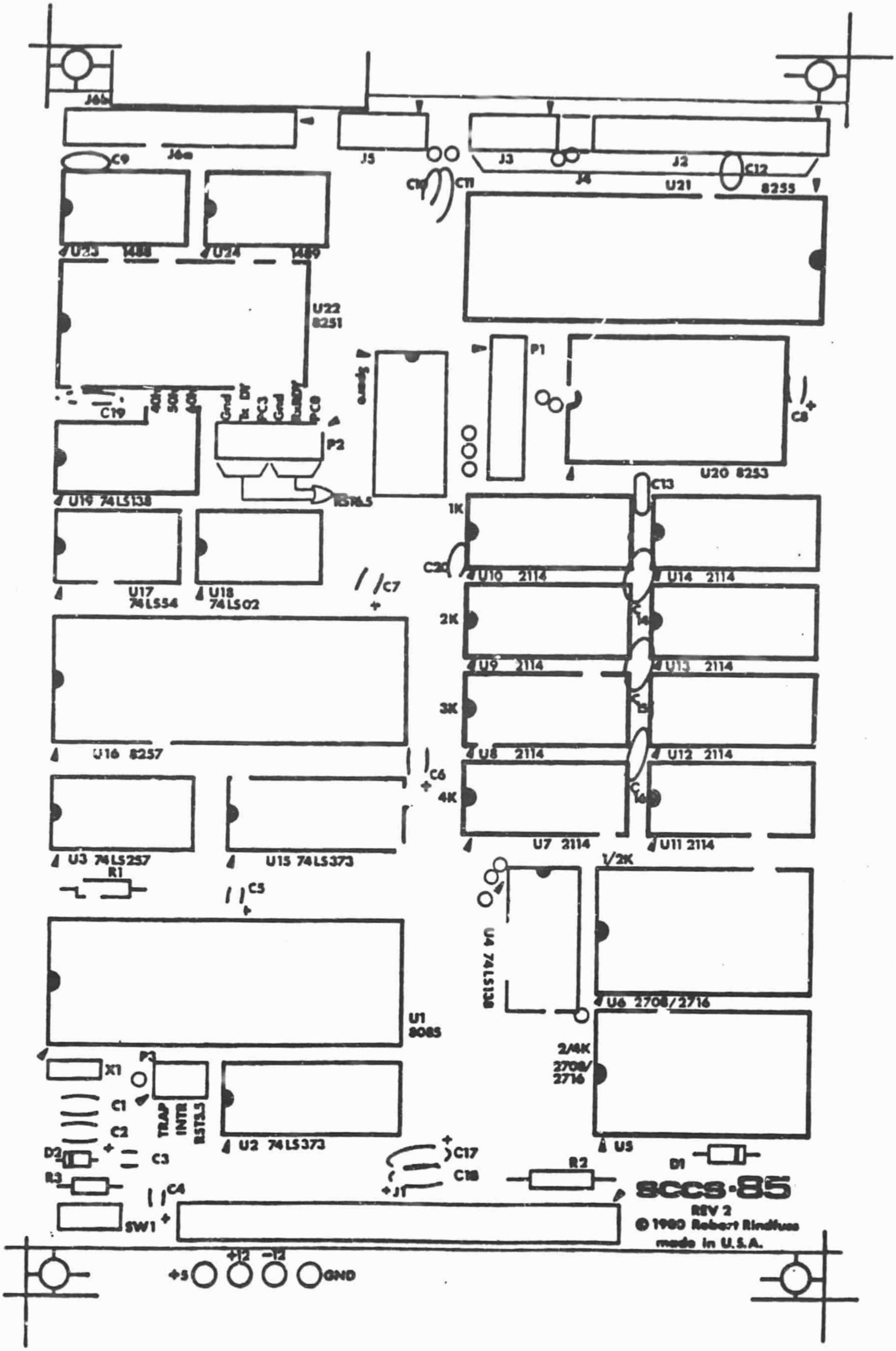
DWNG. 8 OF 8

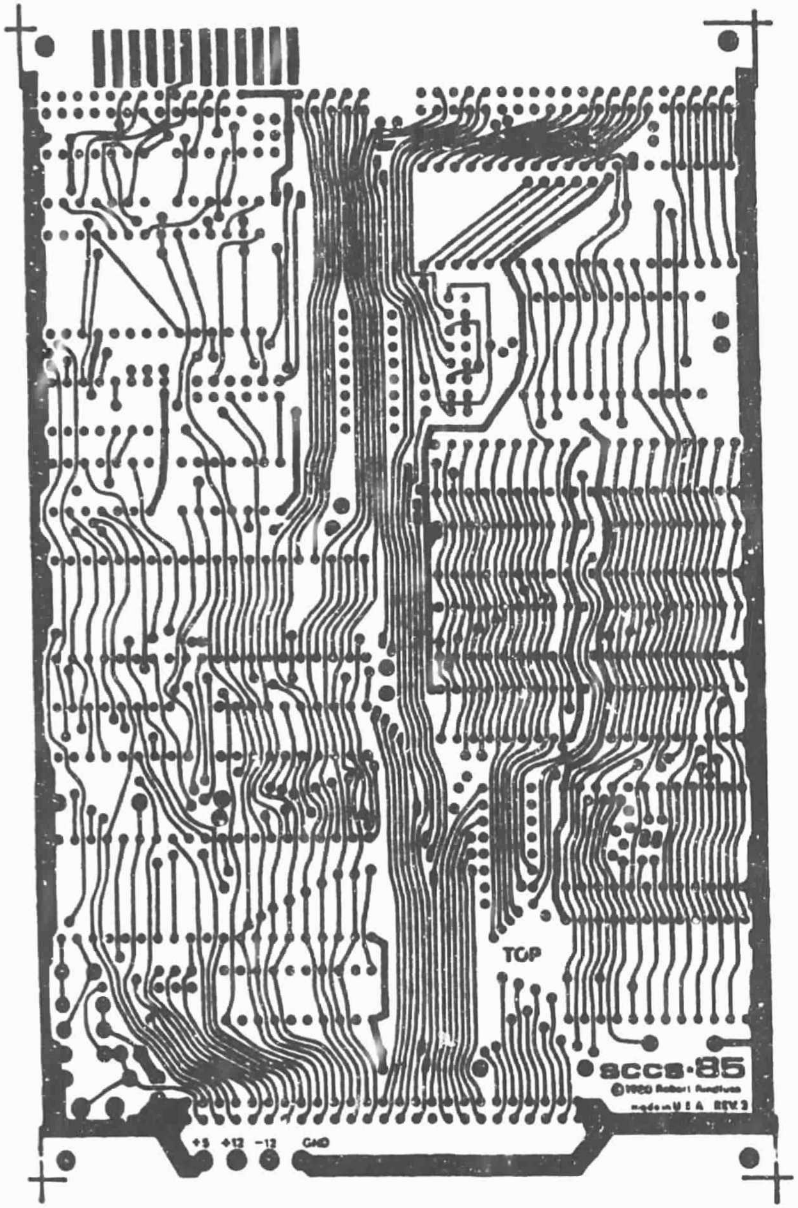
80. 2. 1

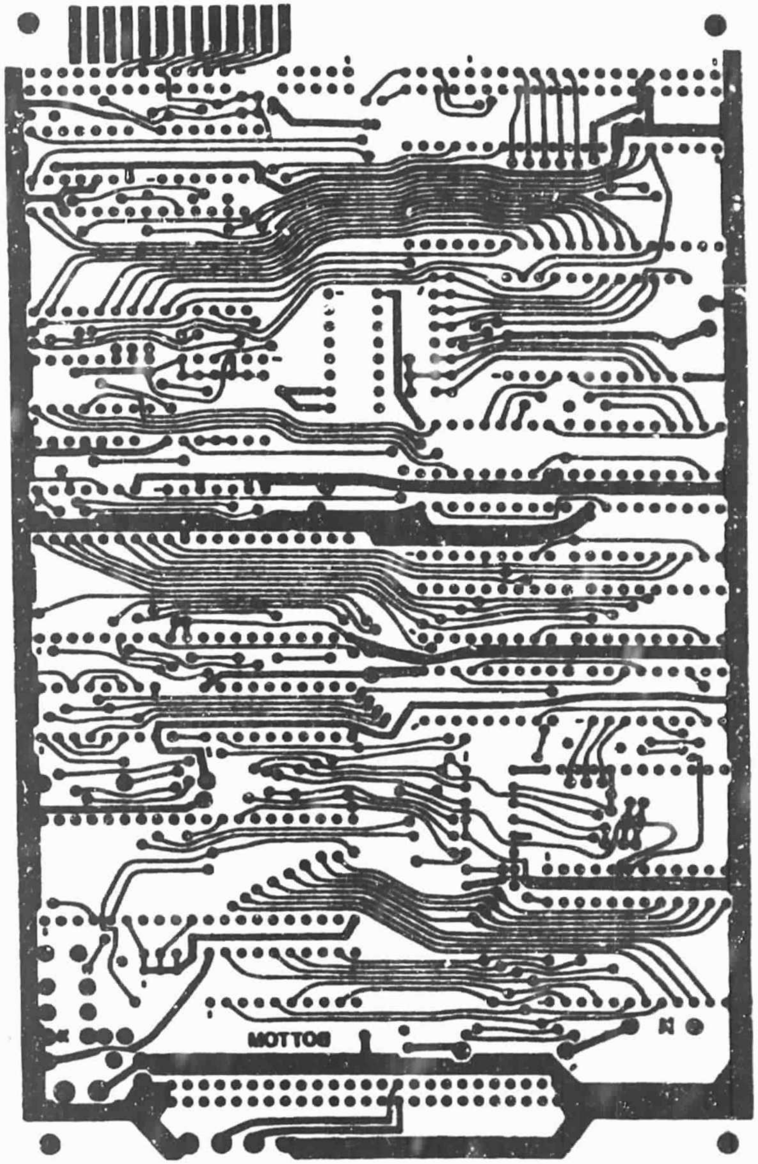
ADDED DECOUPLING CAPACITORS 80 12 ?


sccs-85
Single Card Computer System
8085
CPU
4 channel
DMA Controller
8257
8-BIT LATCH
DATA SELECTOR
I/O ADDRESS DECODER
D0 - D7
A0 - A7
A8 - A15
8251
USART
serial interface
INTERRUPT SELECTOR GROUP
8253
3 channel TIMER
2/4 K ROM 4 K RAM
8255
Parallel
Peripheral
Interface

DWNG. 9 OF 9


J6b
J6a
J5
J3
J4
J2
C9
C10
C11
C12
U21 8255
U23 1488
U24 1489
U22 8251
P1
U20 8253
C8
C19
P2
RST6.5
C13
U19 74LS138
1K
C20
U10 2114
U14 2114
U17 74LS54
U18 74LS02
C7
2K
U9 2114
U13 2114
3K
U8 2114
U12 2114
U16 8257
C6
4K
U7 2114
U11 2114
U3 74LS257
U15 74LS373
R1
C5
1/2K
U4 74LS138
U6 2708/2716
U1 8085
2/4K 2708/2716
U5
X1
P3
C1
C2
C3
D2
R3
C4
SW1
TRAP
INTR
RST5.5
U2 74LS373
C17
C18
J1
R2
D1
SCCS-85
REV 2
© 1980 Robert Rindfuss
made in U.S.A.
+5
+12
−12
GND


TOP
sccs-85
made in U S A REV 3
+5 +12 -12 GND


BOTTOM

```
;01.30.81 17:51         RJC: Renamed "form" "edm1" and "form2" "edm2".
;                       Moved "edm1" "edm2" "phi" and "prmpt" to
;                       spare bytes of RSTs.  Removed CICO and
;                       it's entry in the jump table.  Added
;                       RAM byte "echofl" and code to set it on
;                       pwrup, and at "loader:" and "done:".
;                       Changed CI to echo if "echofl"=0 only.
;                       Effect is to turn off echo on "l" comnd.
;                       WARNING! Jump table changed (sorry).
;12.03.80 00:00         RJC: CR delay: removed "CRDLY equ", added "sta
;                       dlyram" after "answer" init, changed all
;                       "mvi a,' '" to "xra a" to init answer,
;                       moved pwrup "answer" and "dlyram" init
;                       to before printing of startup message,
;                       changed "mvi a,CRDLY+1" to "lda dlyram"
;                       in CO, changed "jz" to "jm" in CO, added
;                       "dlyram db 1" to RAM constants.
;                 Changed "rz" to "rnz" in CO.
;                 Removed extraneous "cpi 'D'" at end of "d_p"
;11.21.80 17:13         RJC: Added "jmp comnd" to jump table.
;11.10.80 20:01         RJC: Changed restart vectors to vector ALL
;                       interrupts except RST0 to RAM.
;                 Rewrote comments in beginning and "loader"
;10.14.80 22:58         RJC: Added initialization of "answer" on power up
;                 Added "ath" entry point in "ghd" and added
;                       to jump table.
;05.10.80 12:17         RJC: Removed extraneous "jmp error" in goto cmd.
;                 Changed "rst 0" to "ret" in edit comments.
;                 Changed "jmp msg" to "call msg" "ret"; prbad
;05.01.80 16:12         RJC: Modified comment about tying RxD to DSR/ to
;                       recommend doing it at TTL signal levels.
;04.29.80 20.17         RJC: Removed "push/pop d" from msg and moved
;                       "inx h" to before "jz mdn" so that d&e
;                       point to byte after eol on exit.
;                 Added "EOL equ 0ffh".
;                 Changed WIDTH comments from 29 and 53 column
;                   to 28 and 52 columns.
;04.15.80 19:38         RJC: Changed CO to take character in a-reg.
;                 Changed repeat to take char in A-register
;                       and count in C-register.
;                 Expanded comments in beginning.
;                 Rearranged comments so that assembled i.lst
;                       would fit in 80 columns.
;04.11.80 18:16         RJC: Added comment-tie RS232 DSR input to RS232
;                       receive data input.
;04.05.80 00:01         RJC: Recalculate cycles not counted by 8253 for
;                       baud rate calculation.
;04.04.80 17:25         RJC: All functions tested.  Monitor "done".
;**************************************************************
;*                                                            *
;*            Copyright 1980 Raymond J. Clark                 *
;*                                                            *
;*    Permission is granted to copy  this  material  in       *
;*    whole  or in part by any means and for any purpose      *
;*    other than sale, barter, or profit  of  any  kind.      *
;*    In   return  it  is  requested  that  this  notice      *
;*    accompany the portion copied in recognition of the      *
;*    time and effort spent developing it.                    *
;*                                                            *
;**************************************************************
;
;     ****          sccs-85  1024 byte monitor           ****
;
;     **** Start up message:                             ****
;     ****                                               ****
;     ****          Mxxx.v                               ****
;     ****          xxx     = last used address          ****
;     ****          v       = version number which       ****
;     ****                    is incremented every       ****
;     ****                    time code is changed.      ****
;
;   Hardware configuration assumed:
;
;     8253:   Timer 1 clock input driven by high frequency
```

```
                ;               greater than 20*baud-rate.  Lower frequencies
                ;               may work if freq/(16*n) is within 5% of baud
                ;               rate where n is some integer.  Monitor finds n.
                ;       8251:   RxC and TxC driven by 8253 timer . output.
                ;               DSR/ input same as RxD input.  See "baud1"
                ;       RAM:    From RAM to RAM+03fH
                ;               From MEMTOP down for internal storage and stack
                ;
                ;    On power up        type "d".. 8253 divisor calculated automatically.
                ;
                ;       With 4 MHz crystal:
                ;               Works from 150 to 3600 baud.
                ;
                ;       With 3.579 MHz Color Burst crystal:
                ;               Works from 110 to 9600 baud.  Communicates at
                ;               19200 baud but tape load function will not work.
                ;
                ;
                ;     ***       Commands:
                ;
                ;       L       load intel hex tape. Bias not implemented.
                ;               "answer" = G on exit if no errors detected.
                ;               If errors were found, answer = B
                ;       E       edit memory in hex.  See comments in code.
                ;       G       goto address.  Stack pointer reset.
                ;       D       dump memory in hex.
                ;       P       punch intel hex tape.
                ;       (cr)    nop.  Does not change one byte "answer"
                ;       ?       print one byte message (answer) left by last
                ;               command.  This is cleared before the execution
                ;               of each command.
                ;
                ;       The "answer" is cleared before each command is run,
                ;       except ? and (cr) do not clear it.
                ;
                ;       The L, G, D, and P commands call an "ok" subroutine to
                ;       give you a last chance to abort the command.  A (cr) will
                ;       go ahead and execute the command, anything else aborts.
                ;
                ;       RAM address MEMTOP contains a number from 00h to 80h
                ;       which is the number of times d10ms is to be called after
                ;       outputing each carriage return (0dh).  On rst 0 it is
                ;       initialized to 00h.  It will have to be made non-zero
                ;       for some terminals, up to 40h for some.

       0        BASE   equ      00000h   ;base address of monitor
    1000        RAM    equ      1000h    ;address of first byte of RAM
    1fff        MEMTOP          equ      1fffh    ;address of last byte of RAM
      ff        EOL    equ      0ffh     ;end of string (line) character
       f        WIDTH equ       0fh      ;controls the width of "dump" "punch"
                ;                        ;commands:
                ;                        ;       0fh = 16 bytes, 52 columns
                ;                        ;       07h = 8 bytes, 28 columns

      21        timel equ       21h
      23        timctl          equ      23h

       1        sercon          equ      01h
       0        serdat          equ      00h

                ;==============================================================
                ;
                ; vectors for hardware interrupts
                ;
    1000        rst0   equ      RAM+     000h     ; not used - monitor reset
    1008        rst1   equ      RAM+     008h     ;
    1010        rst2   equ      RAM+     010h     ;
    1018        rst3   equ      RAM+     018h     ;
    1020        rst4   equ      RAM+     020h     ;
    1024        trap   equ      RAM+     024h     ;
    1028        rst5   equ      RAM+     028h     ;
    102c        rst55 equ       RAM+     02ch     ;
    1030        rst6   equ      RAM+     030h     ;
    1034        rst65 equ       RAM+     034h     ;
```

```
     1038          rst7  equ    RAM+    038h     ;
     103c          rst75 equ    RAM+    03ch     ;

                   ;--------------------------------------------------------------
                   ;
                   ; RST 0         entry point - power up reset        ;rst 0

     0                   org    BASE+0
     0 31 fd1f           lxi    sp,STACKINIT     ; initialize stackpointer
     3 c3 6d00           jmp    entry
     6  0                nop
     7  0                nop

                   ;--------------------------------------------------------------
                   ;
                   ; RST 1         entry point

     8                   org    BASE+08h                   ; rst 1
     8 c3  810           jmp    rst1
                   ;
                   phi:  db     ' TO '           ;these lines were moved from the
     b 20          54 4f 20
     f ff                db     EOL              ;ROM constant sect. to save space

                   ;--------------------------------------------------------------
                   ;
                   ; RST 2         entry point

     10                  org    BASE+10h                   ; rst 2
     10 c3 1010          jmp    rst2

                   edm1: db     ') = '           ;these lines were moved from the
     13 29         20 3d 20
     17 ff               db     EOL              ;ROM constant sect. to save space

                   ;--------------------------------------------------------------
                   ;
                   ; RST 3         entry point

     18                  org    BASE+18h                   ; rst 3
     18 c3 1810          jmp    rst3
                   edm2: db     0dh, 0ah         ;these lines were moved from the
     1b  d          a
     1d 28               db     '('              ;ROM constand sect. to save space
     1e ff               db     EOL              ;
     1f  0               nop

                   ;--------------------------------------------------------------
                   ;
                   ; RST 4         entry point

     20                  org    BASE+20h                   ; rst 4
     20 c3 2010          jmp    rst4
     23  0               nop

                   ;--------------------------------------------------------------
                   ;
                   ; TRAP entry point

     24                  org    BASE+24h                   ; trap
     24 c3 2410          jmp    trap
     27  0               nop

                   ;--------------------------------------------------------------
                   ;
                   ; RST 5         entry point

     28                  org    BASE+28h                   ; rst 5
     28 c3 2810          jmp    rst5
     2b  0               nop

                   ;--------------------------------------------------------------
                   ;
```

```
                        ; RST 5.5 entry         point

     2c                         org     BASE+2ch                ; rst 5.5
     2c c3 2c10                 jmp     rst55
     2f  0                      nop

                        ;=====================================================================
                        ;
                        ; RST 6         entry point

     30                         org     BASE+30h                ; rst 6
     30 c3 3010                 jmp     rst6
     33  0                      nop

                        ;=====================================================================
                        ;
                        ; RST 6.5 entry         point

     34                         org     BASE+34h                ; rst 6.5
     34 c3 3410                 jmp     rst65
     37  0                      nop

                        ;=====================================================================
                        ;
                        ; RST 7         entry point

     38                         org     BASE+38h                ; rst 7
     38 c3 3810                 jmp     rst7
     3b  0                      nop

                        ;=====================================================================
                        ;
                        ; RST 7.5 entry         point                   ;rst 5.5

     3c                         org     BASE+3ch
     3c c3 3c10                 jmp     rst75
     3f  0                      nop

                        ;=====================================================================
                        ;=====================================================================

     40                         org     BASE+40h

                        ; Jump table for monitor subroutines
                        ;
                        ;       All references to these labels should go through this
                        ;       so that changes in the actual routine's location in
                        ;       future versions of the monitor do not effect non-monitor
                        ;       programs.  These locations will never change.
                        ;
     40 c3 9802                 jmp     CI      ;char returned in A register
     43 c3 aa02                 jmp     CO      ;char passed in A register
     46 c3 c702                 jmp     crlf    ;prints (cr) (lf)
     49 c3 de02                 jmp     ghw     ;word ret in h&l or cy=1 & bad char in A
     4c c3 f502                 jmp     ghb     ;byte ret in A or CY=1 & bad char in A
     4f c3  a03                 jmp     ghd     ;digit ret in A or CY=1 & bad char in A
     52 c3 2403                 jmp     msg     ;address of EOL terminated msg in d&e
     55 c3 5103                 jmp     phw     ;word passed in h&l
     58 c3 5c03                 jmp     phb     ;byte passed in A
     5b c3 6c03                 jmp     phd     ;digit passed in A
     5e c3 9003                 jmp     space   ;print space
     61 c3 9803                 jmp     sub16   ;(h&l) <- (h&l) - (d&e)
     64 c3 a403                 jmp     ucase   ;upper to lower case conversion
     67 c3  d03                 jmp     ath     ;ascii to hex conversion
     6a c3 df00                 jmp     comnd   ;beginning of monitor command loop
                        ;       jmp     cmp16   ;uncomment when cmp16 routine included

                        ; Power-up and Reset initialization
                        ;
                        ;Twiddle, twiddle little thumbs. . . .
                        ;
     6d                 entry:          equ     $               ;
                        ;
                        ; hardware              mvi     a,10            ;
```

```
                ; delay       is      twiddle:call    d1Øms           ;delay 1Ø mS
                ; long                        dcr     a               ;
                ; enough              jnz     twiddle         ;
                ;
                ; now initialize usart chip
                ;
 6d 3e 82               mvi     a,Ø82h  ;force usart to expect command word
 6f d3  1               out     sercon
 71 3e 4Ø               mvi     a,Ø4Øh  ;now make usart to expect mode word
 73 d3  1               out     sercon
                ;
 75 3e ce               mvi     a,Øceh  ;mode byte -
 77 d3  1               out     sercon  ;  11 ØØ 11 1Ø
                ;                       ;   |  |  |  | ------- X16 clock
                ;                       ;   |  |  --------- 8 bits of data
                ;                       ;   |  ------------ no parity
                ;                       ;   --------------- 2 stop bit
                ;
 79 3e 37               mvi     a,Ø37h  ;command byte -
 7b d3  1               out     sercon  ;  Ø Ø 1 1 Ø 1 1 1
                ;                       ;  | | | | | | | -- xmit enable
                ;                       ;  | | | | | | ---- dtr/ = 1
                ;                       ;  | | | | | ------ rcvr enable
                ;                       ;  | | | | -------- norm op. (not break)
                ;                       ;  | | | ---------- reset error flags
                ;                       ;  | | ------------ rts/ = 1
                ;                       ;  | -------------- 1 = internal reset
                ;                       ;  ---------------- asyncronous mode
                ;
                ; Calculate baud rate assuming user types a control-d
                ;
                ; *****        DSR/ must be connected to RxD.                      *
                ;                                                                  *
                ;        These could be connected on the RS232 side of the        *
                ;     1489 RS232 receiver.  On the SCCS-85 be sure to cut the     *
                ;     trace on the bottom of the board connecting the RS232       *
                ;     dataset ready input to the +12 volt power supply.           *
                ;        An RS232 transmiter is only intended to drive one        *
                ;     RS232 receiver.  For this reason it would be better to      *
                ;     to connect these at TTL signal levels between the 1489      *
                ;     and the 8251.  This can be done by jumpering pins 3         *
                ;     and 22 on the 8251.  Be sure to disconnect the output       *
                ;     of the 1489 that was originally driving the 8251 DSR/       *
                ;     pin.  Either bend pin 3 up on the 1489 so that it does      *
                ;     not go in the socket or cut the trace from the 1489 to      *
                ;     the 8251.                                                   *
                ;
 7d db  1       baud1:          in      sercon          ;wait for line to drop
 7f  7                  rlc                     ;
 8Ø d2 7dØØ             jnc     baud1           ;
                ;
 83 3e 7Ø               mvi     a,Ø7Øh          ;set up timer to time next 4 bits
 85 d3 23               out     timctl          ;
 87 3e dd               mvi     a,-35   ; 8Ø85  ;34 cycles not counted by timer
                ;       mvi     a,-41   ; 8Ø8Ø  ;4Ø cycles not counted by timer
 89 d3 21               out     timel           ;  plus one for +1 after 1's comp
 8b 3e ff               mvi     a,Øffh          ;
 8d d3 21               out     timel           ;
                ;
                ;       at baud1:       ;avg time out of loop after drop.
                ;                       ;  loop 24 cycles long...............-    12
                ;                       ;rlc.................................-     4
                ;                       ;jnc    baud1...with cy=Ø 8Ø85/8Ø8Ø - 7/1Ø
                ;                       ;mvi    a,Ø7Øh.......................-     7
                ;                       ;out    timctl.......................-    1Ø
                ;                       ;mvi    a,-36........................-     7
                ;                       ;out    timel........................-    1Ø
                ;                       ;mvi    a,Øffh.......................-     7
                ;                       ;out    timel........................-    1Ø
                ;
                ;       at baud4:       ;avg time into loop since rose.
                ;                       ;  loop 24 cycles long...............+    12
                ;                       ;rlc.................................+     4
                ;                       ;jnc    baud4   with cy=Ø 8Ø85/8Ø8Ø + 7/1Ø
```

```
                    ;               ;mvi    a,40h.......................+     7
                    ;               ;out    timctl......................+    10
                    ;               ;                                     ------
                    ;               ;total cycles not counted by timer..-34/40
                    ;               ;                                    0085/00
                    ;
8f db  1      baud3:        in      sercon          ;wait for line to rise
91  7               rlc                     ;
92 da 8f00          jc      baud3           ;
95 db  1      baud4:        in      sercon          ;wait for line to drop
97  7               rlc                     ;
98 d2 9800          jnc     baud4           ;
                    ;
9b 3e 40            mvi     a,40h           ;counter latching command
9d d3 23            out     timctl          ;
9f 3e 70            mvi     a,70h           ;set up timer to read lsb,msb
a1 d3 23            out     timctl          ;
a3 db 21            in      timel           ;get lsb of count
a5 6f               mov     l,a             ;
a6 db 21            in      timel           ;get msb of count
a8 67               mov     h,a             ;
a9 7d               mov     a,l             ;compliment count - don't need to
aa 2f               cma                     ;  add one because taken into
ab 6f               mov     l,a             ;  account in initial load of
ac 7c               mov     a,h             ;  counter.
ad 2f               cma                     ;
ae 67               mov     h,a             ;
                    ;
af  e  6            mvi     c,6             ;shift count right 6
b1 b7         baud5:        ora     a               ;cy=0 to come in left end of H
b2 7c               mov     a,h             ;
b3 1f               rar                     ;
b4 67               mov     h,a             ;
b5 7d               mov     a,l             ;
b6 1f               rar                     ;
b7 6f               mov     l,a             ;
b8  d               dcr     c               ;
b9 c2 b100          jnz     baud5           ;
bc ce  0            aci     0               ;round
be 6f               mov     l,a             ;
bf 7c               mov     a,h             ;propagate possible round-up
c0 ce  0            aci     0               ;  carry into H.
c2 67               mov     h,a             ;
                    ;
                    ; initialize timer chip         to generate 16X baudrate for
                    ;
c3 3e 76            mvi     a,76h           ;init timer 1 to divide by n
c5 d3 23            out     timctl          ;
c7 7d               mov     a,l             ;
c8 d3 21            out     timel           ;
ca 7c               mov     a,h             ;
cb d3 21            out     timel           ;
                    ;
                    ; Initialize "answer"
                    ;
cd af               xra     a               ; on power up answer is blank
ce 32 fd1f          sta     answer          ;
d1 32 fe1f          sta     echofl          ; 0=echo 1=no echo
d4 32 ff1f          sta     dlyram          ; number of 10ms delays on <CR>
                    ;
                    ; Print         startup message
                    ;
d7 11 f503          lxi     d,start         ;print startup message
da cd 2403          call    msg             ;
                    ;
dd db  0            in      serdat          ;eat possible garbage character
                    ;
                    ; COMMAND LEVEL         - get character; jump to appropriate routine

df            comnd:        equ     $

df 11 f003          lxi     d,prmpt ;print command prompt
e2 cd 2403          call    msg
                    ;
```

```
e5 cd 9802          call    CI      ;
e8 cd 9003          call    space   ;
               ;    ani     7fh     ;put in if ucase taken out
eb cd a403          call    ucase   ;convert low to up case & strips parity
               ;
ee fe  d            cpi     0dh     ;special case, (cr) is nop that does not
f0 ca df00          jz      comnd   ;  clear the answer
               ;
f3 11 df00          lxi     d,comnd ;addr for pseudo call completed by pchl
f6 d5               push    d       ;
               ;
f7 fe 3f            cpi     '?'     ;special case '?', must not clear
f9 ca 1b01          jz      ask     ; answer first.
               ;
fc 21 c503          lxi     h,cmds  ;scan command table
ff be          cmdnxt:      cmp     m       ;
100 ca  f01         jz      cmdfnd  ;if matches go process
103 23              inx     h       ;
104 23              inx     h       ;
105 23              inx     h       ;
106 46              mov     b,m     ;check for end of table
107  5              dcr     b       ;
108 f2 ff00          jp      cmdnxt  ;not end...try next entry
               ;
10b cd 7f03 error:          call    prbad   ;print error message and return.  "comnd"
10e c9              ret             ;  is on stack as return addr for command
               ;
10f            cmdfnd:      equ     $
10f f5              push    psw
110 af              xra     a       ;clear answer
111 32 fd1f         sta     answer  ;
114 f1              pop     psw
115 23              inx     h       ;get address
116 5e              mov     e,m     ;
117 23              inx     h       ;
118 56              mov     d,m     ;
119 eb              xchg            ;
11a e9              pchl            ;
               ;
               ;****************** end       of command level ***********************
               ;
               ;*****************************************************************
               ;
               ;    print   one byte note left by last command
               ;
11b cd 9003 ask:    call    space
11e 3a fd1f         lda     answer
121 cd aa02         call    CO
124 c9              ret
               ;
               ;***************** end of 20 questions *************************
               ;
               ;*****************************************************************
               ;
               ; GOTO routine - starts      execution in memory location
               ;
125 cd de02 goto:   call    ghw     ;get hex word
128 da  b01         jc      error   ;
12b cd 3403         call    okck    ;
12e d8              rc              ;
               ;
12f 31 fd1f         lxi     sp,STACKINIT    ;initialize stack pointer
132 e9              pchl            ;jmp to location, return addr is on stack
               ;
               ;*************** end of      goto routine ***************************
               ;
               ;
               ;*****************************************************************
               ;
               ; Memory editor      routine
               ;
               ; MEMED        - Hexadecimal Memory Editor
               ;
               ;      1) Computer types (cr),(lf),"("
```

```
                 ;       2) User enters one of the following:
                 ;          a) Valid Hex  Word  (four hex  digits)  -  goto
                 ;             step 3
                 ;          b) "/" - Exit editor by doing a "ret"
                 ;          c) Anything else - type " what ?" and
                 ;             goto 1
                 ;       3) Computer types ") = xx ", where xx is the  con-
                 ;          tents (in hex) of the memory byte addressed.
                 ;       4) The  user  now has  a number  of things  he can
                 ;          type:
                 ;          a) Valid Hex Byte (two hex digits) -  overwrite
                 ;             the  memory  location  addressed  with  this
                 ;             byte.  Then read  another  character  from
                 ;             the user and continue with 4b.
                 ;
                 ;          b) A non-hex character - do one of the  follow-
                 ;             ing:
                 ;
                 ;             i)    (cr) or " " - Address the next  sequen-
                 ;                   tial location and print the address and
                 ;                   contents like this:
                 ;
                 ;                           (cr),(lf),"(addr) = xx "
                 ;
                 ;             ii)   '.' - Re-display the same location like
                 ;                   this:
                 ;
                 ;                           (cr),(lf),"(addr) = xx "
                 ;
                 ;             iii) "-" - Address the next  previous  loca-
                 ;                   tion and print the address and contents
                 ;                   like this:
                 ;
                 ;                           (cr),(lf),"(addr) = xx "
                 ;
                 ;             iv)   "/" - Goto step 1 and read  a  new  ad-
                 ;                   dress
                 ;
                 ;             v)      Anything  else  -  Type  "  what
                 ;                   ?" and treat like a "."
                 ;
                 ;        Note - If option "a" is not executed  then  memory
                 ;        is not altered.
                 ;
133 11 1b00 memed:           lxi     d,edm2  ;print "cr, lf, ("
136 cd 2403          call    msg

139 cd de02          call    ghw
13c d2 4801          jnc     ok      ;get hex word into HL, jump if valid

13f fe 2f            cpi     '/'     ;bad char received - was it "/"
141 c8               rz              ;go back to command level if so

142 cd 7f03          call    prbad   ;print "what ?"
145 c3 3301          jmp     memed   ;then try again

148 cd 9901 ok:      call    discon  ;display contents of location
14b cd 5101          call    edit    ;then begin editing
14e c3 3301          jmp     memed   ;loupe if edit returns
                 ;
                 ;       end     memed
                 ;
                 ;
                 ; Get either a new hex byte to be written where      HL points,
                 ; followed by another command, or just another command.
                 ;
151 cd f502 edit: call   ghb     ;get the new hex byte if typed
154 da 6001          jc      next    ;jmp if other than hex byte received
157 77               mov     m,a     ;else store it in memory
158 cd 9003          call    space   ;space to reinforce that once two digits
                 ;                       ;  are entered, location is changed.
15b cd 9802          call    CI      ;and get another char & echo it
15e e6 7f            ani     7fh     ;kill top bit
                 ;
```

```
160 fe  d    next: cpi     0dh     ;carriage return?
162 c2 6901        jnz     e1
165 23             inx     h
166 c3 8601        jmp     pr      ;yes- print NEXT location
                   ;
169 fe 20    e1:   cpi     ' '     ;or blank
16b c2 7201        jnz     e2
16e 23             inx     h
16f c3 8601        jmp     pr      ;yes- do the same
                   ;
172 fe 2e    e2:   cpi     '.'     ; period?
174 ca 8601        jz      pr      ;print current location
                   ;
177 fe 2d    e3:   cpi     '-'     ; dash?
179 c2 8001        jnz     e4
17c 2b             dcx     h
17d c3 8601        jmp     pr      ;yes - print previous location
                   ;
180 fe 2f    e4:   cpi     '/'     ;slash?
182 c8             rz              ;edit all done if so
                   ;
183 cd 7f03        call    prbad   ;if none of the above, print "what ?"
                   ;
186 cd 8c01  pr:   call    dismem  ;display the new current memory location
189 c3 5101        jmp     edit    ;and loop

             ; Print         CR, LF then an ( followed by the contents of HL in hex.

18c 11 1b00  dismem:       lxi     d,edm2  ;do cr,lf, "("
18f cd 2403        call    msg
192 cd 5103        call    phw
195 cd 9901        call    discon
198 c9             ret
             ;
             ; **** discon ****
             ;
             ; print        ') = ' followed by the contents of the memory loc.
             ; pointed to by        HL
             ;
199 11 1300  discon:       lxi     d,edm1
19c cd 2403        call    msg
19f 7e             mov     a,m     ;get contents of mem loc.
1a0 cd 5c03        call    phb     ;print it
1a3 cd 9003        call    space
1a6 c9             ret
             ;
             ;
             ;***********************        end of memory editor ********************
             ;
             ;******************************************************************************
             ;
             ; Hex-format loader
             ;
             ;       ":"     record mark
             ;       xx      record length - number of data bytes
             ;       xxxx    load address of first byte, remaining bytes in
             ;               record go sequentially
             ;       xx      record type - "00" = data, "01" = eof
             ;       ...     data bytes
             ;       xx      check sum such that sum of ALL hex bytes,
             ;               including checksum = 0
             ;
             ;       NOTE:   record length = 00 taken as eof
             ;
1a7 32 fe1f  loader:       sta     echofl  ;non-zero value (1) turns off echo
             ;                     ;
1aa cd c201  load1:        call    getrec  ;read in one rec, (a) = record length
1ad b7             ora     a       ;set z-flag on record length
1ae 3e 47          mvi     a,'G'   ;answer to question = Good
1b0 ca ba01        jz      done    ;if length = 0 then done
             ;
1b3 7a             mov     a,d     ;(d) = error flag on getrec return
1b4 b7             ora     a       ;see if the "error" flag is non-zero.
1b5 ca aa01        jz      load1   ;if not, go do next record
```

```
                        ;
1b8 3e 42               mvi     a,'B'   ;store "Bad" flag in answer to question
1ba 32 fd1f     done:   sta     answer  ;
1bd af                  xra     a       ;zero echofl to turn off echo
1be 32 fe1f             sta     echofl  ;
1c1 c9                  ret             ;return to command level
                        ;
                        ;       end     loader
                        ;
                        ;
                        ; *** getrec *** read in one record
                        ;
1c2 cd dc01     getrec:         call    fndmark  ;skip to record mark
                        ;
1c5 cd f601             call    lghb    ;get the record length
1c8 4f                  mov     c,a     ;  into the C reg.
                        ;
1c9 cd f601             call    lghb    ;get load address field into h & l
1cc 67                  mov     h,a     ;
1cd cd f601             call    lghb    ;
1d0 6f                  mov     l,a     ;
                        ;
1d1 cd f601             call    lghb    ;get the record-type byte and ignore
                        ;
1d4 cd e901             call    data    ;put the next (C) bytes into memory
                        ;                ;starting where HL points
1d7 cd f601             call    lghb    ;read the checksum byte
                        ;
1da 79                  mov     a,c     ;put the record length back into A reg.
                        ;
1db c9                  ret             ;return from getrec. (d) contains the
                        ;                ;  sum off all hex bytes read, and so
                        ;                ;  is effectively an error flag
                        ;       end     getrec
                        ;
                        ;
                        ; *** fndmark *** - find        record mark
                        ;                 ignores all text until ":" found, then ret
                        ;
1dc cd 9802     fndmark:        call    CI      ;get character
1df e6 7f               ani     07fh    ;strip off 8th bit
1e1 fe 3a               cpi     ':'     ;
1e3 c2 dc01             jnz     fndmark ;not record mark - get next char
                        ;
1e6 16  0               mvi     d,0     ;clear D register (error accumulator)
1e8 c9                  ret             ;
                        ;
                        ;       end     fndmark
                        ;
                        ; *** data *** - input all data        bytes
                        ;               (c) = number of bytes to read in
                        ;               (d) = error flag accumulator maintained by lghb
                        ;
1e9 41          data:   mov     b,c     ;copy C reg. to B
1ea 78          loop:   mov     a,b     ;get remaining byte count
1eb b7                  ora     a       ;get flags
1ec c8                  rz              ;return from subr. if none left
1ed  5                  dcr     b       ;else decrement b reg.
1ee cd f601             call    lghb    ;get byte from data field
1f1 77                  mov     m,a     ;store in memory
1f2 23                  inx     h       ;bump pointer
1f3 c3 ea01             jmp     loop    ;go back for next char.
                        ;
                        ;       end     data
                        ;
                        ;
                        ; *** lghb *** - loader        get hex byte
                        ;               same as ghb except adds byte gotten to error
                        ;               accumulator in d register
                        ;
1f6 cd f502     lghb:   call    ghb     ;get byte
1f9 f5                  push    psw     ;save byte
1fa 82                  add     d       ;add to (d)
1fb 57                  mov     d,a     ;put sum in d-reg
```

```
1fc f1                pop     psw     ;restore byte
1fd c9                ret             ;
                ;
                ;       end     lghb
                ;
                ;************* end of loader ******************************************
                ;
                ;**********************************************************************
                ;
                ;       Common code for dump and punch routine.
                ;         Must not destroy a-register
                ;
1fe 11 e903     d_p:    lxi     d,plo   ;prompt for lo limit
201 cd 2403             call    msg
204 cd de02             call    ghw
207 da  b01             jc      error   ;jump if error
20a 11  b00             lxi     d,phi   ;prompt for hi limit
20d cd 2403             call    msg
210 eb                  xchg
211 cd de02             call    ghw
214 eb                  xchg
215 da  b01             jc      error
218 cd 3403             call    okck
21b d8                  rc              ;return if aborted by okck call
                ;
                ; (h&l)         = beginning address    (d&e) = ending address
                ;
21c e5                  push    h       ;
21d cd 9803             call    sub16   ;calc number of bytes to be processed
220 eb                  xchg            ;
221 e1                  pop     h       ;
222 1b                  dcx     d       ;d&e = number of bytes
                ;
                ; Call routine originally requested
                ;
223 fe 50               cpi     'P'
225 ca 5502             jz      punch
                ;       else    dump
                ;
                ; Dump routine
                ;
228 cd c702     dump:   call    crlf    ;go to new line
22b cd 5103             call    phw     ;print memory address
22e 7d                  mov     a,l     ;make locations with same lower 4 bits
22f e6  f               ani     WIDTH   ;  land in same columns in first line
231 4f                  mov     c,a     ;  as in other lines
232 87                  add     a       ;multiply by 3...
233 81                  add     c       ;
234 4f                  mov     c,a     ;move count to c
235 3e 20               mvi     a,' '   ;space over to appropriate column
237 cd 8603             call    repeat  ;print (c) (a) times

23a cd 9003     dl1:    call    space
23d 7e                  mov     a,m     ;get byte
23e cd 5c03             call    phb     ;print it
241 23                  inx     h       ;point to next byte
242 13                  inx     d       ;decrement count of number of bytes left
243 7b                  mov     a,e     ;
244 b2                  ora     d       ;
245 c8                  rz              ;return if zero left
246 7d                  mov     a,l
247 e6  f               ani     WIDTH   ;print crlf on multiple of 16
249 c2 3a02             jnz     dl1

24c cd c702             call    crlf    ;go to new line
24f cd 5103             call    phw     ;print memory address
252 c3 3a02             jmp     dl1     ;
                ;****************** end of dump ********************************
                ;
                ; *** Punch Hex        Tape in INTEL format  ***
                ;
                ;preliminary processing        done at d_p
                ;
```

```
255 cd c702 punch:        call    crlf           ;
258 3e 3a         mvi     a,':'          ;print record mark
25a cd aa02       call    CO             ;
25d 7b            mov     a,e            ;calc number of bytes in record
25e 2f            cma                    ; & start accumulating check sum
25f e6  f         ani     WIDTH          ;
261 3c            inr     a              ;
262 cd 5c03       call    phb            ;print number of bytes in record
265 84            add     h              ;add load address to check sum
266 85            add     l              ;
267 47            mov     b,a            ;initialize checksum
268 cd 5103       call    phw            ;load address
26b af            xra     a              ;
26c cd 5c03       call    phb            ;record type
                ;
26f 7e     pnxtbyt:mov    a,m            ;
270 cd 5c03       call    phb            ;
273 80            add     b              ;accumulate checksum
274 47            mov     b,a            ;
275 13            inx     d              ;
276 7b            mov     a,e            ;
277 b2            ora     d              ;
278 ca 8b02       jz      pdone          ;
27b 23            inx     h              ;
27c 7b            mov     a,e            ;test for end of record
27d e6  f         ani     WIDTH          ;
27f c2 6f02       jnz     pnxtbyt        ;
                ;
282 78            mov     a,b            ;end of record processing
283 2f            cma                    ;compliment checksum
284 3c            inr     a              ;
285 cd 5c03       call    phb            ;
288 c3 5502       jmp     punch          ;
                ;
28b 78     pdone:         mov     a,b            ;compliment last checksum
28c 2f            cma                    ;
28d 3c            inr     a              ;
28e cd 5c03       call    phb            ;
291 11 d503       lxi     d,endrec       ;
294 cd 2403       call    msg            ;
297 c9            ret                    ;
                ;                        ;
                ;         end     punch  ;
                ;
                ;
                ;**************************************************************
                ;
                ;         UTILITY ROUTINES - in alphabetical order (sort of)
                ;
                ;**************************************************************
                ;
                ; I/O routines
                ;
                ;
298 db  1  CI:    in      sercon         ;wait for data ready
29a e6  2         ani     2              ;
29c ca 9802       jz      CI             ;
29f db  0         in      serdat         ;get byte
2a1 f5            push    psw            ;save psw
2a2 3a fe1f       lda     echofl         ;check echo flag
2a5 b7            ora     a              ;
2a6 c2 c502       jnz     c3             ;if not zero echo-ret on CO
2a9 f1            pop     psw            ;echo character
                ;
                ; *****       CI must be directly followed by CO so that it can drop
                ;         through !!! *****
                ;
                ;**** CO Console Output        - destroys only flags...
                ;
                ;                        ...char passed in a register
2aa f5     CO:    push    psw            ;
2ab db  1  c1:    in      sercon         ;
2ad  f            rrc                    ;
2ae d2 ab02       jnc     c1             ;
```

```
2b1 f1               pop     psw                 ;
2b2 d3   0           out     serdat              ;
2b4 fe   d           cpi     0dh                 ;if cr then delay
2b6 c0               rnz                         ;
2b7 f5               push    psw                 ;
2b8 3a ff1f          lda     dlyram              ;
2bb 3d        c2:    dcr     a                   ;
2bc fa c502          jm      c3                  ;
2bf cd d002          call    d10ms               ;delay 10mS
2c2 c3 bb02          jmp     c2                  ;
2c5 f1        c3:    pop     psw                 ;used by CI - do not change
2c6 c9               ret
              ;
              ;;****** cmp16 ** 16 bit compare h&l and d&e *******************
              ;;
              ;;       if( h&l = d&e ) z=1, cy=0       *** crafty and very   ***
              ;;       if( h&l > d&e ) z=0, cy=0       *** useful routine if ***
              ;;       if( h&l < d&e ) z=0, cy=1       *** ever room         ***
              ;;
              ;cmp16:           push    h                  ;save psw & h&l
              ;        push     psw          ;
              ;        mov      a,h          ;if h != d enough info found
              ;        sub      d            ;
              ;        jnz      cmp16e       ;
              ;        mov      a,l          ;if h=d then compare lower bytes
              ;        sub      e            ;
              ;cmp16e: pop      h            ;
              ;        mov      a,h          ;
              ;        pop      h            ;
              ;        ret                   ;
              ;;                             ;
              ;;       end      cmd16        ;
              ;
2c7 d5        crlf:  push    d
2c8 11 ba03          lxi     d,mcrlf
2cb cd 2403          call    msg
2ce d1               pop     d
2cf c9               ret
              ;
              ; d10ms          - Delay 10 mS
              ;
2d0 e5        d10ms:         push    h        ;
2d1 f5               push    psw     ;
2d2 21   103         lxi     h,769   ;
2d5 7d        dtwidl:        mov     a,l      ;    ;~0.01 seconds on a 4 MHz 8085      5
2d6 b4               ora     h       ;    ;                                      4
2d7 2b               dcx     h       ;    ;                                     10
2d8 c2 d502          jnz     dtwidl  ;    ;              8085/8080            7/10
2db f1               pop     psw     ;    ;              total               26/29
2dc e1               pop     h       ;
2dd c9               ret             ;
              ;                      ;
              ;      end     d10ms   ;
              ;
              ; GHW -        Get Hex Word
              ;
              ;      Read 4 hex digits frm terminal & convert to 16 bit word
              ;
              ;      INPUT  : None
              ;      OUTPUT : if (no non-hex charaters typed)
              ;
              ;                       (h&l) = hex word typed
              ;                       (a)   = garbage
              ;                       CY    = 0
              ;               else
              ;                       (h&l) = garbage
              ;                       (a)   = bad character as received from CO
              ;                       CY    = 1
              ;
              ;      REGISTERS CHANGED: h, l, flags
              ;
2de c5        ghw:   push    b
2df f5               push    psw
2e0 cd f502          call    ghb                ; get first byte in a-register
```

```
2e3 da f202        jc      ghwend             ; return if bad char
2e6 67             mov     h,a                ; move byte to final destination
2e7 cd f502        call    ghb                ; get second byte
2ea da f202        jc      ghwend             ;
2ed 6f             mov     l,a                ;
2ee c1             pop     b                  ;
2ef 78             mov     a,b                ;
2f0 c1             pop     b                  ;
2f1 c9             ret                        ;
2f2 c1     ghwend:         pop     b          ;
2f3 c1             pop     b                  ; do NOT restore a
2f4 c9             ret                        ;
           ;                                  ;
           ;       end     ghw                ;
           ;
           ;
           ; GHB -         Get Hex Byte
           ;
           ;       Read 2 hex digits from terminal & convert to 8 bit word
           ;
           ;       INPUT  : None
           ;       OUTPUT : if (no non-hex charaters typed)
           ;
           ;                       (a)    = Hex byte typed
           ;                       CY     = 0
           ;               else
           ;                       (a)    = bad character as received from CO
           ;                       CY     = 1
           ;
           ;       REGISTERS CHANGED: a, flags
           ;
2f5 c5     ghb:    push    b                  ; save b&c
2f6 cd  a03        call    ghd                ; get first hex digit in a-reg
2f9 da  803        jc      ghbend             ; if bad char quit and pass back
2fc  7             rlc                        ; shift to upper half of byte
2fd  7             rlc                        ;    .
2fe  7             rlc                        ;    .
2ff  7             rlc                        ;    .
300 47             mov     b,a                ; save first digit
301 cd  a03        call    ghd                ; get second digit
304 da  803        jc      ghbend             ; bad char read, ret it to caller
           ;                                  ;
307 b0             ora     b                  ; combine first and second digits
           ;                                  ;
308 c1     ghbend:         pop     b                ; restore original b&c
309 c9             ret                        ;
           ;                                  ;
           ;       end     ghb                ;
           ;
           ;
           ; GHD -         Get Hex Digit
           ;
           ;       Read 1 hex digit from terminal & convert to 4 bit nibble
           ;
           ;       INPUT  : None
           ;
           ; ATH -         Ascii To Hex
           ;
           ;       Alternate entry point does not call CI.  User passes
           ;       character to be converted in a-register.
           ;
           ;       INPUT   : character to be converted in a-register
           ;
           ; Both:
           ;
           ;       OUTPUT : if (valid hex character typed)
           ;
           ;                       (a)    = 0xh, x = hex digit typed
           ;                       CY     = 0
           ;               else
           ;                       (a)    = bad character as received from CO
           ;                       CY     = 1
           ;
           ;       REGISTERS CHANGED: a, c, flags
```

```
                      ;
30a cd 9802   ghd:    call    CI                ; get character & echo
              ;       ani     07fh              ; put in if ucase taken out
30d cd a403   ath:    call    ucase             ; map lower to upper case and
              ;                                 ;   strip parity.
310 fe 30             cpi     '0'
312 d8                rc                        ;       non-hex character
313 fe 3a             cpi     ':'               ; if  (a) =< '9'+1
315 da 2103           jc      ghd2              ;       '0'-'9' typed - convert
318 fe 41             cpi     'A'               ; if  (a) < 'A'
31a d8                rc                        ;       non-hex character
31b fe 47             cpi     'G'               ; if  (a) >= 'G'
31d 3f                cmc                       ;       .
31e d8                rc                        ;       non-hex character
31f d6  7             sui     07h               ; shift 'A'-'F' down
321 d6 30     ghd2:   sui     '0'               ; convert
323 c9                ret                       ;
              ;                                 ;
              ;       end     ghd               ;
              ;
              ; Subroutine to       print message pointed to by DE and
              ; terminated by       EOL byte.
              ;       DE left pointing to the byte following the EOL so that
              ;       strings of messages can be easily printed if they are
              ;       in sequntial memory.
              ;       *       No other registers destroyed *

324 f5        msg:    push    psw
325 1a        loupe:          ldax    d       ;get char
326 fe ff             cpi     EOL     ;end of string?
328 13                inx     d       ;bump pointer
329 ca 3203           jz      mdn     ;jump if so
32c cd aa02           call    CO      ;else print it
32f c3 2503           jmp     loupe   ;do it again
332 f1        mdn:    pop     psw
333 c9                ret
              ;
              ;
              ; routine to verify an entry
              ;
334 d5        okck:   push    d
335 f5                push    psw
336 11 e303           lxi     d,mok
339 cd 2403           call    msg
33c cd 9802           call    CI
33f e6 7f             ani     07fh
341 fe  d             cpi     0dh
343 ca 4d03           jz      okckend
346 11 b003           lxi     d,abort
349 cd 2403           call    msg
34c 37                stc
34d d1        okckend:pop     d
34e 7a                mov     a,d
34f d1                pop     d
350 c9                ret
              ;
              ;       end     okck
              ;
              ;
              ; PHW -       Print Hex Word
              ;
              ;       Convert 16 bit word to ascii and print
              ;
              ;       INPUT  : (h&l) = word to be printed
              ;       OUTPUT : None
              ;
              ;       REGISTERS CHANGED: None
              ;
351 f5        phw:    push    psw               ; save a-register and flags
352 7c                mov     a,h               ;
353 cd 5c03           call    phb               ; print high-order byte
356 7d                mov     a,l               ;
357 cd 5c03           call    phb               ; print low-order byte
35a f1                pop     psw               ; restore a-register and flags
```

```
35b c9                  ret                            ;
                  ;                                    ;
                  ;     end     phw                    ;
                  ;
                  ;
                  ; PHB -       Print Hex Byte
                  ;
                  ;     Convert 8 bit byte to ascii and print
                  ;
                  ;     INPUT  : (a) = Byte to be printed
                  ;     OUTPUT : None
                  ;
                  ;     REGISTERS CHANGED: Flags
                  ;
35c c5            phb:  push    b                      ; save b&c
35d 47                  mov     b,a                    ; save lower nibble
35e  f                  rrc                            ; shift to lower half of byte
35f  f                  rrc                            ;   .
36Ø  f                  rrc                            ;   .
361  f                  rrc                            ;   .
362 cd 6cØ3             call    phd                    ; print upper hex digit
365 78                  mov     a,b                    ; get lower nibble
366 cd 6cØ3             call    phd                    ; ...and print
369 78                  mov     a,b                    ; restore original byte to a
36a c1                  pop     b                      ; restore b&c
36b c9                  ret                            ;
                  ;                                    ;
                  ;     end     phb                    ;
                  ;
                  ;
                  ; PHD -       Print Hex Digit
                  ;
                  ;     Convert hex digit to ascii and print it
                  ;
                  ;     INPUT  : (a) = ?xh where x is the hex digit to be printed
                  ;                        the ? nibble is immaterial
                  ;     OUTPUT : None
                  ;
                  ;     REGISTERS CHANGED: flags
                  ;
36c c5            phd:  push    b                      ;save a&c
36d 47                  mov     b,a                    ;
36e e6  f               ani     Øfh                    ; mask off lower nibble
37Ø c6 3Ø               aci     'Ø'                    ; convert 'Ø'-'9' to ascii
372 fe 3a               cpi     '9'+1                  ; if 'Ø'-'9'
374 da 79Ø3             jc      phd1                   ;       then done
377 c6  7               adi     'A'-':'                ; convert 'A'-'F'
379 cd aaØ2 phd1:       call    CO                     ; print digit
37c 78                  mov     a,b                    ;restore registers
37d c1                  pop     b                      ;
37e c9                  ret                            ;
                  ;                                    ;
                  ;     end     phd                    ;
                  ;
                  ;
                  ; *****       prbad - print ' WHAT ?' **** DESTROYS D&E ****
                  ;
37f 11 bdØ3 prbad:              lxi     d,bad    ;
382 cd 24Ø3            call     msg     ;
385 c9                 ret              ;
                  ;
                  ;     end     prbad
                  ;
                  ;
                  ;Subroutine to print (a) (c) times
                  ;     uses a, c...(c) = Ø on exit
386  c            repeat:       inr     c        ;check for printing (c) Ø times
387  d                   dcr    c       ;
388 c8            repl: rz
389 cd aaØ2             call    CO
38c  d                  dcr     c
38d c3 88Ø3             jmp     repl
                  ;
                  ; *****       space ***** print space
```

```
                  ;
 390 f5           space:          push    psw
 391 3e 20              mvi       a,' '
 393 cd aa02            call      CO
 396 f1                 pop       psw
 397 c9                 ret
                  ;
                  ; *****        sub16 ***** 16 bit subtract (h&l) <- (h&l) - (d&e)
                  ;
                  ;      if    (d&e)  < (h&l)     CY = 1
                  ;      if    (d&e) >= (h&l)     CY = 0
                  ;
 398 d5           sub16:          push    d        ;
 399 f5                 push      psw     ;
 39a 7d                 mov       a,l     ;
 39b 93                 sub       e       ;
 39c 6f                 mov       l,a     ;
 39d 7c                 mov       a,h     ;
 39e 9a                 sbb       d       ;
 39f 67                 mov       h,a     ;
 3a0 d1                 pop       d       ;
 3a1 7a                 mov       a,d     ;
 3a2 d1                 pop       d       ;
 3a3 c9                 ret               ;
                  ;
                  ; UCASE         - subroutine which checks the A reg for a lower case
                  ; ASCII         letter. If one present, it is converted to upper case.
                  ; If not present, nothing done.        Strips parity first.
 3a4 e6 7f        ucase:          ani     07fh    ;strip parity
 3a6 fe 61              cpi       61h
 3a8 3f                 cmc
 3a9 d0                 rnc               ;don't convert if before 'a'
 3aa fe 7b              cpi       7bh
 3ac d0                 rnc               ;don't convert if after 'z'
 3ad d6 20              sui       20h     ;convert lower to upper
 3af c9                 ret
                  ;
                  ; ROM constant allocation - alphabetical order (sortof)
                  ;
                  abort:          db      ' ABORTED !'
 3b0 20           41 42 4f 52 54 45 44 20 21
                  mcrlf:          db      0dh,0ah,EOL
 3ba  d            a ff
                  bad:  db        ' WHAT ?'
 3bd 20           57 48 41 54 20 3f
 3c4 ff                 db        EOL
 3c5 4c           cmds: db        'L'                ;command table
 3c6 a7  1              dw        loader             ;
 3c8 45                 db        'E'                ;
 3c9 33  1              dw        memed              ;
 3cb 47                 db        'G'                ;
 3cc 25  1              dw        goto               ;
 3ce 44                 db        'D'                ;
 3cf fe  1              dw        d_p                ;common code for dump and punch
 3d1 50                 db        'P'                ;  commands.
 3d2 fe  1              dw        d_p                ;  .
 3d4  0                 db        0                  ;end of table mark
                  endrec:         db      0dh,0ah          ;end of record for punch
 3d5  d            a
                        db        ':00000001FF'      ;
 3d7 3a           30 30 30 30 30 30 30 31 46 46
 3e2 ff                 db        EOL
                  ;edm1:          db      ') = '           ;these lines were moved to
                  ;       db      EOL                ;5 extra bytes of RST 2
                  ;edm2:          db      0dh, 0ah         ;these lines were moved to
                  ;       db      '('                ;4 of the 5 bytes of RST 3
                  ;       db      EOL                ;
                  mok:  db        ' OK ?'
 3e3 20           4f 4b 20 3f
 3e8 ff                 db        EOL
                  ;ph1:  db       ' TO '             ;these lines were moved to
                  ;      db       EOL                ;5 extra bytes of RST 1
                  plo:  db        ' FROM '
 3e9 20           46 52 4f 4d 20
```

```
     3ef ff                db      EOL
                    prmpt:         db        0dh, 0ah
     3f0  d           a
                           db      ' >'
     3f2 20         3e
     3f4 ff                db      EOL
                    start:         db        0dh,0ah
     3f5  d           a
                           db      'M3FF.E'
     3f7 4d         33 46 46 2e 45
                           db      0dh,0ah,EOL
     3fd  d           a ff

                    ;
                    ; RAM allocation if alphabetical order
                    ;
    1ffd                   org     MEMTOP-2          ;MEMTOP - (# bytes alloc - 1)
                    ;
    1ffd            STACKINIT      equ       $       ;initial stack pointer overlaps
                    ;                                ;lowest byte allocated.
                    ;
                    answer         ds        1                ;answer to question
                    echofl         ds        1                ;echo flag: 0=echo 1=no echo
                    dlyram         ds        1                ;number of 10mS delays on <CR>
                    ;                                ;range: 00h to 80h
                    ;
                    ; At this point       $ should = MEMTOP
                    ;
                           end
```

```
   0 BASE                 298 CI                     2aa CO
  ff EOL                 1fff MEMTOP                1000 RAM
1ffd STACKINIT              f WIDTH                  3b0 abort
1ffd answer               11b ask                    30d ath
 3bd bad                   7d baud1                   8f baud3
  95 baud4                 b1 baud5                  2ab c1
 2bb c2                   2c5 c3                     10f cmdfnd
  ff cmdnxt               3c5 cmds                    df comnd
 2c7 crlf                 2d0 dl0ms                  1fe d_p
 1e9 data                 23a dl1                    199 discon
 18c dismem               1fff dlyram                1ba done
 2d5 dtwidl               228 dump                   169 e1
 172 e2                   177 e3                     180 e4
1ffe echofl               151 edit                    13 edm1
  1b edm2                 3d5 endrec                  6d entry
 10b error                1dc fndmrk                 1c2 getrec
 2f5 ghb                  308 ghbend                 30a ghd
 321 ghd2                 2de ghw                    2f2 ghwend
 125 goto                 1f6 lghb                   1aa load1
 1a7 loader               1ea loop                   325 loupe
 3ba mcrlf                332 mdn                    133 memed
 3e3 mok                  324 msg                    160 next
 148 ok                   334 okck                   34d okckend
 28b pdone                35c phb                    36c phd
 379 phd1                   b phi                    351 phw
 3e9 plo                  26f pnxtbyt                186 pr
 37f prbad                3f0 prmpt                  255 punch
 388 repl                 386 repeat                1000 rst0
1008 rst1                1010 rst2                  1018 rst3
1020 rst4                1028 rst5                  102c rst55
1030 rst6                1034 rst65                 1038 rst7
103c rst75                  1 sercon                   0 serdat
 390 space                3f5 start                  398 sub16
  23 timctl                21 timel                 1024 trap
 3a4 ucase
```

Information relating to active devices on SCCS-85

1488	data
1489	data
1N4733	data
2114	data
2708	data
2716	data

Programming Intel 2708s and 2716s

Note! Texas Instruments makes a "TMS2716" which is NOT compatable with an Intel 2716. The TI 2716 is a 2K x 8 version of the 2708, and requires three power supplies to operate and a 2708 like programming procedure. The Intel 2716 requires only a single 5 volt supply to operate and is programmed in a very differnt manner. See TI programming instructions if the TMS2716 is used.

8085	data
8080	mnemonic and opcode reference sheet
8085	Applications of MCS-85
8251	data
AP-16	Using the 8251 Universal Synchronous/Asyncronous Receiver/Trnsmitter
8253	data
8255	data
AP-15	8255 Programmable Peripheral Interface Applications
8257	data
74xx	Family TTL General Information
7400	data
7402	data
7454	data
74LS138	data
74257	data
74373	data

APPENDIX A2

MAIN PROGRAM

```
;                            --- MAIN ---

;***************************************************************************
;          This  program is written for control of ink jet printer.
;          This program uses data structure as shown in pattern program
;          to create a certain desired printing pattern.
;          For a desired printing pattern, only a new pattern program
;          is needed.
;          For programming a new pattern program, understanding of this
;          MAIN program is necessary.
;
;***************************************************************************

        ;There are three ports for the microcomputer:
        ;Port A:  addr 0010h, output port.
        ;Port B:  addr 0011h, input port.
        ;Port C:  addr 0012h, handshaking port, lower 4 bits is
        ;         output, higher 4 bits is input.
        ;Control register for the ports above: addr 0013h.

        ;x-axis enable: 0001h
        ;y-axis enable: 0002h
        ;negative direction: 002dh
        ;positive direction: 0000h
;*********************************************************************

          org     1900h
          mvi     a,8ah  ;select input and output ports, 8a means that port
                          a:out,b:in,c:in, lower out
          out     13h    ;set input output ports, control register is at 13h.
          mvi     a,0Ch
          out     12h     ;clear enables (port c) and head control
          lxi     h,1050h ;load data pointer
          call    bdset  ;set stepper board parameters (x-axis)
          call    bdset   ;set stepper board parameters (y-axis)

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

main      mov     a,m     ;move one byte of data from memory to acumulator.
          cpi     002eh   ;check for end of file,.=2e (2e is the end flag
                           of data program)
          jz      006ah   ;if end of file,jump to moniter.
          call    load    ;load index values
          call    goto    ;move table to desired location by x and y index value.
          call    load    ;load length and width (load x-dim and y-dim value)

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;the next four commands pick the shorter dimension for jog axis.
;jog: move without print.

          mov     a,d     ;compare upper half of x-dim and y-dim only.
          cmp     b       ;sets carry if a<b (x<y),if a>=b sets nocarry (y<x)
```

```
        jz      xjog     ;if x=y jump to xjog
        jnc     yjog     ;no carry indicates x>or=y

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

xjog    push    d   ;store y-dim on stack.
        mov     d,b    ;move x-dim from b c to d e register.
        mov     e,c
        pop     b        ;now x and y have switched reg.
        push    b      ;store y-dim to b c register.

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

boxy    mvi     b,002dh    ;- move (002d means negative direction
                               in ASCII code)
        mvi     c,02h      ;y axis (select y-axis enable)
        call    movel      ;printing sweep
        pop     b       ;pop x-dim from stack.
        call    test       ;check if x-dim is zero,test is here for lines.
        push    b     ;push x-dim on stack.
        mvi     c,01h      ;x axis (select x-axis enable)
        call    jog      ;(do jogging)
        pop     b      ;(pop x-dim)
        call    min5       ;decrement jog value (x-dim value)
        call    test       ;check if zero,if so get new data
        push    b     ;push x-dim on stack.
        mvi     b,00h ;+ move, positive direction.
        mvi     c,02h    ;select y-axis enable.
        call    movel       ;print sweep other direction
        mvi     c,01h      ; select x-axis enable.
        call    jog     ;do jogging.
        pop     b     ; pop x-dim from stack
        call    min5       ;decrement jog value, decrement x-dim value by 5
        call    test       ;check if zero
        push    b     ;store x-dim value
        jmp     boxy

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

   ;print retangle, y-axis is jog axis

yjog    push    b          ;print rectangle,y axis is jog axis
boxx    mvi     b,002dh    ;- move
        mvi     c,01h      ;enable x axis
        call    movel      ;printing sweep
        pop     b     ; pop y-dim from stack
        call    test       ;test if zero,for lines
        push    b     ;push y-dim from stack
        mvi     c,02h      ;enable y axis
        call    jog    ;do y-dim jogging
        pop     b    ;pop y-dim
        call    min5       ;decrement jog value, decrement y-dim by 5
        call    test       ;check if zero
```

```
        push    b     ;push y-dim on stack
        mvi     b,00h      ;+ move
        mvi     c,01h      ;x axis
        call    movel      ;printing sweep other direction
        pop     b    ;pop y-dim from stack
        call    min5     ;decrement y-dim value by 5
        call    test     ;ck if zero, if so get new data
        push    b    ;push y-dim on stack
        mvi     c,02h      ;enable y axis
        call    jog    ;do y-dim jogging
        jmp     boxx

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

     ;load 4 bytes of data into register.

load    mov     e,m      ;loads 4 bytes of data into registers
        inx     h     ;increment pointer
        mov     d,m      ;x coords in de
        inx     h
        mov     c,m
        inx     h
        mov     b,m      ;y coords in bc
        inx     h
        ret

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

     ;this routine moves table to box location.

goto    push    b       ;this routine moves table to box location
        mov     b,m     ;get x direction from memory (note: x-dir and
                         y-dir interchange after assembly)
        mvi     c,01h   ;select x axis
        call    move    ;do move
        pop     d       ;put y index in de reg for move
        inx     h
        mov     b,m     ;get move direction
        mvi     c,02h   ;select y axis
        call    move
        inx     h
        ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

test    mvi     a,00h   ;checks if bc is zero,if yes get more data
        cmp     b   ;compare 00 with the content of bc
        rnz             ;return, bc is not zero
        cmp     c       ;b was zero is c ?
        jz      main    ;bc was zero,print done,get new line of data
        ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
```

```
min5     dcx   b        ;decrements bc reg by 5 for jog
         dcx   b
         dcx   b
         dcx   b
         dcx   b
         ret

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

jog      push  d        ;jog increments after a printing pass
         lxi   d,0005h  ;jog will move 5 steps
         mvi   b,002dh  ;jog is always in negative table direction.
         call  move     ;do the jog
         pop   d        ;put the original number back in de register
         ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

move     call  doit     ;doit provides stepper bd. commands
         call  hand     ; output the g command from doit
         ret

;:::::::::::::::::::::::::::::::::::::::::::::::::

movel    call  doit     ;movel turns head on and off
         out   10h      ;put on buss
         mov   a,c      ;get enable
         out   12h      ;enable axis
         call  dataken  ;watch to see that index board has data
         mvi   a,08h    ;reset enable,turn on head
         out   12h
         call  busy     ;wait untill move is done
         mvi   a,00h
         out   12h      ;turn head off
         ret

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

           ; provide stepper board command format

doit     mvi   a,004dh  ;m register
         call  hand     ;send in
         mvi   a,003ch  ;<, open register M
         call  hand
         call  add3     ;looks to de for index number
         mvi   a,003eh  ;>, close register M
         call  hand
         mvi   a,47h    ;g stepper bd 'go' cmd.
         ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

         ;bdset gets stepper board setup commands
```

```
bdset     mov    c,m       ;store axis enable byte
          call   getnext   ;get next data from memory to cpu , and
                            then to index board.
          call   getnext
          call   getnext
          call   getnext
          call   getnext
          inx    h
          ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::

getnext   inx    h    ;increment pointer
          mov    a,m  ;move data from memory to accumulator
          call   hand;  move data from cpu to index board
          ret

;::::::::::::::·:::::::::::::::::::::::::::::::::::::::::::

;add3 converts hex to ascii,gets index # from de reg.
;outputs 6 digits,we need only 4,top 2 are zero

add3      mov    a,b         ;get minus if tbere
          call   hand        ;output sign,0=+
          mvi    a,30h       ;select 0
          call   hand        ;output 1st 0 digit
          mvi    a,30h        ;select 0
          call   hand        ;output 2nd 0 digit
          mov    a,d    ;select hex value
          ani    00f0h       ;mask out lower nibble,also c=0
          call   rart        ;move upper nibble down
          call   letck       ;output 3rd digit
          mov    a,d    ;select hex value for low nibble
          ani    000fh       ;mask upper
          call   letck       ;output 4th digit
          mov    a,e   ;select another hex value in e register
          ani    00f0h
          call   rart
          call   letck       ;output 5th digit
          mov    a,e
          ani    0fh
          call   letck       ;output 6th digit
          ret

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;letck looks for hex letter,if so converts to proper ascii code
; it converts number from 0 to 9 in hex to ASCII.

letck     adi    30h         ;add ascii prefix
          cpi    3ah         ;ck if hex "letter" number
          cnc    adj4
          call   hand        ;output digit
```

```
        ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;adj4 converts A-F from hex to ASCII

adj4    sui    09h      ;adj lower nibble to ascii code
        adi    10h      ;set upper to ascii code
        ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;rart rotates 4 digit down.

rart    rar
        rar
        rar
        rar
        ret

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;hand puts data from a reg. on buss
;passes axis enable in c reg.

hand    out    10h       ;put data on buss of output port A
        mov    a,c   ; move axis enable byte to reg. a
        out    12h      ;enable axis
        call   dataken   ;check if data has been taken ?
        mvi    a,00h
        out    12h      ;reset axis enable
        call   busy   ;check if index board busy ?
        ret

;:::::::::::::::::::::::::::::::::::::::::::::::::::::::::

  ;check if data has been taken

dataken in     12h       ;moniter data taken pin 37 stepper bd
        ani    00f0h     ;mask lower bits
        cpi    40h       ;sets zero flag if line is high
        rz
        jmp    dataken

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

  ;check if index board busy

busy    in     12h       ;moniter busy pin 26 ret when high,ie done.
        ani    00f0h   ;mask low bits
        cpi    80h   ; compare
        jz     wait      ;if busy line high,go wait untill low
        jmp    busy
```

```
;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;wait until index board is not busy

wait      in     12h
          ani    00f0h
          cpi    80h
          rnz                   ;if busy is low, return
          jmp    wait

;::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

;joysk is for interrupt of manually joystick control.

joysk     mvi    a,8ah  ;select control word for for port control register
          out    13h        ;set pio in case of reset
          mvi    a,08h
          out    12h     ;clear enables, leave head on
joy       in     11h
          mov    b,a
          mvi    a,04h      ;move right
          cmp    b
          jnz    two
          mvi    c,01h
          mvi    a,004ah    ;J for jog+
          call   hand
          call   time
two       mvi    a,08h      ;move forward
          cmp    b
          jnz    three
          mvi    c,02h
          mvi    a,004ah
          call   hand
          call   time
three     mvi    a,01h     ;move left
          cmp    b
          jnz    four
          mvi    c,01h
          mvi    a,49h
          call   hand
          call   time
four      mvi    a,02h
          cmp    b
          jnz    joy
          mvi    c,02h
          mvi    a,49h        ;I for jog-
          call   hand
          call   time
          jmp    joy
time      mvi    c,0030h
ti        mvi     d,0022h
me        dcr     d
          jnz     me
```

```
dcr     c
jnz     ti
ret
org      1024h
jmp      la7fh
end
```