To the Graduate Council:

I am submitting herewith a thesis written by Richard Paul Beam entitled "A Microcontrolled Input-Output System for a Personal Multimedia Machine." I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

Robert W. Rochelle

Dr. Robert W. Rochelle, Major Professor

We have read this thesis
and recommend its acceptance:

Robert E. Lenkerheimer

Donald W. Bouldin

Accepted for the Council:

Carolyn Minkel

Associate Vice Chancellor
and Dean of The Graduate School

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a Master's degree at the University of Tennessee, Knoxville, I agree that the Library shall make it available to borrowers under the rules of the Library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgement of the source is made.

Permission for extensive quotation from or reproduction of this thesis may be granted by my major professor, or in his absence, by the Head of Interlibrary Services when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Signature R Paul Beam

Date 12/03/93

A Microcontrolled Input-Output Device for a Personal Multimedia Machine

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Richard Paul Beam

December 1993

ACKNOWLEDGEMENTS

The author would like to thank the members of his committee, Dr. Rochelle, Dr. Bodenheimer, and Dr. Bouldin. Special thanks is due to Dr. Rochelle for introducing the author to Empruve where this project was undertaken, and for his guidance throughout the process.

Thanks is also due to Danny McCall of Empruve, Inc. for making this project necessary and especially for finding funding for it.

The author would like to also express sincere gratitude to Dr. Roberto Lenarduzzi for his guidance at Empruve, for the time he has spent proofing the manuscript, and for his constant encouragement without which this thesis would have never succeeded.

ABSTRACT

This thesis describes both the design and the design process for a unique input and output device intended to be part of a consumer multimedia machine. Because of the peculiar demands of the multimedia machine, "off the shelf" solutions did not exist.

The configuration of this design is a quality analog joystick for input and a series of LED displays for output. The input and output systems are controlled by a single microcontroller which interfaces to the host machine via an RS-232 asynchronous communications channel.

Surface-mount components were used in its construction to minimize size and weight, which were critical factors for this configuration.

Several units were built and tested successfully. This system was intended for both consumer and industrial markets and performed well in the laboratory. In many ways, this design is superior to common devices currently available.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	INTRODUCTION	1
II.	SPECIFICATIONS	6
	Input Device	6
	Icon Display	7
III.	SURVEY OF APPROPRIATE TECHNOLOGY	9
	Input Device	9
	Icon Display	12
IV.	PROTOTYPES	14
	Input Device	14
	Icon Display	18
V.	FINAL HARDWARE DESIGN	22
	Input Device	24
	Icon Display	27
	Controller	29
VI.	SYSTEM SOFTWARE	31
	Initialization	34
	Main Routine	35
	Serial Communications Interface Interrupt Service Routine	37
	Timer Interrupt Service Routine	38
VII.	CONCLUSION	40

TABLE OF CONTENTS

BIBLIOGRAPHY 42

APPENDIXES 44

APPENDIX A. HARDWARE SCHEMATICS 45

APPENDIX B. SOFTWARE FLOW DIAGRAMS 50

APPENDIX C. SOFTWARE SOURCE LISTING 57

VITA 67

LIST OF FIGURES

FIGURE PAGE

1. Input device and icon display as part of the multimedia machine concept. 3

2. First prototype using two 10-bit A/D converters on a PC card. 15

3. Second input device prototype using Z-80 CPU. . . 17

4. The final input device prototype using a 68HC05 microcontroller, dual channel A/D converter, and conductive plastic potentiometers. 19

5. Icon display prototype. 20

6. Final combined design with single controller. . . 23

7. Commands recognized by the i/o control system . . 33

CHAPTER I

INTRODUCTION

Recent developments in technology have put computers into nearly every aspect of modern life. A great deal of research resources has been dedicated to the task of making these machines much easier to use since they are often operated by individuals with little or no computer experience.

The latest development in man-machine interaction is the Graphical User Interface (GUI). Microsoft Windows and the Apple Macintosh machines are contemporary examples of mature GUI's. The GUI has become popular because it creates a graphical environment in which objects, also known as icons, are manipulated by an input device. Often, a "point and click" operation on an icon is required to perform the desired function.

The most important aspects of the GUI system are the graphical display and the input device. This is the sole means of communication between a human user and the computer. A properly conceived output display coupled to an intuitive input display will make the machine's operation obvious, while a poor display and awkward input device will make the machine's operation confusing.

Common high resolution video monitors and the ever popular mouse are used effectively in most desktop computer

applications, but they fail miserably when applied to a portable unit. Such a situation presented itself when Empruve, Inc. of Knoxville, Tennessee developed the concept for a portable "electronic book." Manufacturers have attempted to fill the void in portable input devices created with the introduction of powerful portable computers, but no good solution existed for the platform Empruve was developing. Similarly, display technology has advanced rapidly, but the special needs of this product required new development.

Empruve's idea took advantage of the advances in flat panel displays, microprocessors, and digital data compression to produce a machine that can deliver text, graphics, audio, and video information. The coordinated presentation of the multimedia information was via a refined GUI that made the complex machine inherently intuitive to operate. A simple input device, instead of a keyboard, gave the user access whether the environment was an office, a car, or a plant floor.

As shown in Figure 1, the visual information from this machine is presented on three screens: one for video, one for text and computer graphics, and one for the controlling icons required in the user environment. All visual information could have been presented on a single screen, but a large color screen capable of displaying video is currently prohibitively expensive. A typical personal computer displays

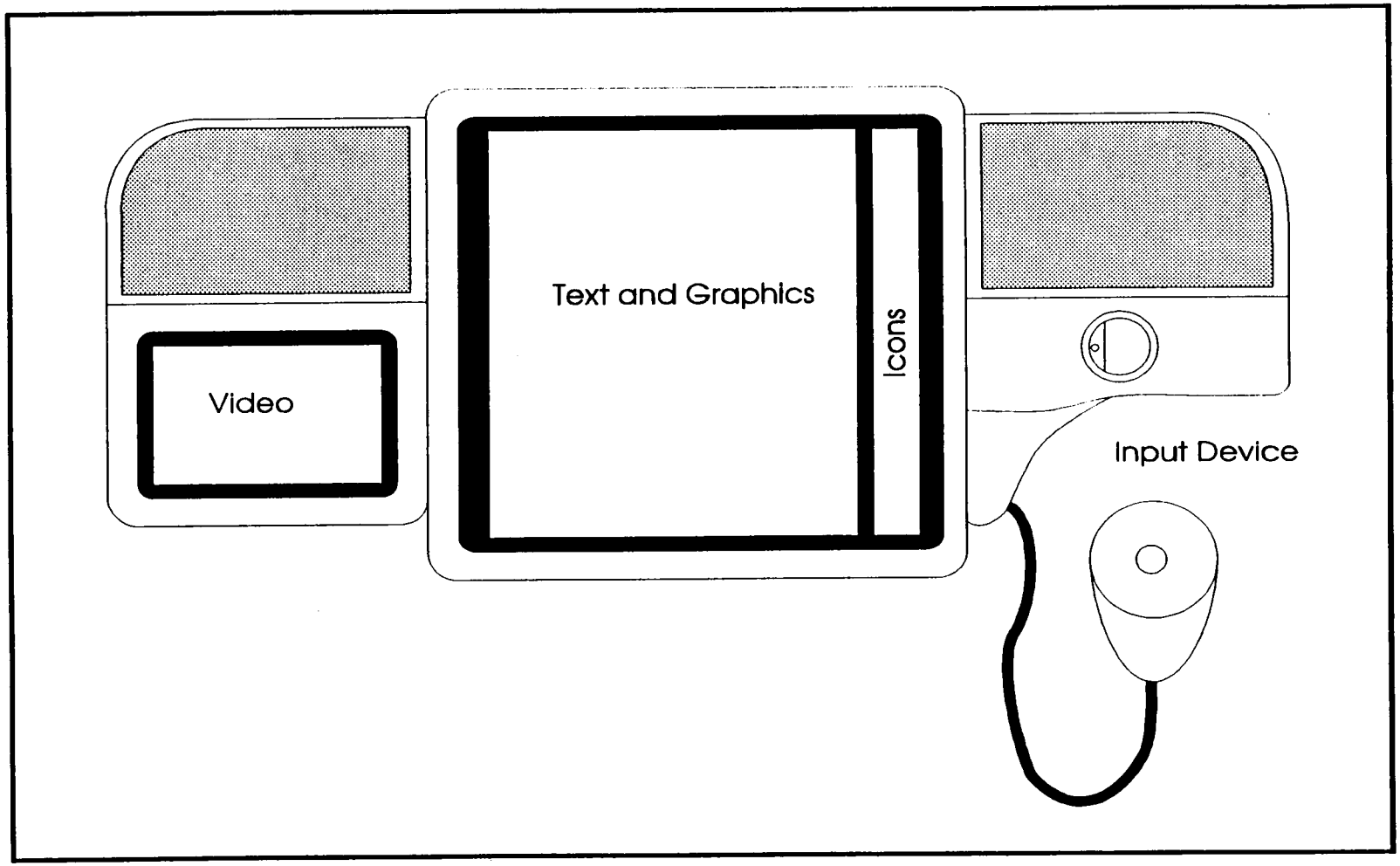


Figure 1. Input device and icon display as part of the multimedia machine concept.

information in "windows" on a single screen. When these windows overlap, information is hidden. Segregating the visual information eliminates this overlap making better use of expensive flat panel real estate. The most appropriate and inexpensive technology will be used for each display.

The largest screen displays text and computer animation. A variety of VGA-quality flat-panel displays have been developed for portable computers, with the best being some form of super twisted neumatic LCD. These are the best choice for text and graphics since they are the largest and least expensive. The marginal brightness and contrast, and the extremely slow screen response time make them inappropriate for video.

The consumer electronics industry has developed small thin film transistor LCDs intended for ultra compact televisions. These are quite appropriate for video, but their small size (6" max) and high cost make them unsuitable for text.

The graphics screen and the video screen use technology developed for other markets, but the icon display uses technology specifically designed for this multimedia machine. No off-the-shelf screen seemed to satisfy the specific demands of this unique application. The display consists of a row of indicators that, coupled with the GUI application, allow the user to navigate through hundreds of megabytes of information. The authors of the GUI specified the set of icons.

Similarly, the input device is a unique design. A variety of devices from various manufacturers were tested before a new device was developed, and, though some devices could work in the environments for which this machine is intended, none seemed ideal.

Although the design of this multimedia machine is unique and interesting, this thesis deals only with the specialized icon display and input device. Appropriate technology had to be located and potentially suitable devices had to be tested before any new system was designed. A small, money-conscious company is wary of tackling new, expensive R & D if appropriate devices already exist in the marketplace.

CHAPTER II

SPECIFICATIONS

Input Device

The input device may be the most crucial part of the multimedia machine because it defines the "feel" aspect of the system. A poor input device will make the unit awkward and difficult to use and will not give potential buyers a good impression. The input device must work well with the GUI to make the system intuitive to use and therefore accessible to the public. The environmental considerations dictate that the input device must be usable either mounted on the machine, in the hand, or on a desk top. This is not a device solely intended for the office environment, so immunity to dirt is also necessary. Its use in either a home or industrial environment means that it must be of rugged and reliable design, but it must also be inexpensive.

The three-screen design of the multimedia machine dictates that the input device must have sufficient range to select objects on all three screens, while it must have sufficient resolution to select an individual pixel. With the arrangement of the screens, the device must have a range of at least 960 pixels.

Ergonomically, the device must be small enough to fit in

an average sized hand, making one hand operation possible for most users. The mechanical design must be such that it is physically simple to operate and does not place undo stress upon fingers or tendons.

The input device must also interface to the host computer in a straightforward manner. Minimal additional circuitry is highly desirable. The host system's power supply has only +5 V, +12 V, and -12 V available, so these will need to be converted if another voltage is necessary.

Icon Display

The icon display is also a very important part of the multimedia machine because it is part of the "look" of the product. A poor icon display will make the machine difficult and confusing to operate. A good icon display will attract attention and make the machine intuitive to operate.

The icon display must work fluidly with the GUI which requires some communication from the host computer. It must have several icons to show all the available functions of the GUI application. The authors of the GUI application dictate 15 icons for the display, and it is highly desirable that these icons be easily changed for different corporate customers. On command from the GUI, icons will be shown as available or unavailable, and the position of the cursor on the display must be shown.

Environmentally, the icon display must be viewable in

direct light and be resistant to dirt and vibration. Industrial and consumer uses mean the display must be rugged, reliable, and inexpensive.

CHAPTER III

SURVEY OF APPROPRIATE TECHNOLOGY

Input Device

By far, the most common input device for personal computers is a standard keyboard. The design concept calls for a user friendly multimedia machine based upon a GUI. A keyboard is often an awkward manipulator of a graphical environment, and so it was eliminated from consideration.

Most GUI users on personal computers use the standard mouse to manipulate objects in the graphical environment. The mouse is, by far, the most economical potential solution, but its suitability to this particular function is poor because of its reliance upon a clean, horizontal surface on which to operate. The laptop computer industry also has found the mouse to be an unsuitable input device.

Because of the emergence of more portable personal computers based upon a GUI, the alternative input device industry has become much more important. Manufacturers such as Microsoft, Honeywell, Logitech, and others are developing new devices to fill the need for a portable input device. Some of the more promising designs from these suppliers will need to be evaluated to see if any meet the specific requirements for this particular device.

Many laptop computers are now shipped with a trackball as the input device. The trackball is a very common input device, not because it performs exceedingly well, but because it uses the same opto-mechanical technology used in most mice. The trackball has some problems, however. It performs adequately when selecting objects, but dragging objects is extremely awkward with most designs. The trackball is also susceptible to dirt which makes it unsuitable for many of the environments for which this machine is intended.

Other manufacturers have produced devices such as mouse pens which are essentially redesigned mice. These devices still require a clean environment and are awkward to use.

One of the most promising technologies is touch screens. These can be virtually impervious to dirt. Three basic types exist: LED, resistive/capacitive, and acoustic wave. LED type displays use a matrix of photo emitters and detectors embedded in the bezel of a screen; they are the least expensive, but do not have the resolution required. Resistive/capacitive types require a special coating on the screen; they have the required resolution, but the coating diminishes the weak contrast and clarity of current LCD screens. Acoustic wave technology does not degrade the screen, but is prohibitively expensive and fairly bulky. All types of touch screens cause the screen to be smudged with grease and fingerprints, and require what becomes a tiring arm movement with constant use.

Several manufacturers incorporate touch pads into their

laptop computers. A special stylus or a common pen is used on the pad to indicate the desired position of the cursor. These are small enough, inexpensive enough, and can work in a dirty environment, but the link between the position on the screen and the position on the touchpad is difficult to make. Through trial and error, a user can eventually find the desired location for the cursor, but this is not intuitive and therefore not suitable.

Suncom Technologies manufactures a small joystick called the Icontroller. This joystick is small and can work in all environments. Joysticks are common in home video games, so it should be an intuitive device. Their joystick, however, is extremely tedious to use since it only measures the direction of travel, not the distance. It really is not much better than the arrow keys on a keyboard and is frustrating to use.

The joystick, however, does seem to be a possible solution. Clearly, the Icontroller proves that a digital joystick is unacceptable, but an analog joystick could eliminate its shortcomings. The low quality joysticks offered for home computers are also unsuitable, but a quality device with a high-resolution interface may be acceptable.

This joystick will need to be designed and developed since none exist on the market which are acceptable. Quality mechanical components do exist, which should make the project possible for even a small company.

Icon Display

The portable nature of this multimedia machine limits display options to those technologies that are small and have low power consumption. Low cost CRTs are not an option. Several technology types exist which may produce a suitable icon display, but these need to be evaluated.

The only off-the-shelf solution would be to use a dot-matrix display. The packaging requirements in the design of the product dictate that if such a screen were to be used, it must be a custom unit which would require a long lead time and significant expense. It is obvious that some design is necessary and merited since the display needs of this device can be much less complicated.

This display needs to only indicate available options and cursor position. The icons will be pre-determined for the major application of the unit and do not need to be software customizable. It is possible for the display to be either a light emissive type or one that requires a backlight. Both types will need to be examined.

Of the light emitting displays, two technology types were evaluated: Electro Luminescent (EL) and Vacuum Florescent (VF). EL displays are smaller and have lower cost, but they require a special power supply, and their brightness degrades over time. VF displays have excellent brightness and contrast, and a rather wide range of colors, but they also require a special power supply. Both types require that

custom units be produced by the manufacturer for the displays to show the custom icons that the GUI application requires.

The backlight technology has a few more possibilities. In this type of display, the icons will be produced on a transparent film which will be illuminated by a light source. The film is very straight forward, but several light sources are possible.

Incandescent lamps are probably the most inexpensive, but they are prone to failure and require significant power. Some means of replacing blown lamps would need to be provided which does not seem possible from the packaging concept. Both EL and VF are possible light sources, but the special power supply requirements make them a poor choice.

Hewlett-Packard and other companies have made significant improvements in LEDs. LEDs are simple to drive, have a very long life if driven properly, and are inexpensive. The more recent light bars manufactured by HP have sufficient brightness for this application, and their availability in a bi-color version means that one color can be used to show that an icon is available and the second to indicate the position of the cursor.

LED light bars, either from Hewlett-Packard or some other manufacturer, seem to be the best choice for an icon display which will be incorporated into this multimedia machine

CHAPTER IV

PROTOTYPES

To verify the initial design assumptions, several prototypes were constructed and tested. Prototyping is sometimes not a smooth process and is prone to failures, but is invaluable when doing a design that will be mass produced and must work. The joystick's evolution had several phases. The icon display, which is simpler, had only one prototype that differed significantly from the final design.

Input Device

A joystick must be composed of at least two sensors that measure the X and Y travel of the stick. Good, inexpensive optical encoders are available which could be used, but to get the resolution required to move the cursor across the three screens, some type of complex gearing mechanism would need to be employed. Potentiometers are used most often in joysticks, but they suffer from temperature drift and non-linearity. Simplicity dictates that potentiometers be used. Hopefully, a good design with quality components will overcome the potentiometer's apparent shortcomings.

The first prototype is shown in Figure 2. It used some National Semiconductor 10-bit analog to digital converters that were interfaced to a standard MS-DOS personal computer.

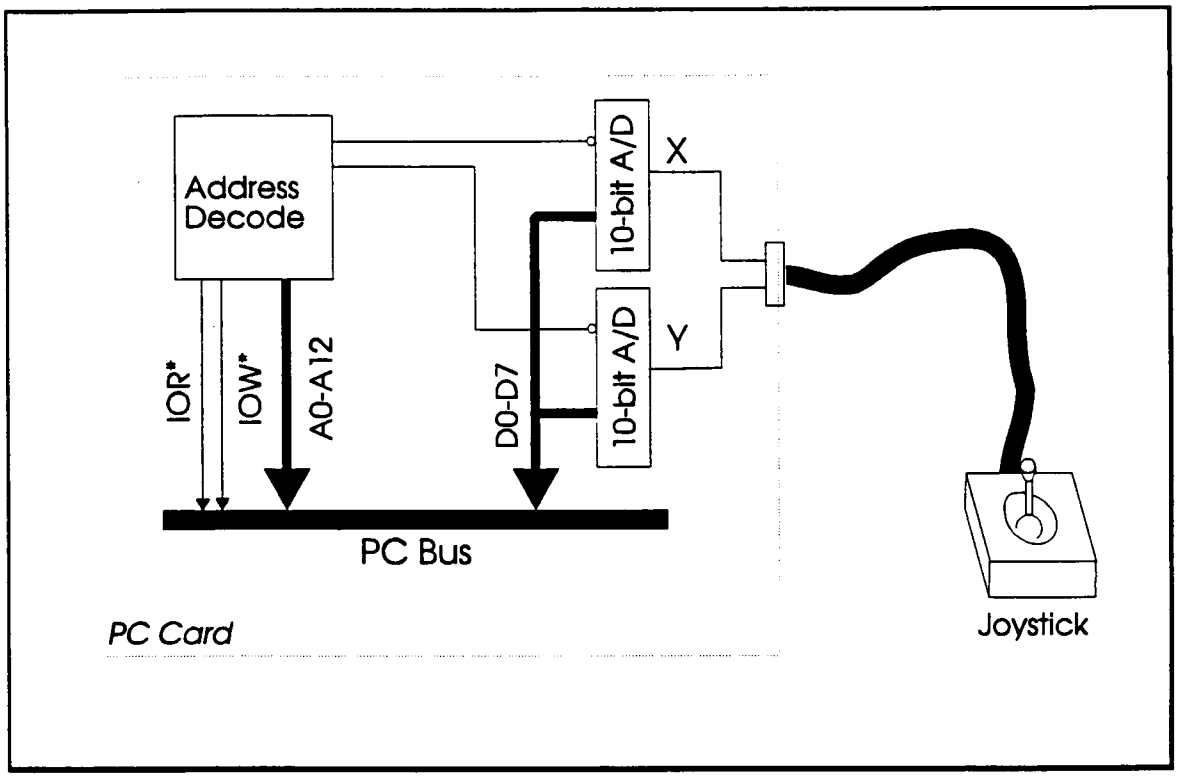


Figure 2. First prototype using two 10-bit A/D converters on a PC Card.

A joystick, originally designed for a Magnavox video game, was attached to the A-D interface via a three-foot four-conductor umbilical cord. The A-D interface was wire-wrapped and placed on a board inside the computer; the joystick housed only the potentiometers.

A short BASIC program was written to interrogate the joystick and place an imitation of a cursor on the screen. Immediately, the design's shortcomings were obvious. The potentiometers in the joystick were poor and a small movement of the joystick sometimes meant a large movement of the cursor. The cursor also would not sit still. Obviously, the analog circuitry had so much noise that it caused the cursor

to shake violently. Some software filtering was possible in BASIC, but not enough to make the system usable. Certainly, great care needs to be exercised in the design of the analog circuitry to overcome this problem.

The first prototype was unusable for any "real world" application, but it proved a good proof of concept. Ideally, the joystick should emulate a popular mouse so that extensive software drivers need not be developed while the hardware is still being tested. The second prototype (Figure 3) addresses this issue by placing a small computer between the joystick and the host computer. This small computer would read, filter, and send the data to the host in a format identical to the Logitech C7 mouse which is well documented.

The processor chosen was the Zilog Z80 because of its low cost and familiarity. The Z80 is not ideally suited to this task because it requires a large amount of support circuitry, but the availability of development software and a history of its reliability outweighed the problems introduced by additional components. The Z80 computer would incorporate the same A-D converter as the first design, but it would process the data from the joystick, format it, and send it to the personal computer via the RS-232 interface.

The components for the Z80 system were wire-wrapped and placed on a board inside the personal computer. Sufficient Z80 assembler software was written to test the system and "fool" applications into believing that a Logitech mouse was

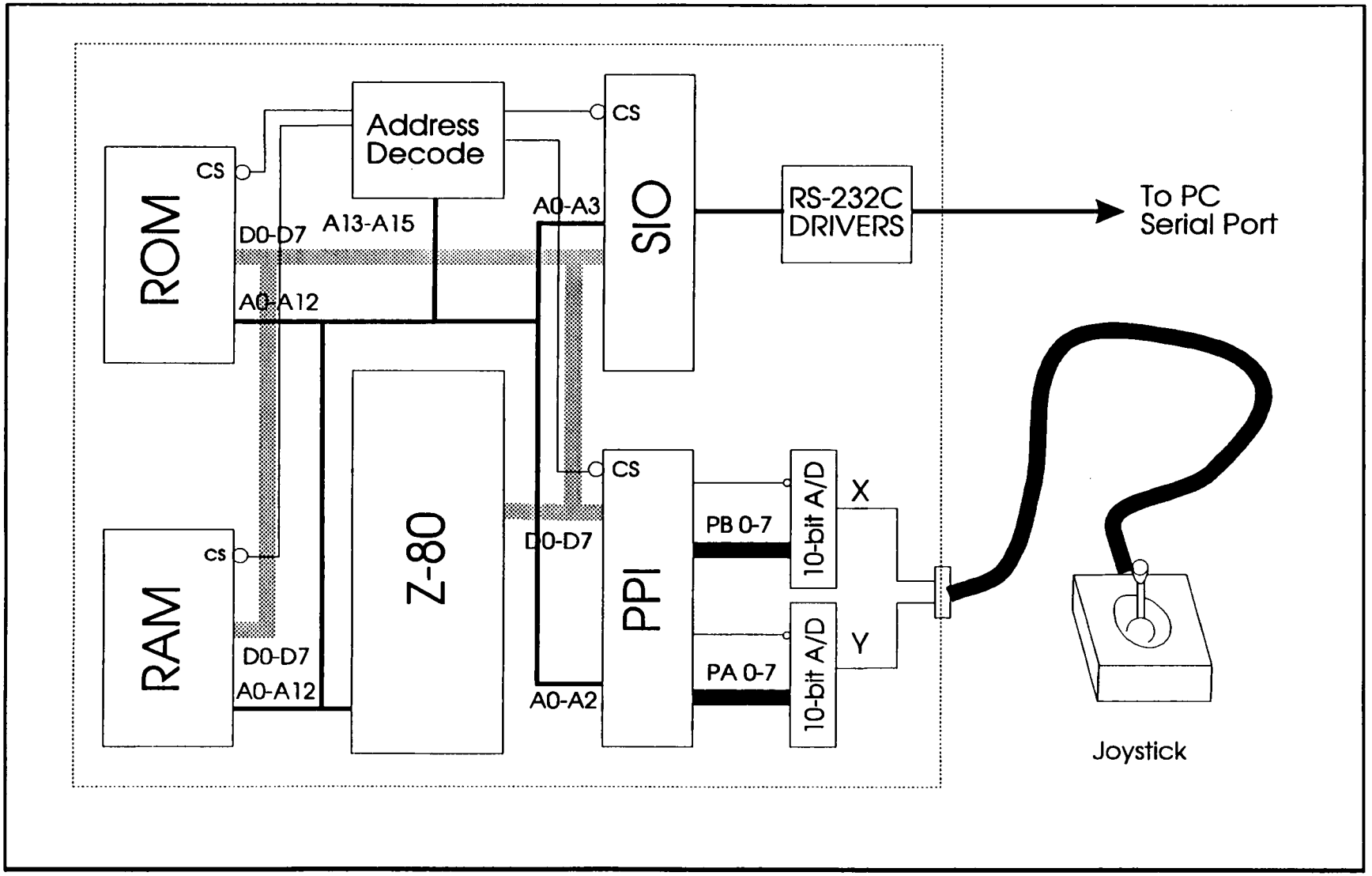


Figure 3. Second input device prototype using a Z-80 CPU.

connected to the serial port. The system worked well, but some problems still prevailed. Still, noise was a problem and the potentiometers were not up to the task. Other than that, the device proved to be controllable and intuitive even in complex Windows applications.

Research into potentiometers revealed two quality types: wire wound and conductive plastic. The conductive plastic pots are generally smaller and less expensive and have a shorter lifetime that the wirewound variety. The next prototype used conductive plastic pots in an attempt to minimize noise.

Research into microcontrollers yielded the Motorola 68HC05 family of single chip microcontrollers. These devices have a serial port, parallel ports, 8K ROM, and 256 bytes of RAM, and should be capable of performing all the functions of the Z80 computer with far fewer components. Certainly, this should be used in the next prototype.

The last prototype is shown in Figure 4 and did incorporate conductive plastic potentiometers and a 68HC05 derivative microcontroller. The combination worked extremely well and would have been the final design had the idea of the icon display not been developed.

Icon Display

The GUI application's developers decided that 15 icons

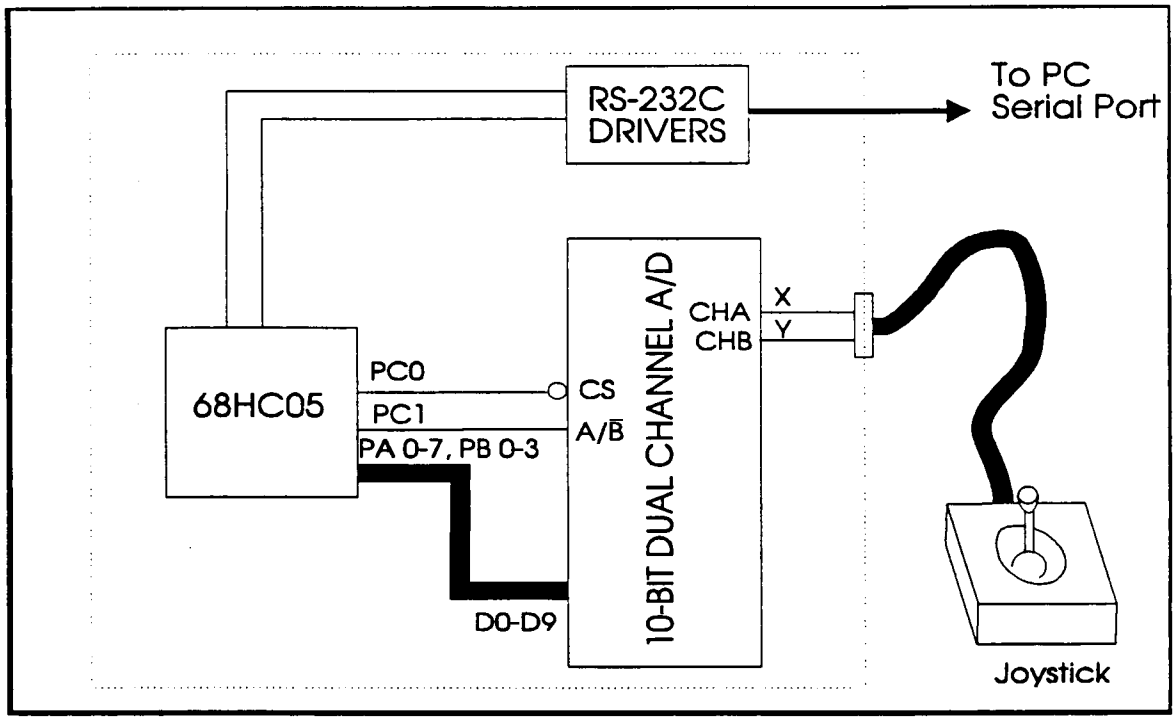


Figure 4. The final input device prototype using a 68HC05 microcontroller, dual channel A/D converter, and conductive plastic potentiometers.

were needed by the system. These items would be blacked out when not available, yellow when available, and red when selected by the cursor. The mechanical designers determined that the display's length must be the same as the text screen of the final multimedia machine.

The first prototype is shown in Figure 5 and took advantage of some things learned from the input device. It seemed appropriate to communicate with the icon display via the serial communications port. This would allow for test and development on a standard personal computer. The 68HC05 microcontroller also seemed like the logical choice to drive the display.

The technology survey already revealed that the bi-color

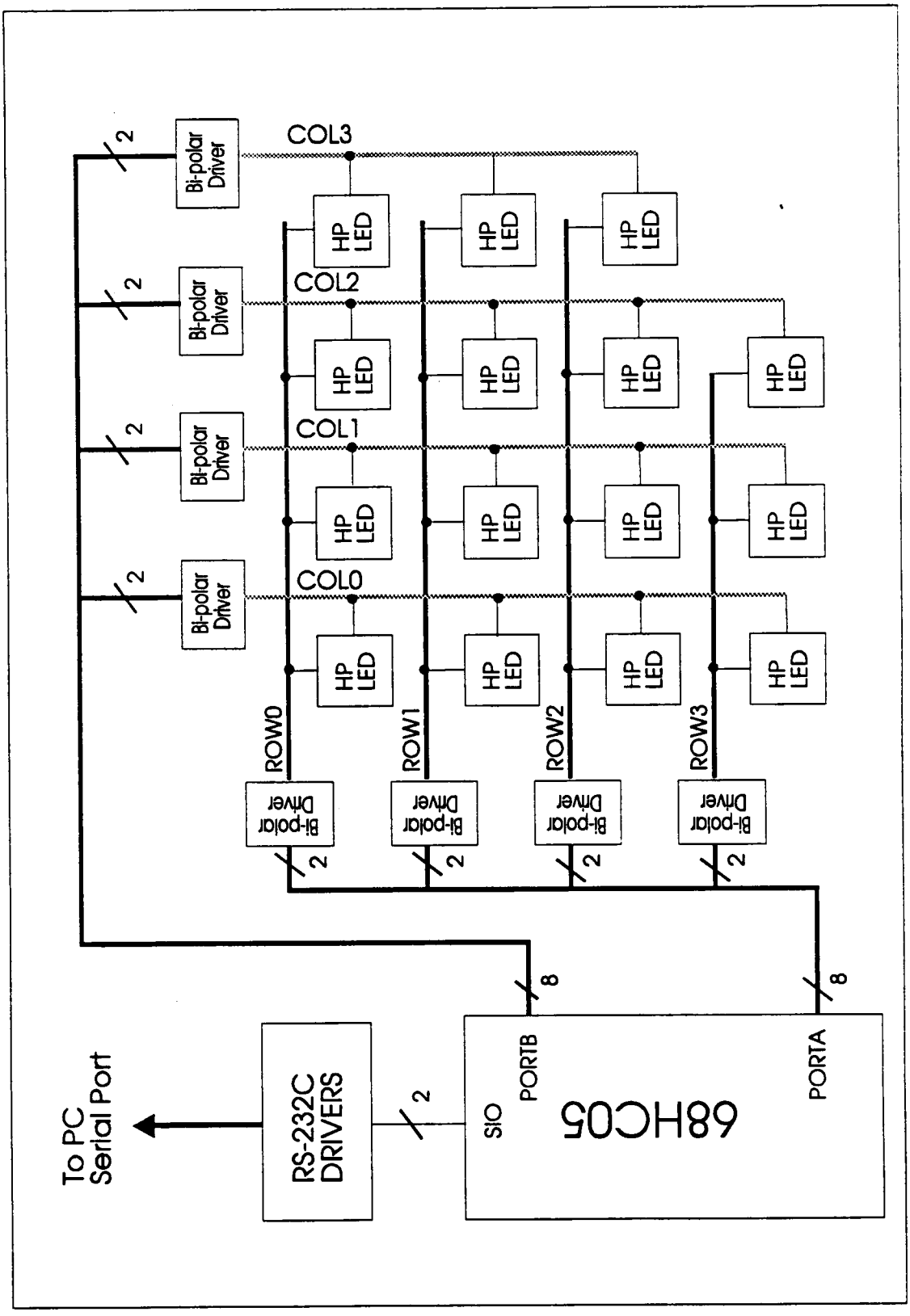


Figure 5. Icon display prototype.

LED light bars were the components to use in the display. It seemed obvious that these displays could be multiplexed and that would reduce the amount of circuitry needed in the display.

Discrete bipolar transistors were used to form row and column drivers for a 4 x 4 matrix of light bars. The 68HC05 received rudimentary commands from the host computer and updated the display accordingly. Yellow was used to indicate that an icon was available, and red was used to show the position of the cursor. Unavailable icons were off.

The display was constructed using point-to-point wiring and performed well. A problem encountered was that the multiplexed display generated power supply noise that interfered with the video display in the first model. Increasing the refresh rate and filtering the power supply alleviated the problem. The only other problem was that the icon display required a second serial port, something the main board in the multimedia machine did not provide.

CHAPTER V

FINAL HARDWARE DESIGN

Earlier prototype attempts solved most of the problems in this input and output device concept, so the final design is primarily a refinement of earlier work.

The major difference in the final design is that the circuitry for the input device and icon display are combined into one system under the control of one 68HC05 microcontroller. The configuration of the final design in shown in Figure 6, while the schematics for the system are contained in Appendix A. This configuration reduces the number of components to accomplish both functions, and requires only one serial port for communication with the host. The software, however, will be more complex.

Additionally, the ergonomic criteria that the input device must fit in the hand, the packaging specifications of the final product, and the expected number of components based upon earlier prototypes, dictate that surface-mount technology needs to be incorporated wherever possible. This limits component selection and increases cost, but it also places more severe restrictions upon power consumption since, in general, surface mount components cannot dissipate as much heat as their through-hole counterparts. Therefore, power efficiency must also be a major part of the component

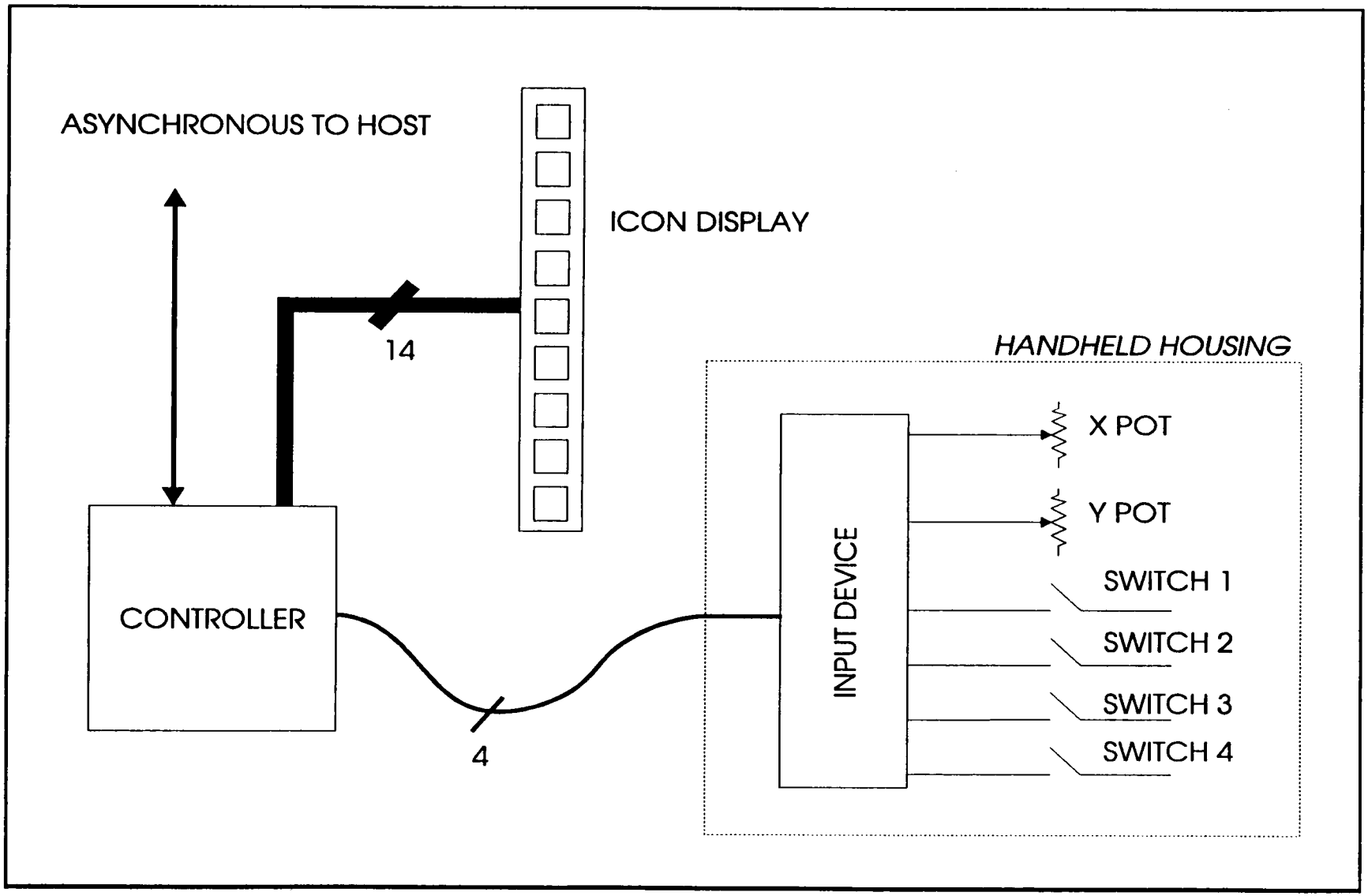


Figure 6. Final combined design with a single controller.

selection process.

Finally, the major factors in the component selection process are availability, cost, and size. This system is intended to be mass produced for the consumer marketplace, so component shortages and high prices cannot be tolerated. Availability is a bigger factor than one may think as getting information on a component and actually purchasing a component are two entirely different processes. Some parts exist only on the pages of manufacturers' databooks.

Input Device

In previous prototypes, a 10-bit resolution proved adequate, and so its use will continue in the final design. Likewise, the 'outboard' input device proved reliable in earlier tests provided that careful attention is paid to the analog circuitry and that analog-to-digital conversion is performed as close to the analog inputs as possible. This dictates that at minimum, an analog to digital converter is required within the joystick itself.

Some form of serial communication between the input device and the console is needed to reduce the number of conductors in the umbilical cord. This could be accomplished either through the use of a microprocessor or an analog-to-digital converter with serial input and output.

Because space is at a premium within the input device, a

serial A-D converter would be ideal. A survey of manufacturers' products yielded the LTC1090 series produced by Linear Technology. The LTC1093 is a six-channel, 10-bit serial analog-to-digital converter which is readily available in a sixteen-pin surface-mount package. The LTC 1093 requires only a three-pin serial data link and a stable reference to produce accurate unipolar or bipolar conversions. This component is the heart of the input device, and all other circuitry is designed to cater to its needs.

To make the design more straightforward, unipolar conversion over a 0 - +5 V range is used. This means that the voltage reference could also be the power source for all other components and the serial interface to the controller will not require additional voltage conversion. The choice for the voltage reference is the LP2951 produced by National Semiconductor which is also readily available in a very small 8-pin surface-mount package.

This regulator requires a voltage source of at least +7 V in order to give a stable reference. Therefore, the power supplied to the input device must be at least +7 volts with +12 V being the assumed logical choice.

Of course, the main purpose of the LTC1093 is to convert the X and Y outputs from the joystick's potentiometers into digital form which represent the desired cursor position. Unfortunately, the joystick mechanism has a rather large offset and a small range which needs signal conditioning in

order to give accurate and precise position.

Each potentiometer is centered at +2.5 V, but has only a 1.25V span. Eliminating the offset, and amplifying the range will give a full 10-bit conversion over the joystick's travel. This function is performed by U1, an LM660 quad operational amplifier. The use of two operational amplifiers on each channel provides a high impedance inverting differential amplifier with a gain of 4 and an adjustable 1.25 V offset adjustment. Software within the controller can correctly interpret and convert the inverted data.

The final need of the input device is to provide four push buttons. Since the LTC1093 has four spare analog channels, these can be used to monitor the four push buttons and use the same serial data link. A very high conversion number or a very low conversion number indicate a switch open or closed, respectively.

After some initial tests with various lengths of the umbilical cord, it was discovered that the data driver from the LTC1093 was too weak. Adding Q1, a 2N7002 N-channel MOSFET, solved this problem, but it inverts the data coming from the converter. Software will have to take this inversion (the second one for the X and Y position) into account when it interprets the data from the input device.

The final consideration for the input device is power consumption. The LP2951 can source 100 mA and dissipate no more than 0.4 W. The worst case requirement for the input

device is only 11 mA which is well within the LP2951's capabilities. With a 12V supply, the LP2951 would only dissipate 0.07W which is also safely within design specifications.

Icon Display

The Hewlett Packard light bars performed well in earlier tests, so there is no compelling reason to change. The driving circuitry could perhaps be optimized to reduce the number of components and the size of the circuitry involved. If surface mount components are to be used to optimize space, careful attention must be placed upon power consumption and heat dissipation since LEDs are relatively high current devices.

The light bars are bi-color devices which need bipolar drivers in order to display each color. The 15 LEDs can be configured in a 4 x 4 matrix, with the rows being bi-polar and the columns being tri-state. Tri-state drivers are needed to turn individual light bars off.

Each light bar contains four discrete LED segments. Since the display will be multiplexed, HP design criteria states a maximum pulsed current of 90mA with a corresponding forward voltage of 3.5 V for the red and 3.4V for the green. This pulsed current must have a maximum duration of 2 milliseconds and must follow the criteria from the derating

curve provided in the Hewlett Packard data sheet.

If the four LEDs are wired in parallel, each light bar would require 360mA and a voltage of 3.5V. This is a rather high current requirement, and the 1.5 V difference from the +5V supply may not be enough. Conversely, if the four LEDs were wired in series, the light bars would need only 90 mA at 14 V, but this exceeds the +12 V supply. A compromise using two series pairs in parallel might be the best since each light bar would require 180 mA and 7 V, which could easily be derived from the +12 V supply. This does mean that the display will require 2.16 W -- 0.9 W of which is dissipated in the row and column drivers.

An integrated solution would be the cleanest for the display drivers, but the relatively high current requirement eliminates most available devices. A technology survey revealed some small, surface mount discrete MOSFETs which are readily available, easy to use, and able to handle the power demands with very low losses. The row drivers use both the N-channel and P-channel MOSFETS in a totem pole configuration which can be controlled by an open collector signal on the gates. Series resistors limit the current to the light bars.

The column drivers use the same MOSFETS, but the gates are controlled separately in order to provide the high impedance state required. The gates to the N-channel can be controlled with +5 V CMOS logic, while the P-channel gates require and open collector signal.

Controller

The controller provides the intelligence for both the icon display and the input device. The control section must communicate with the host computer while at the same time updating the multiplexed icon display and polling the input device.

The 68HC05 microcontroller should be appropriate for the final design because it has proven to be reliable and available. It does come in a surface mount package, so space should not be a problem.

The interface to the host computer is through the 68HC05's Serial Communications Interface (SCI). The SCI is an asynchronous serial interface which can be tied to the serial interface of the host computer. It is assumed that the host computers serial interface will be TTL compatible instead of the RS-232C level signals. This saves the additional drivers that would otherwise be required in both the host computer and on this device.

The input device will be accessed via the 68HC05's Serial Peripheral Interface (SPI). The SPI is a three wire synchronous serial interface which is often used with serial clocked latches. In this case it can be programmed to be compatible with the serial interface of the LTC1093.

The interface to the icon display is via 12 of the 68HC05's general purpose I/O lines. The eight lines of port A are used to drive eight very small n-channel MOSFETS which

provide the open collector signals needed for the icon displays drivers. Four lines from port B are used to drive the n-channel gates for the tri-state column drivers. An addition line from port C is used as a master "on/off" for the icon display. This could be used to cut power to the display while the drivers are changing state or could be used to dim the display using a pulse code modulated signal.

A series resistor on the icon interface provides the additional current limiting needed by the LED light bars. An additional filtering capacitor is included to help isolate the noise generated by high current switching.

Similarly, the input device needs to have a current limiting resistor to assure that a failure in the external device will not bring the whole system to a halt. This is particularly important in a consumer device which is prone to abuse and damage.

CHAPTER VI

SYSTEM SOFTWARE

The software for the microcontroller is probably the most important part of the system. Not only must the software control the icon display and input device, but it must also communicate with the host computer. This communication is vital to making the system functional and to provide seamless integration with the host computer's operating system.

It is necessary that the host computer be able to send commands to the microcontroller. These commands may be used to either set parameters for the input device or update information on the icon display. These commands form a language which must also be adhered to by a device driver resident in the host system.

Commands to the input device and the format of data from the input device should adhere to some standard in order to maintain compatibility with some currently available software. Data formats for mice and digitizers are commonly known and well documented, so they can be used to standardize the input device. This input device will support the popular mouse formats used by Mouse Systems, Microsoft, and Logitech, but the primary format intended for this application is "Absolute Bit Pad One Packed Binary" developed by Summa Graphics for its digitizers.

Mouse formats do not work as well with a joystick because they report only relative movement, not absolute position. With a mouse, this is dealt with by the user picking up the mouse, moving it, and then putting it back down in an area where there is more room. The movement of the joystick is confined, therefore an absolute reporting mode is much preferred.

The commands to the system are a superset of those available with the Logitech C7 mouse. In addition to the mouse mode and format commands, a set of commands to update the cursor and the icon display are provided. These commands are summarized in Figure 7.

The design of the system makes the hardware simple but software more complex. Multiplexing the icon display, polling the input device, and maintaining a communications link with the host computer must all be done in real time. The icon display is especially susceptible to catastrophic hardware failure if the software is not written correctly.

Motorola's design of the MC68HC05 family of microcontrollers prohibits the easy use of high level languages since they did not allow for a user stack. Therefore, assembly language is the tool of choice, especially in a design such as this where timing is critical and resources are limited.

Fortunately, the 68HC05 microcontrollers have a variety of interrupt sources which makes the writing of real time

Command	Description
'A'	Select Absolute Bit Pad One Format.
'D'	Disable reporting, enter prompt mode.
'J'	Set to 5 reports/second.
'K'	Set to 10 reports/second.
'L'	Set to 15 reports/second.
'M'	Set to 25 reports/second.
'N'	Set to 35 reports/second.
'O'	Enable continuous reporting.
'P'	Prompt for report and set to prompt mode.
'Q'	Set to 30 reports/second.
'R'	Set to 20 reports/second.
'S'	Set to Microsoft Mouse Format.
's'	Return one byte status.
't'	Return two byte status.
'U'	Set to Mouse Systems format.
'W'	Enable hex mode.
'*c' <curpos>	Set cursor position to <curpos>.
'*i' <msB> <lsB>	Turn icons on. <msB> corresponds to LEDs 8-14, <lsB> corresponds to LEDs 0-7. (1 = on)
'*n'	Set to 1200 baud.
'*o'	Set to 2400 baud.
'*p'	Set to 4800 baud.
'*q'	Set to 9600 baud.

Figure 7. Commands recognized by the i/o control system.

control programs somewhat easier. Interrupts are available from a real time clock, the SPI, and the SCI. All three primary software functions can be interrupt service routines, if desired, but it seems better to make one of the three functions the main program loop and make the other two interrupt driven.

Because the multiplexing of the icon display is very time sensitive, it will be driven by the real time clock interrupt. Since the polling of the input device and the formatting of its data appear to require the most code and processing time, this will comprise the main program loop. This leaves the host communications link which can be easily driven by the serial communications interface interrupts.

The flow diagrams for the various routines are contained in Appendix B, while the assembly language source code is listed in Appendix C.

Initialization

The 68HC05 hardware must be appropriately configured and the machine put in a known state upon program initialization. All ports are configurable as are the timer and interrupt sources. The icon display requires that Port A and Port B be set to output, and these ports must be cleared to make sure the LED's are in an "off" state. Similarly, the CS for the LTC1095 must be initialized for output, and the line released.

The SPI has several controlling registers which must be

appropriately configured. The device is used in its "master" mode with a clock rate of 125 kHz. Since the LTC1095 requires three bytes per sample, this yields a conversion rate of 5.2 kHz, which is well within the design limits of the A/D and well above the sample rate required. Interrupts from the SPI are not enabled since polling the input device is part of the main program.

Similarly, the SCI has several registers which control its operation. Since many baud rates and several data formats are possible, Absolute Bit Pad One Packed Binary format at 2400 baud was selected as the initial state. Also, transmission of data packets to the host system is not enabled; this is left as an initialization task for the host's device driver.

The 68HC05 has a versatile 16 bit free running timer that is used to generate the real time interrupt used for multiplexing the LED's in the icon display. The "output-compare" interrupt on this timer is enabled; the frequency of the interrupt must be handled by the timer interrupt service routine.

Finally, RAM variables must be initialized and interrupts must be enabled.

Main Routine

Earlier prototypes proved that noise in the analog circuitry caused conversion errors which resulted in a

"nervous" cursor which moved erratically on the screen. Filtering the data helped significantly, so it will be incorporated here. It is assumed that white noise is the culprit, and so, instead of more complex filtering schemes, sample averaging is used. Sixteen samples for the X and Y coordinates are sampled to give the filtered X and Y coordinates for the cursor.

Taking a sample from the LTC1095 is more complex than a parallel A/D converter. Firstly, the CS for the device must be asserted, and then a command must be transmitted via the SPI to the device. The commands select the appropriate analog channel, bipolar or unipolar conversion, and whether the output is MSB first or LSB first. Both X and Y are unipolar conversions on channel 0 and channel 1, respectively. LTC1095 commands are described in the Linear Technology data sheet. For the 68HC05, three bytes must be transmitted in order to send a command and receive the 10 bit conversion.

Reading the push buttons from the input device is similar to reading the X and Y coordinates, except that no sample averaging is performed. The push buttons are connected to channels 2-5 on the LTC1095. Since these are digital inputs, only the MSb is used to detect whether a button is depressed.

The last responsibility of the main routine is to format the X, Y, and pushbutton data. If the data in the current packet is identical to that of the last, no data is transmitted. The formatting of the data is quite tedious.

The three main data formats have different requirements for the number of data bits and parity. The SCI hardware does not have the ability in hardware to calculate parity, so this must be done with a software routine. To make things easier, data is always transmitted to the host with 11 data bits: 1 start, 1 stop, and 9 data. If the format calls for 7 bit data with parity, the extra bit is converted to a stop bit, otherwise the 9th bit is the computed parity.

Once the data is converted into the appropriate format, it is stored in a buffer awaiting transmission. Two bytes of buffer storage is used for each byte in the data packet; the second byte holds the parity bit.

Transmission of the packet is handled by the transmit interrupt service routine which is enabled by the timer interrupt service routine at the appropriate time.

Serial Communications Interface Interrupt Service Routine

A Serial Communications Interface (SCI) interrupt has two sources: transmitter empty and receive buffer full. The SCI ISR must determine which caused the interrupt and service each appropriately.

The main routine formats the X, Y, and button information and stores it in a buffer at memory location 80 H. For each byte transmitted byte, a second byte is used to hold the parity bit since the SCI does not contain parity generation hardware. The transmitter interrupt service routine (ISR)

simply stuffs the receiver buffer with the appropriate data, and updates the buffer pointer. When the last byte is transmitted, transmitter empty interrupts are disabled.

The receive ISR is a bit more complex since several commands require multiple bytes, and some require a response to be transmitted to the host. The multiple byte commands require a simple state machine to keep track of what portions of what command have already been received.

Since a command which requires a response may be received while a data packet is being transmitted to the host, the transmit ISR is disabled and the response is sent immediately. This means that the host may receive one 'garbage' character before the correct response is received. The remainder of the data packet is discarded and the host must wait until another reporting period in order to get the full X, Y, and button information.

Timer Interrupt Service Routine

The output-compare function of the 16-bit free-running counter is used to generate the real time clock interrupt. This interrupt is used both in the multiplexing of the LED icon display and in the SCI transmit ISR to control the reporting interval. The interrupt frequency must be adjusted to work with both functions.

The multiplex frequency of the icon display is not arbitrary since some undesirable affects, such as extreme

flicker or damage to the LEDs, may occur if the frequency is not correct. Setting the interrupt rate to 0.5 milliseconds will allow the icon display to have a refresh rate of 133 Hz, which is well within the safety limit of the LED light bars, and would allow a report rate as low as 8 reports/second. Running the LEDs in this fashion will reduce their brightness by 60%, but the light output, with a good filter, is acceptable.

CHAPTER VII

CONCLUSION

No design is ever perfect, and this one is no exception. In general, it met all the performance criteria. The cursor's movement was fluid, predictable, and highly controllable -- in many ways superior to common input devices. The fact that the device sat in one place was often a blessing as clutter built up on desk tops and lab counters -- common desktop environments in which the mouse fails miserably.

Evaluation of the input device took considerable time and effort. Device drivers for both DOS and Windows environments had to be created to test the device in a real environment. Standard mouse drivers could be used to a small extent, but custom software was required to exploit the unique features of the design.

Some shortcomings in the input device's design became apparent as several units were manufactured. Reliance upon analog circuitry meant that each joystick had to be manually calibrated. This calibration would probably not endure for the life of the product which meant that some user service may be necessary. Additionally, the calibration of each input device was slightly different with each potentiometer having a slightly different offset and range.

The icon display, similarly, performed as expected.

Flicker was not a problem, and, with a proper light filter, brightness was adequate for most environments. However, it became clear that the bi-color light bars did not do an adequate job of showing the cursor. The use of the color red for the cursor also did not seem ergonomically correct since that color usually denotes an alert or alarm.

The shortcomings for the icon display are relatively simple to fix. Using single color LED light bars for the icons and a separate small single color LED to backlight a cursor seems to be the solution. This makes the light bar drivers a simple current source or sink. The second LED for each cursor position can easily be driven with TTL or CMOS logic.

The shortcomings for the input device are more complex. One solution is to provide some form of user calibration routine in software which would "teach" the host device driver how to interpret data from the joystick. A second approach would be to devise self-calibration circuitry in the joystick. Some additional digital sensor would need to be incorporated which could be used to adjust the gain and offset of the analog amplifiers in the joystick.

As of the time of this writing, no one is producing a similar device, and many of those who have experienced this design wonder why it is not common. Perhaps, in the future, something similar will be the most popular input device.

BIBLIOGRAPHY

BIBLIOGRAPHY

Optoelectronics Designer's Catalog, Hewlett-Packard Corporation, November 1990

General Purpose Linear Devices Databook, National Semiconductor, 1989 Edition.

Logitech C7 Technical Reference Manual, Logitech Corporation, 1988.

1990 Linear Databook, Linear Technology Corporation, 1989.

M68HC05 Applications Guide, Motorola, Inc., 1989.

MC68HC05P7 Technical Data, Motorola, Inc., 1990.

Castello, Joseph A., Handbook of Display Technology, Academic Press, Inc., 1992.

Howard, Bill, "Field Mice. Different, Yes. Better, Maybe.," PC Magazine, October 15, 1991, pp 111-133.

Howard, Bill, "Point-and-shoot Devices," PC Magazine, August 1987, pp 95-96.

"Mice for Mainstream Applications," PC Magazine, August 1987, pp 117-144.

"Tablets for Precision Graphics," PC Magazine, August 1987, pp 159-181.

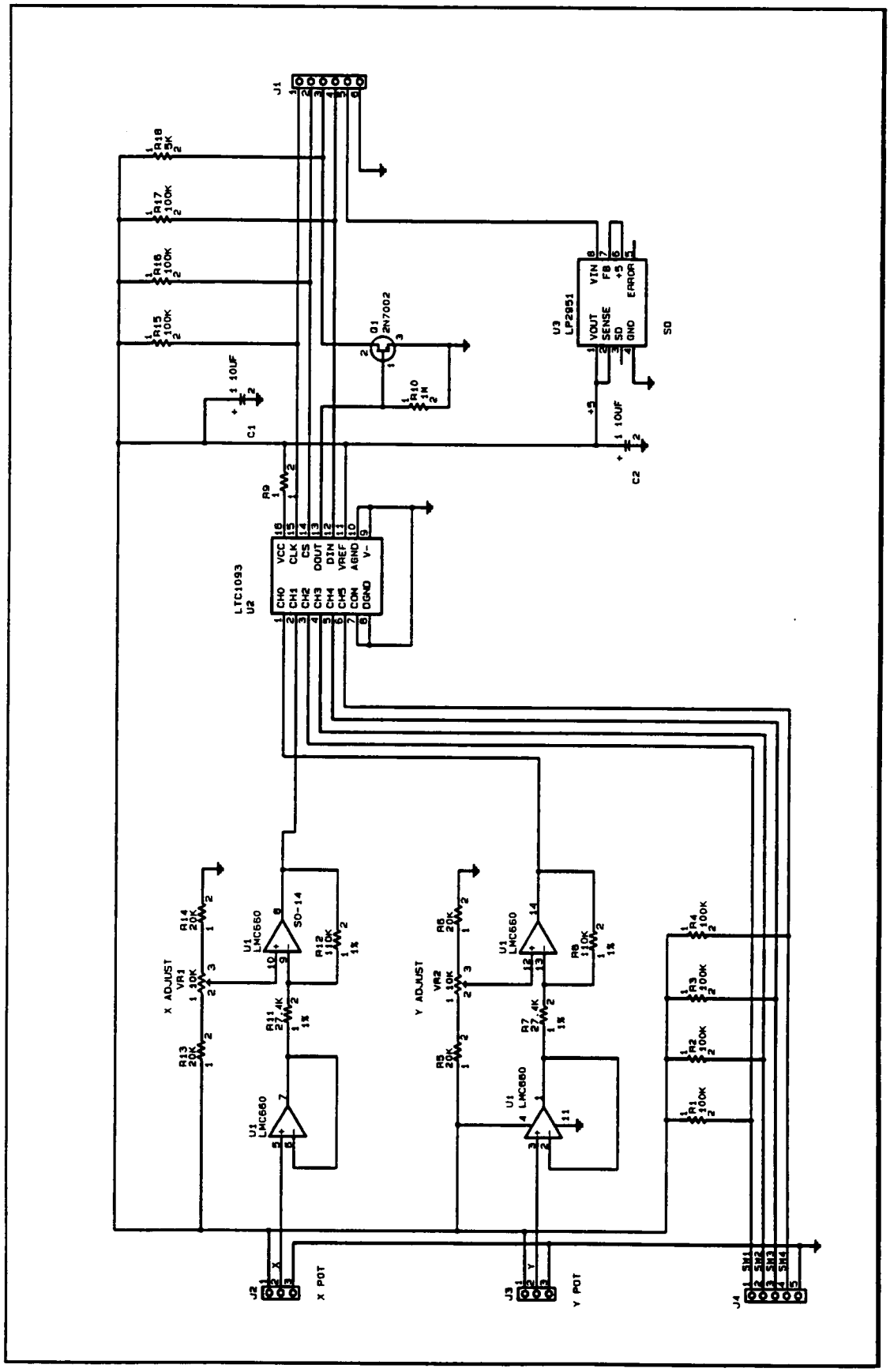
Stone, M. David, "Touch-Screens for Intuitive Input," PC Magazine, August 1987, pp 183-192.

Poor, Alfred, "Trackballs: Stationary Mice," PC Magazine, August 1987, pp 199-202.

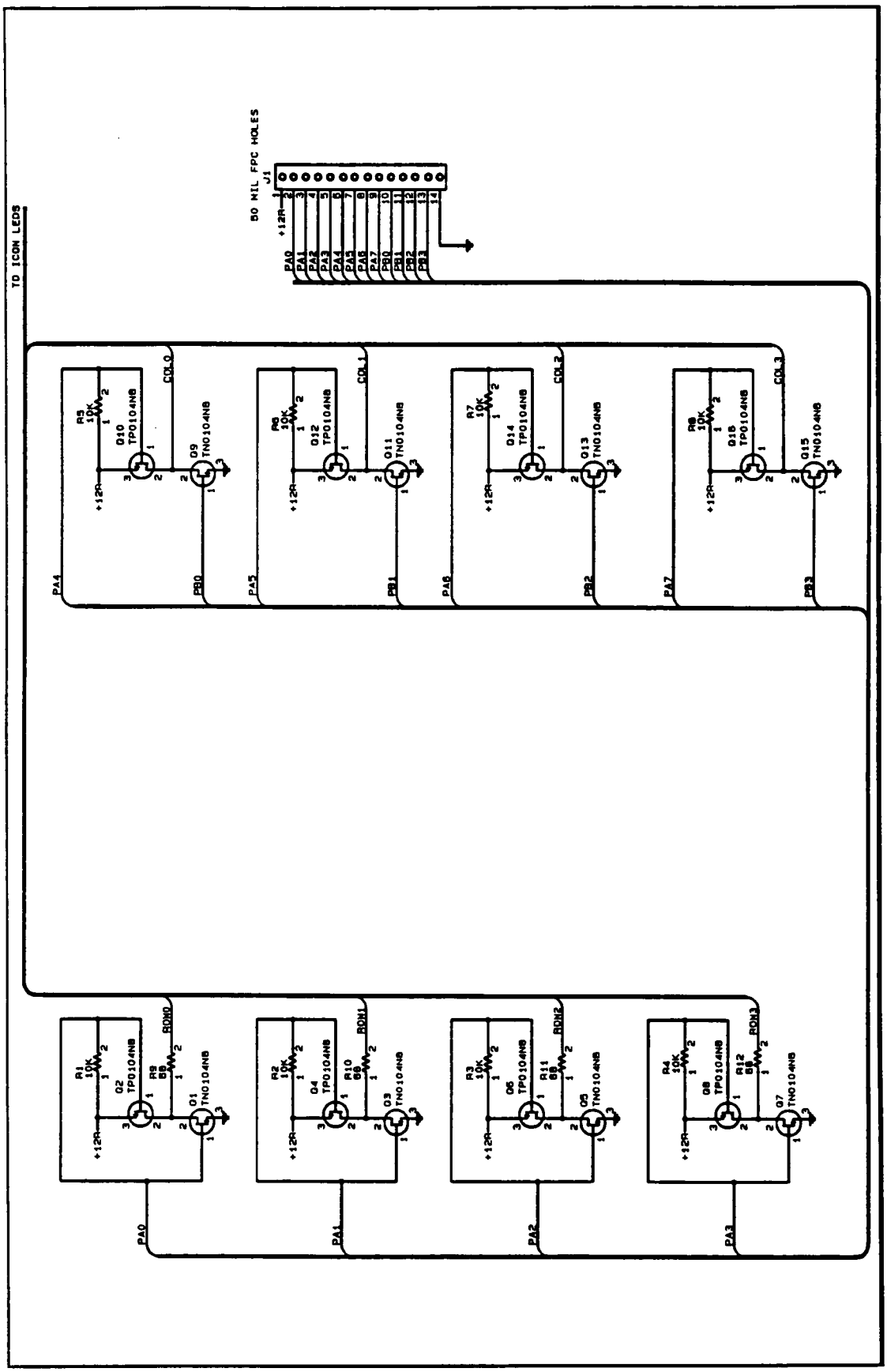
APPENDIXES

APPENDIX A

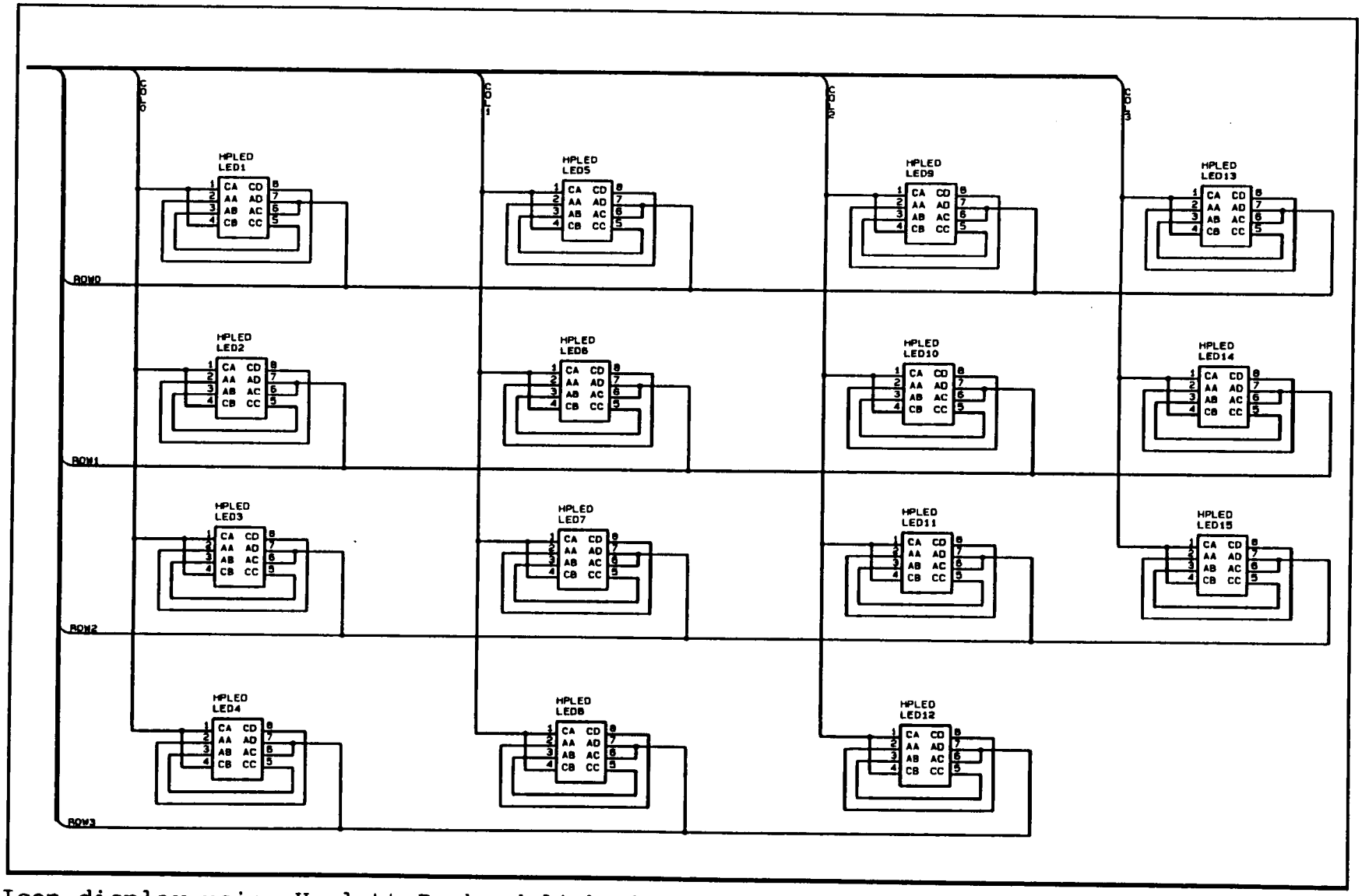
HARDWARE SCHEMATICS



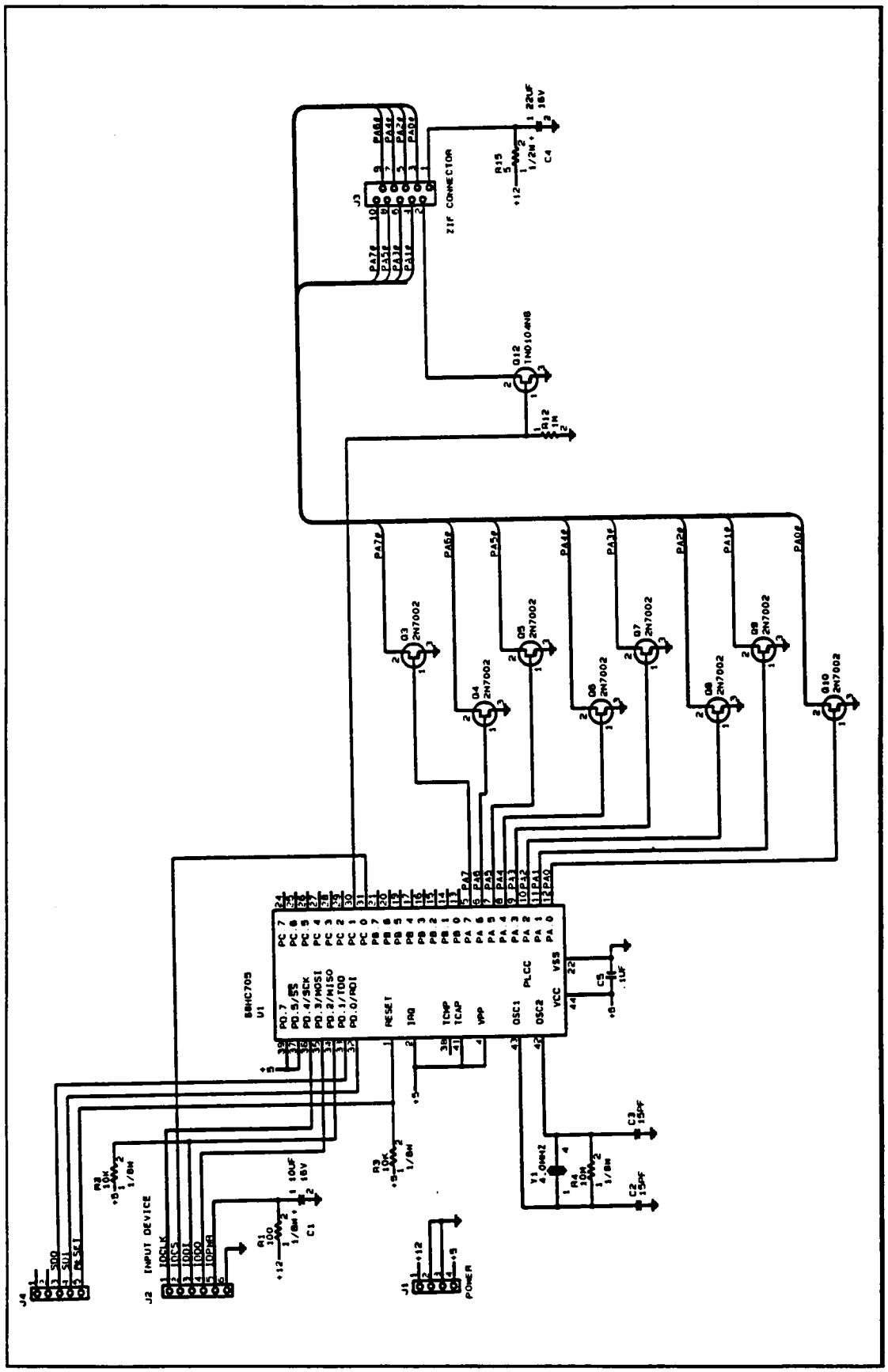
Input device final design



Icon display drivers.



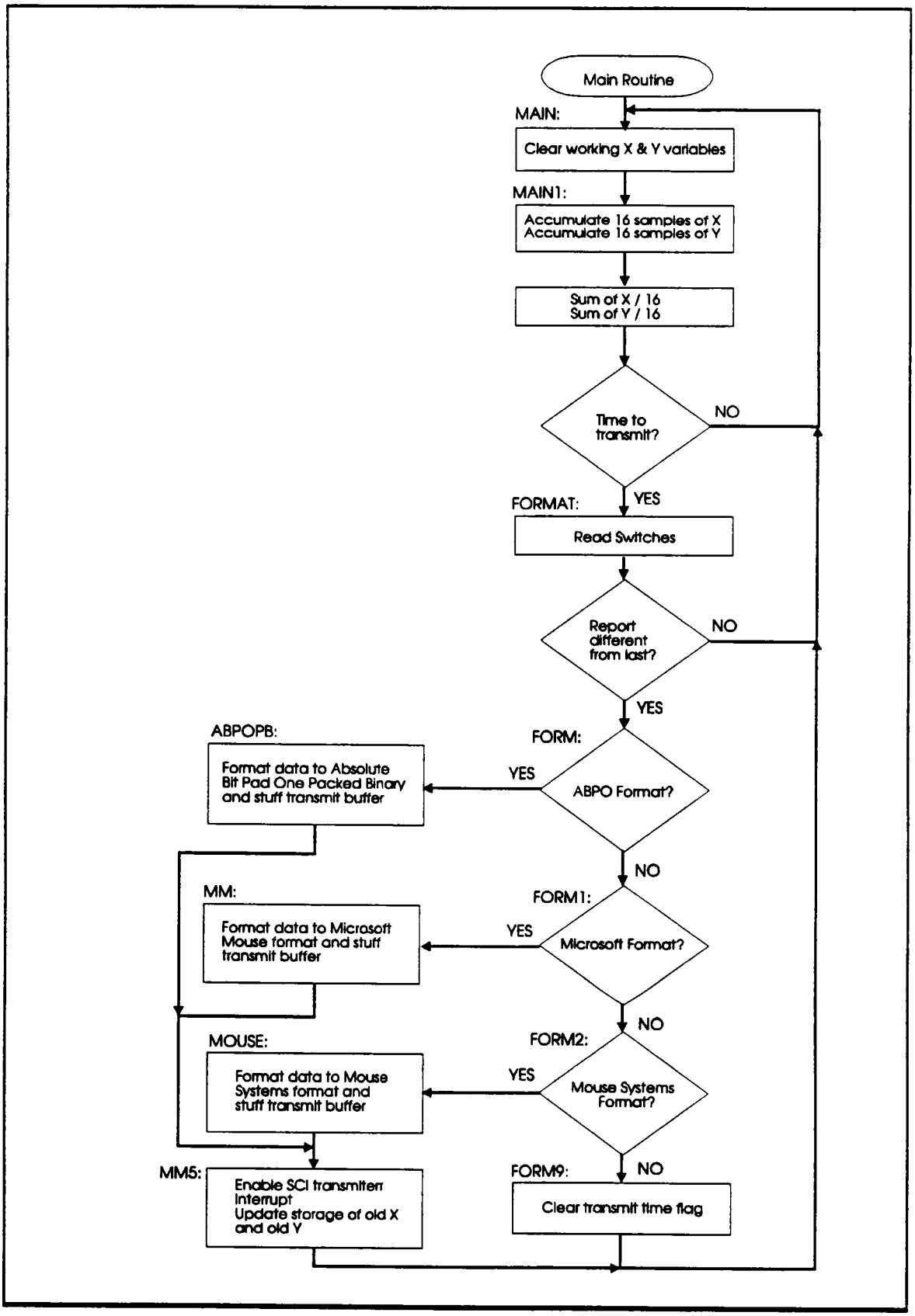
Icon display using Hewlett Packard light bars.



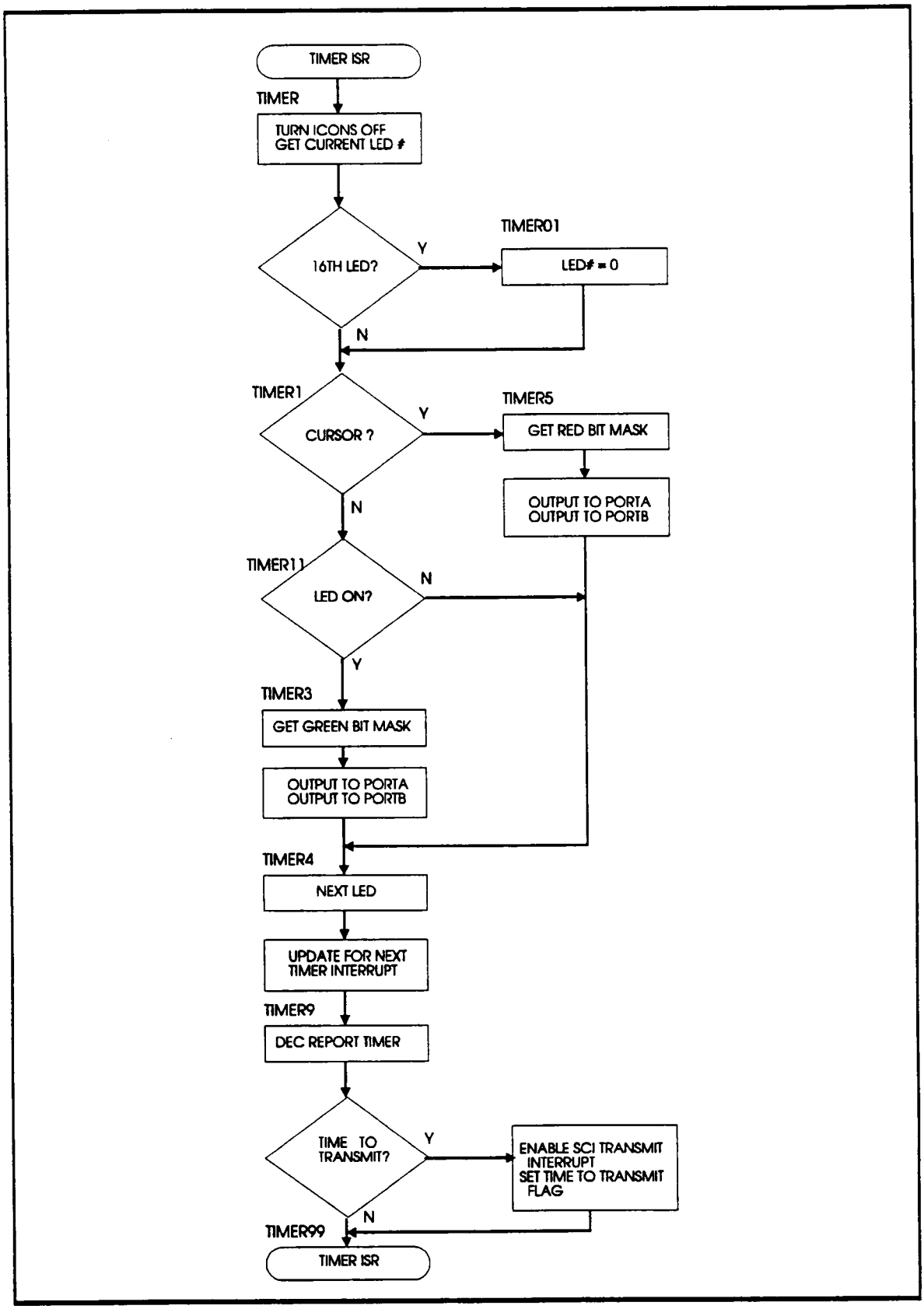
Input device and icon display controller.

APPENDIX B

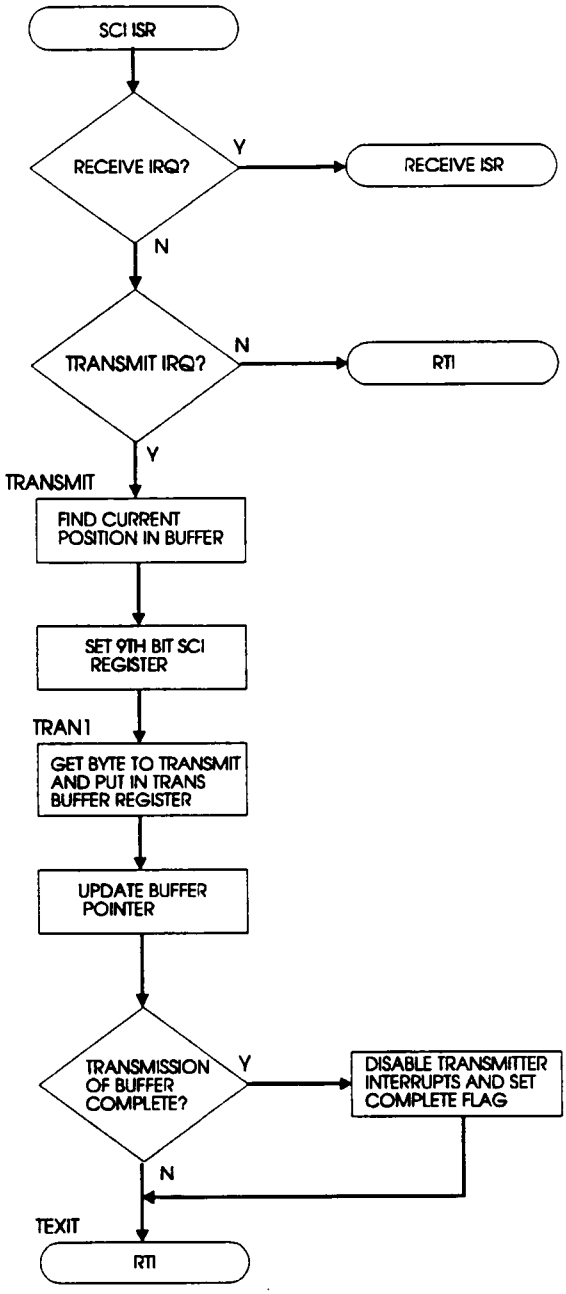
SOFTWARE FLOW DIAGRAMS



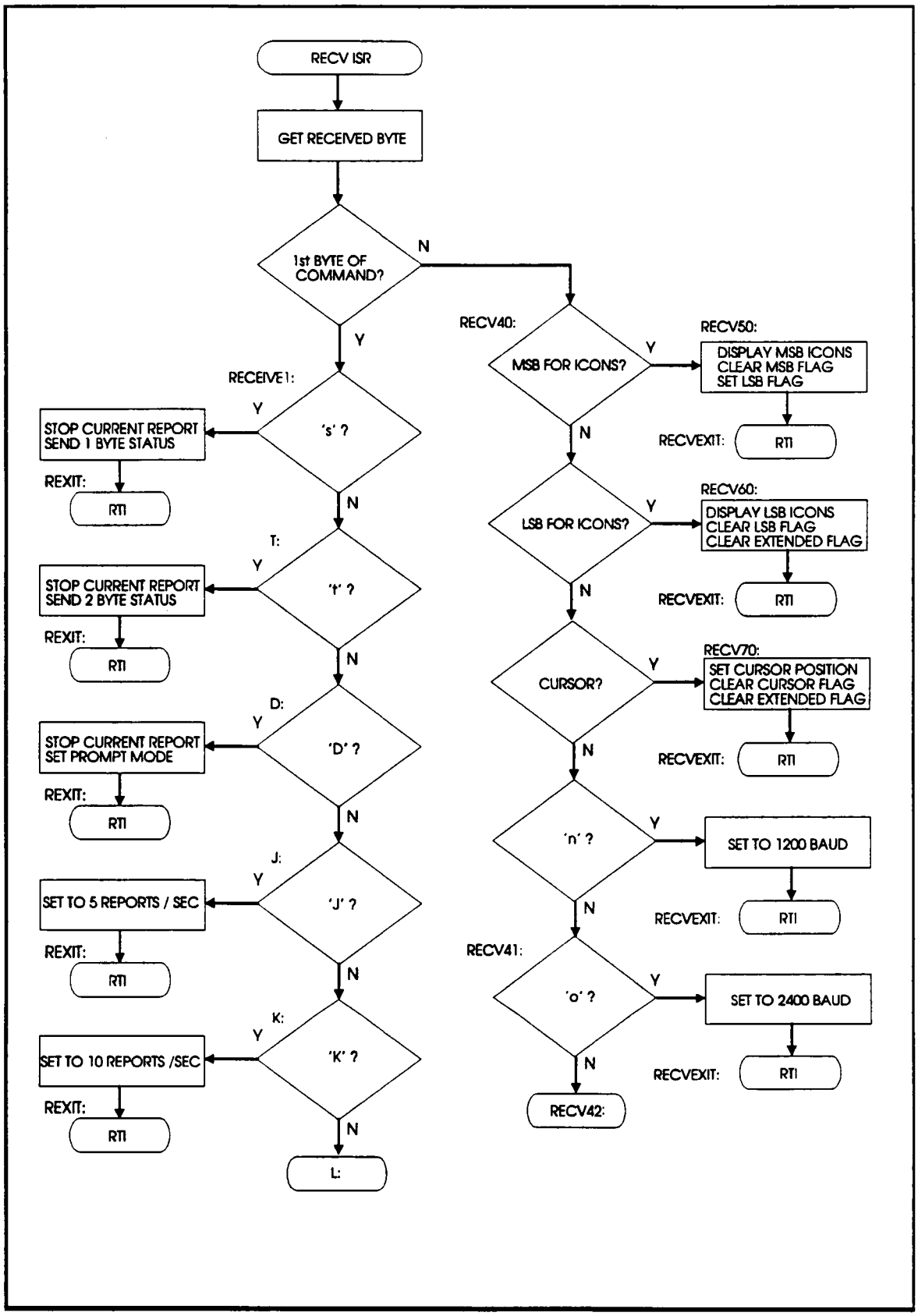
Main Routine.



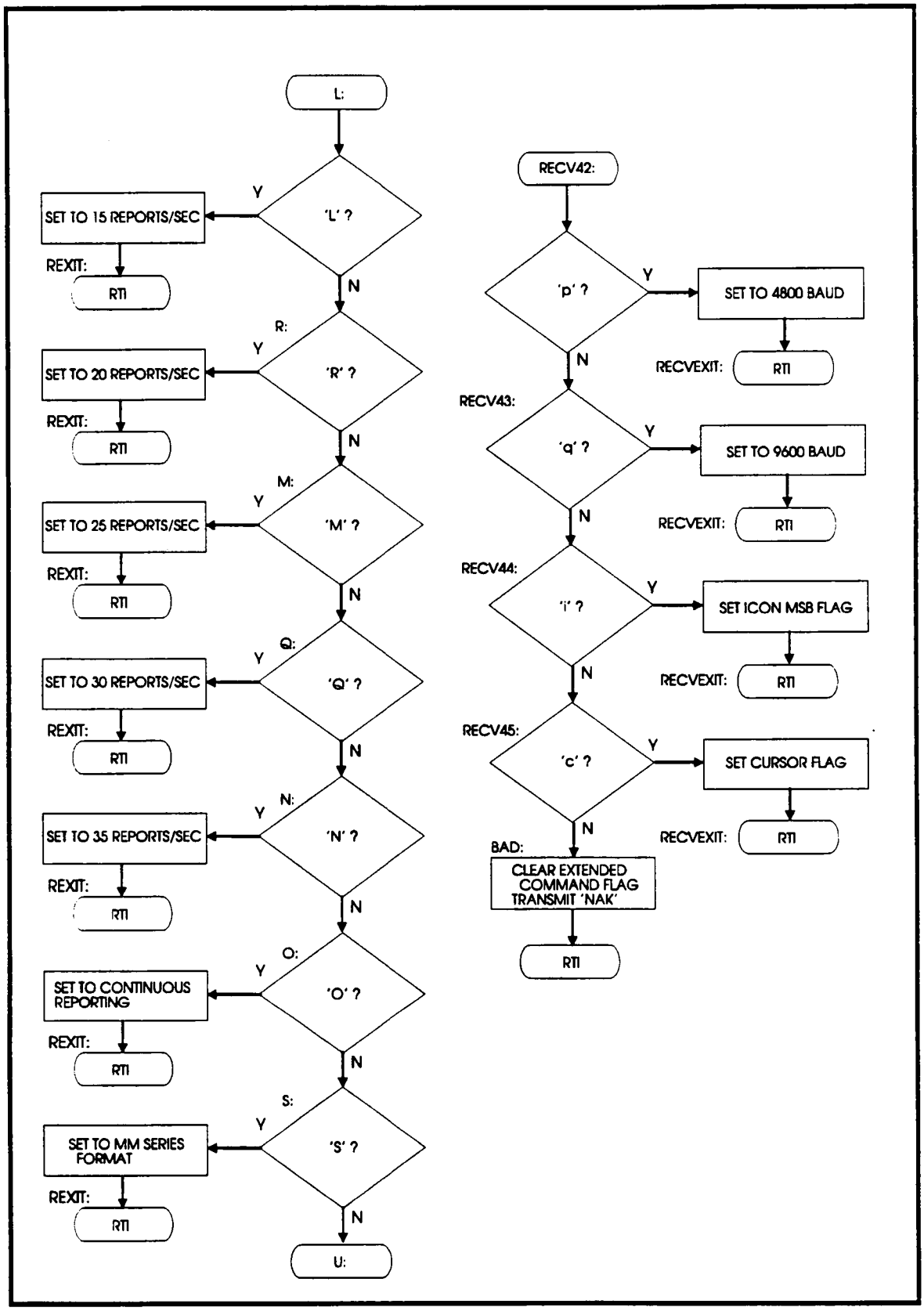
Timer Interrupt Service Routine.



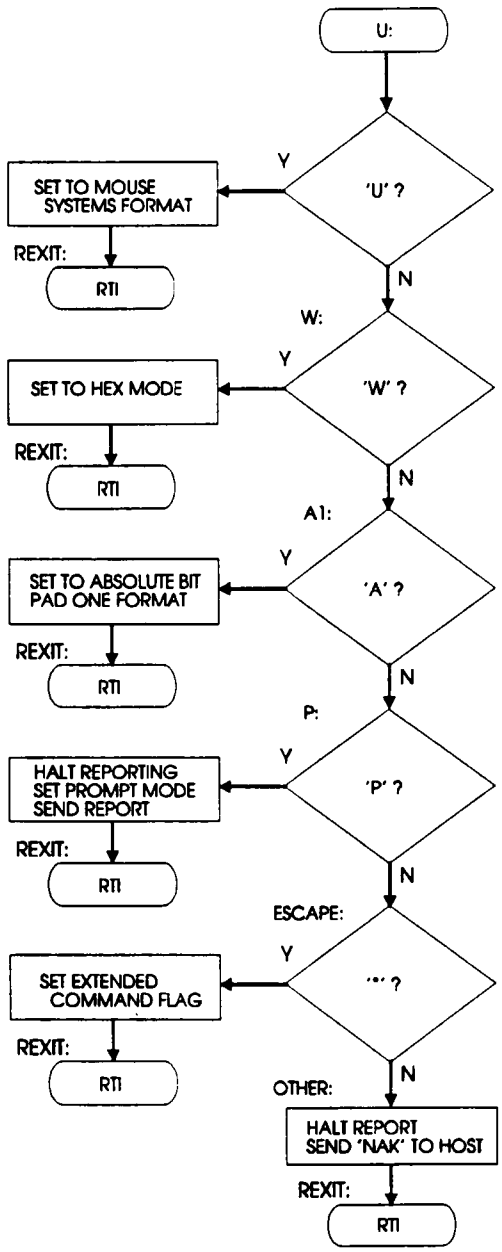
SCI Interrupt Service Routine.



SCI Receive Interrupt Service Routine.



SCI Receive Interrupt Service Routine (cont).



SCI Receive Interrupt Service Routine (cont).

APPENDIX C

SOFTWARE SOURCE LISTING

```
* tadpole.asm
* 03/06/91 RPB



* GENERAL EQUATES FOR 68HC05

ACHR    EQU     $1A
ACLR    EQU     $1B
ARH     EQU     $1E     ;68HC805C4 ONLY
ARL     EQU     $1D     ;68HC805C4 ONLY
BAUD    EQU     $0D
CHR     EQU     $18
CLR     EQU     $19
COPCR   EQU     $1E
COPRR   EQU     $1D
DDRA    EQU     $04
DDRB    EQU     $05
DDRC    EQU     $06
DDRD    EQU     $07
ICHR    EQU     $14
ICLR    EQU     $15
IRQOR   EQU     $1FDF   ;68HC705C8 ONLY
OCHR    EQU     $16
OCLR    EQU     $17
PORTA   EQU     $00
PORTB   EQU     $01
PORTC   EQU     $02
PORTD   EQU     $03
PR      EQU     $1C     ;68HC705C8 ONLY
SCCR1   EQU     $0E
SCCR2   EQU     $0F
TCIE    EQU     6
SCDAT   EQU     $11
SCSR    EQU     $10
SPCR    EQU     $0A
SPDR    EQU     $0C
SPSR    EQU     $0B
TCR     EQU     $12
TSR     EQU     $13


* EQUATES FOR BIT TESTING

BIT7    EQU     $80
BIT6    EQU     $40
BIT5    EQU     $20
BIT4    EQU     $10
BIT3    EQU     $08
BIT2    EQU     $04
BIT1    EQU     $02
BIT0    EQU     $01


* RAM VARIABLES

* X VARIABLES

XSUMMSB EQU     $50     ;Sum of X MSB
XSUMLSB EQU     $51     ;Sum of X LSB
XMSB    EQU     $52     ;Filtered X MSB
XLSB    EQU     $53     ;Filtered X LSB
OLDXMSB EQU     $54     ;Last filtered X
MSB
OLDXLSB EQU     $55     ;Last filtered X
LSB
RAWXMSB EQU     $56     ;raw A/D X MSB
RAWXLSB EQU     $57     ;raw A/D X LSB

* Y VARIABLES

YSUMMSB EQU     $58     ;Sum of Y MSB
YSUMLSB EQU     $59     ;Sum of Y LSB
YMSB    EQU     $5A     ;Filtered Y MSB
YLSB    EQU     $5B     ;Filtered Y LSB
OLDYMSB EQU     $5C     ;Last filtered Y MSB
OLDYLSB EQU     $5D     ;Last Filtered Y LSB
RAWYMSB EQU     $5E     ;raw A/D Y MSB
RAWYLSB EQU     $5F     ;raw A/D Y LSB



* FLAGS

FLAG1   EQU     $60
 ABPO   EQU     0       ;absolute bid pad one
                        ;flag
 HF     EQU     1       ;hex mode flag
 MSF    EQU     2       ;mouse systems mode
                        ;flag
 MMF    EQU     3       ;microsoft mouse flag
 TT     EQU     4       ;time to transmit flag
 TE     EQU     5       ;Transmit enabled flag
 SAME   EQU     6       ;this packet same as
                        ;last packet flag

FLAG2   EQU     $61
 EXTEND EQU     0       ;extended command flag
 ICON1  EQU     1
 ICON2  EQU     2
 CURSOR EQU     3


* MISC RAM VARIABLES

SWITCHES EQU    $62     ;current switch state
CHANNEL EQU     $63
TBYTES  EQU     $64     ;position in storage
                        ;buffer
BYTES   EQU     $65     ;bytes in storage
                        ;buffer
STATUS  EQU     $66     ;for Logitech compat.
MODE    EQU     $67
RPTIME  EQU     $69     ;time between reports
RPTIMER EQU     $6A
OLDSW   EQU     $6B     ;last switch state
CURPOS  EQU     $6C     ;position of cursor
ICONMSB EQU     $6D     ;state of top 7 icons
ICONLSB EQU     $6E     ;state of bottom 8
LEDNO   EQU     $6F     ;current LED mpx #
TIME    EQU     $70


* REPORT STORAGE BUFFER
* THIS IS WHERE DATA PACKETS ARE ASSEMBLED
* FOR TRANSMISSION TO THE HOST
* BYTES AND TBYTES CONTROL ITS USE

BUFFER  EQU     $80

-------------------------------------------------

* START OF PROGRAM

        ORG     $0200


* INIT 68HC05 FOR INPUT DEVICE AND ICON DISPLAY

*  SCI = 2400 BAUD, 9 BITS, 1 STOP BIT
*  SPI = 125 KHz, MASTER
*  SCI, AND TIMER INTERRUPTS ENABLED

INIT:
        RSP             ;RESET STACK POINTER
* SETUP FOR LTC1095 ON SPI, PORT C, AND PORT A
```

```
        BSET    0,DDRC   ;PORT C0 OUTPUT
        BSET    0,PORTC      ;LTC1095 CS
        BSET    1,DDRC       ;LED ENA OUT
        BCLR    1,PORTC      ;LEDS OFF
        BSET    7,DDRC       ;TFT ENABLE
        BCLR    7,PORTC      ;TURN ON TFT
        BSET    6,DDRC       ;LCD ENABLE
        BSET    6,PORTC      ;LCD OFF
        LDA     #$FF
        STA     DDRA     ;PORT A ICON OUT
        STA     DDRB     ;PORT B ICON OUT
        LDA     #$52     ;ENA SPI,125KHz
        STA     SPCR

* SETUP SCI TO 2400 BAUD, 1 START, 1 STOP
*    AND 9 DATA BITS  (FOR PARITY)
        LDA     #$32
        STA     BAUD          ;2400 BAUD
        LDA     #$50
        STA     SCCR1    ;1 START, 9 DATA
                         ;  1 STOP
        LDA     #$2C          ;RIE, TE, RE
        STA     SCCR2

* SETUP TIMER
        LDA     #$40
        STA     TCR       ;ENABLE OUTPUT
                     ;COMPARE INTERRUPT

* INITIALIZE SYSTEM VARIABLES
        CLR     FLAG1    ;CLEAR ALL FLAGS
        CLR     FLAG2
        BSET    ABPO,FLAG1 ;DEFAULT BIT
                         ;PAD ONE FORMAT
        LDA     #3
        STA     STATE
        CLR     XMSB          ;CLEAR X
        CLR     XLSB
        CLR     YMSB          ;CLEAR Y
        CLR     YLSB
        CLR     OLDXMSB       ;CLEAR OLD X
        CLR     OLDXLSB
        CLR     OLDYMSB       ;CLEAR OLD Y
        CLR     OLDYLSB
        LDA     #$0F          ;FOR LOGITECH
                              ;COMPATIBILITY
        STA     STATUS
        CLR     LEDNO
        LDA     #$FF
        STA     ICONMSB     ;TURN LEDS ON
        STA     ICONLSB
        CLI              ;ENABLE INTERRUPTS
```

```
* MAIN ROUTINE
* SAMPLE A/D, READ SWITCHES, FORMAT DATA,
* AND TRANSMIT TO HOST

MAIN:
        CLR     XSUMMSB      ;INIT X AND Y
        CLR     XSUMLSB     ;SUM VARIABLES
        CLR     YSUMMSB
        CLR     YSUMLSB
        LDX     #$10         ;16 SAMPLES TO
                             ;   TAKE
MAIN1:
        JSR     READADC      ;READ X AND Y
                             ;FROM LTC1095
        LDA     XSUMLSB
        ADD     RAWXLSB        ;SUM X LSB
        STA     XSUMLSB        ;16 BIT ADD
        LDA     XSUMMSB   ;WITH 8 BIT ALU
        ADC     RAWXMSB        ;SUM X MSB
        STA     XSUMMSB
        LDA     YSUMLSB
        ADD     RAWYLSB        ;SUM Y LSB
        STA     YSUMLSB
        LDA     YSUMMSB
        ADC     RAWYMSB        ;SUM Y MSB
        STA     YSUMMSB
        DECX
        BNE     MAIN1          ;SUM 16 SAMPLES OF
                               ; X AND Y

* DIVIDE X BY 16 TO GET ACTUAL X READING

        LDA     XSUMLSB        ;X LSB IN A
        LDX     XSUMMSB        ;X MSB IN X
        LSRX                   ;16 BIT DIVIDE
        RORA                   ;/2
        LSRX
        RORA                   ;/4
        LSRX
        RORA                   ;/8
        LSRX
        RORA                   ;/16
        STX     XSUMMSB        ;STORE RESULT
        STA     XSUMLSB

* DIVIDE Y BY 16 TO GET ACTUAL Y READING

        LDA     YSUMLSB        ;Y LSB IN A
        LDX     YSUMMSB        ;Y MSB IN X
        LSRX
        RORA                   ;/2 - 16 BIT
        LSRX
        RORA                   ;/4
        LSRX
        RORA                   ;/8
        LSRX
        RORA                   ;/16
        STX     YSUMMSB        ;STORE RESULTS
        STA     YSUMLSB

* AVERAGE CURRENT X READING WITH PREVIOUS
*   THIS PROVIDES ADDITIONAL FILTERING

        LDA     XSUMLSB        ;ADD CURRENT X
        ADD     XLSB           ;WITH PREVIOUS
        STA     XLSB
        LDA     XSUMMSB
        ADC     XMSB
        LSRA                   ;/2 HIGH BYTE
        STA     XMSB
        ROR     XLSB           ;/2 LOW BYTE

* AVERAGE CURRENT Y READING WITH PREVIOUS

        LDA     YSUMLSB        ;ADD CURRENT Y LSB
        ADD     YLSB           ;WITH PREVIOUS
        STA     YLSB
        LDA     YSUMMSB
        ADC     YMSB           ;MSB
        LSRA
        STA     YMSB           ;/2 MSB
        ROR     YLSB           ;/2 LSB

* GO BACK IF NOT TIME TO TRANSMIT PACKET
        BRCLR   TT,FLAG1,MAIN

* FORMAT AND TRANSMIT DATA
        JSR     FORMAT

* GO BACK AND LOOP CONTINUOUSLY
        BRA     MAIN
```

```
READADC:

* READ X POSITION FROM LTC1095 10 BIT ADC
*   CONNECTED TO SPI -- LTC CHANNEL 0
READX:
        BCLR    0,PORTC     ;ASSERT LTC CS
        LDA     #$03         ;START + SGL
        STA     SPDR         ;SEND COMMAND
READX1:
        TST     SPSR     ;CHECK SPI STATUS
        BPL     READX1   ;WAIT TIL SPI RDY
        LDA     #$18     ;SELECT CH0, UNI-
                         ;POLAR, MSB FIRST
        STA     SPDR     ;SEND COMMAND
READX2:
        TST     SPSR     ;CHECK SPI STATUS
        BPL     READX2   ;WAIT TIL SPI RDY
        LDA     SPDR      ;SPI HAS 2 MSbs
        STA     SPDR    ;DUMMY COMMAND FOR
                        ; REAMINING BITS
        AND     #$03    ;MSK ALL BT 2 MSbS
        STA     RAWXMSB       ;MSB STORAGE
READX3:
        TST     SPSR     ;WAIT TIL SPI RDY
        BPL     READX3
        LDA     SPDR     ;GET LSB FROM LTC
        STA     RAWXLSB       ;LSB STORAGE
        BSET    0,PORTC   ;DIS LTC1095 CS

* READ Y POSITION FROM LTC1095 10 BIT ADC
*  CONNECTED TO SPI -- LTC CHANNEL 1

READY:
        BCLR    0,PORTC     ;ASSERT LTC CS
        LDA     #$03         ;START + SGL
        STA     SPDR         ;SEND COMMAND
READY1:
        TST     SPSR     ;CHECK SPI STATUS
        BPL     READY1   ;WAIT TIL SPI RDY
        LDA     #$98     ;SELECT CH1, UNI-
                         ;POLAR, MSB FIRST
        STA     SPDR     ;SEND COMMAND
READY2:
        TST     SPSR     ;CHECK SPI STATUS
        BPL     READY2   ;WAIT TIL SPI RDY
        LDA     SPDR      ;SPI HAS 2 MSbs
        STA     SPDR    ;DUMMY COMMAND FOR
                        ; REAMINING BITS
        AND     #$03    ;MSK ALL BT 2 MSbs
        STA     RAWYMSB       ;MSB STORAGE
READY3:
        TST     SPSR     ;WAIT TIL SPI RDY
        BPL     READY3
        LDA     SPDR          ;GET LSB
        STA     RAWYLSB       ;LSB STORAGE
        BSET    0,PORTC   ;DIS LTC1095 CS
        RTS


* READ SWITCHES ATTACHED TO LTC1095
*  SW1 = CHANNEL2
*  SW2 = CHANNEL3
*  SW3 = CHANNEL4
*  SW4 = CHANNEL5
*  SWITCH STATUS IS PLACED IN LAST 4 BITS
*  OF 'SWITCHES'  (1=SWITCH OPEN)
READSW:
        CLR     SWITCHES
        LDA     #$38          ;CH2, SW1
        BSR     SWITCH
        LDA     #$B8          ;CH3, SW2
        BSR     SWITCH
        LDA     #$58          ;CH4, SW3
        BSR     SWITCH
        LDA     #$D8          ;CH5, SW4
        BSR     SWITCH
        RTS

* SUBROUTINE WHICH READS ANALOG CHANNEL AND
*  USES MSB TO DETERMINE WHETHER SWITCH IS
*   OPEN OR CLOSED

SWITCH:
        BCLR    0,PORTC       ;ASSERT LTC CS
        STA     CHANNEL       ;SAVE CHANNEL
        LDA     #$03          ;START + SGL
        STA     SPDR          ;SEND COMMAND
SWITCH1:
        TST     SPSR          ;CHECK SPI STATUS
        BPL     SWITCH1       ;WAIT TIL SPI RDY
        LDA     CHANNEL       ;SELECT CHX, UNI-
                              ;POLAR, MSB FIRST
        STA     SPDR
SWITCH11:
        TST     SPSR
        BPL     SWITCH11
        LDA     SPDR          ;SPI HAS 2 MSBs
        STA     SPDR          ;GET ALL 8 BITS
        COMA                  ;1=SWITCH CLOSED
        RORA
        RORA                  ;PUT MSb IN CARRY
        ROL     SWITCHES      ;SAVE SWITCH INFO
SWITCH12:
        TST     SPSR
        BPL     SWITCH12      ;WAIT TIL SPI RDY
        BSET    0,PORTC       ;RELEASE CS
                              ; SWITCH 1 DONE

        RTS                   ;RETURN FROM SUB
```

```
* SCI INTERRUPT SERVICE ROUTINE
* RECEIVES COMMANDS FROM HOST FOR BOTH ICON
*    AND TADPOLE
* TRANSMITS REPORTS TO HOST FROM TADPOLE

SCIISR:
        BRSET   5,SCSR,RECEIVE    ;RECEIVE
        BRSET   6,SCSR,TRANSMIT   ;TRANSMIT
        RTI                      ;BAD INT?

* TRANSMIT ISR
* THIS ROUTINE PASSES DATA FROM 'BUFFER' TO
* SCI OUTPUT PORT
* 'BUFFER' HOLDS 2 BYTES, 2ND IS PARITY


TRANSMIT:
        LDA     TBYTES        ;# OF BYTES TRAN'D
        ASLA                  ;2 BYTES PER BYTE
                              ;OUT(PARITY)
        ADD     #BUFFER       ;START O TRANS BUF
        TAX                   ;X POSITION IN BUF
        LDA     ,X
        BSET    6,SCCR1       ;ASSUME PARITY SET
        RORA
        BCS     TRAN1         ;PARITY CORRECT
        BCLR    6,SCCR1       ;CLEAR 9TH BIT
TRAN1:
        LDA     1,X
        STA     SCDAT         ;TRANSMIT BYTES
        INC     TBYTES
        LDA     TBYTES        ;UPDATE BUF COUNT
        SUB     BYTES
        BNE     TEXIT         ;SET COMPLETE FLAG
```

```
        BCLR    TCIE,SCCR2 ;DISABLE INT
                           ;BUFFER TRANS'D
        CLR     TBYTES
TEXIT:
        RTI                ;RETURN FROM INT


* RECEIVE ISR
* SOME COMMANDS HAVE MORE THAN ONE BYTE
AND
* SO THE STATE MUST BE MAINTAINED

RECEIVE:
        LDA     SCDAT  ;GET RECEIVED BYTE

        BRCLR   EXTEND,FLAG2,RECEIVE1
; GO IF THIS IS FIRST BYTE OF COMMAND

        JMP     RECV40
; EXTENDED COMMANDS HANDLER

RECEIVE1:
        CMP     #'s'           ;STATUS
COMMAND
        BNE     T
        BCLR    TCIE,SCCR2
; STOP CURRENT REPORT BEING SENT
; IF THIS OCCURS IN THE MIDDLE OF A
; REPORT, THE REST OF THE REPORT WILL
; BE LOST.

        LDA     STATUS
        STA     SCDAT          ;1 B STATUS
        JMP     REXIT
T:
        CMP     #'t'           ;2 BYTE MODE
        BNE     D
        BCLR    TCIE,SCCR2     ;STOP REPORT
        LDA     MODE       ;FIRST MODE BYTE
        STA     SCDAT    ;TRANSMIT TO HOST
T1:
        LDA     SCSR     ;CHK TRANS STATUS
        BPL     T1      ;WT TIL TRANS DONE
        LDA     MODE+1      ;2ND MODE BYTE
        STA     SCDAT
        JMP     REXIT
D:
        CMP     #'D'           ;PROMPT MODE
        BNE     J
        BCLR    TE,FLAG1       ;DIS REPORT
        BCLR    TCIE,SCCR2     ;HALT REPORT
        BSET    6,STATUS ;SET PROMPT MODE
        JMP     REXIT
J:
        CMP     #'J'           ;5 REPORTS/S
        BNE     K
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS ;INC STREAM MODE
        BSET    TE,FLAG1     ;ENABLE TRANS
        LDA     #255
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
K:
        CMP     #'K'           ;10 REPS/S
        BNE     L
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS ;INC STREAM MODE
        BSET    TE,FLAG1     ;ENABLE TRANS
        LDA     #200
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
L:
        CMP     #'L'           ;15 REPS/S
        BNE     R
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG1       ;ENABLE TRANS
        LDA     #133
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
R:
        CMP    #'R'            ;20 REPORTS/SECOND
        BNE     M
        STA     MODE           ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG1       ;ENABLE TRANS
        LDA     #100
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
M:
        CMP     #'M'           ;25 REPORTS/SECOND
        BNE     Q
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG        ;ENABLE TRANS
        LDA     #80
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
Q:
        CMP     #'Q'           ;30 REPORTS/SECOND
        BNE     N
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG1       ;ENABLE TRANS
        LDA     #67
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
N:
        CMP     #'N'           ;35 REPORTS/SECOND
        BNE     O
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG1       ;ENABLE TRANS
        LDA     #57
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
O:
        CMP     #'O'           ;CONT REPORTING
        BNE     S
        STA     MODE+1         ;SAVE MODE
        BCLR    6,STATUS       ;INCR STREAM MODE
        BSET    TE,FLAG1       ;ENABLE TRANS
        LDA     #1
        STA     RPTIME         ;SETUP TIMER
        JMP     REXIT          ;EXIT
S:
        CMP     #'S'           ;MM SERIES FORMAT
        BNE     U
        STA     MODE           ;SAVE MODE
        LDA     FLAG1
        AND     #$F0
        STA     FLAG1          ;CLR FORMAT FLAGS
        BSET    MMF,FLAG1      ;SET MM FLAG
        JMP     REXIT          ;EXIT
U:
        CMP     #'U'           ;MOUSE SYSTEMS
                               ;  FORMAT
        BNE     W
        STA     MODE           ;SAVE MODE
        LDA     FLAG1
        AND     #$F0           ;CLR FORMAT FLAGS
        STA     FLAG1
        BSET    MSF,FLAG1      ;SET M S FLAG
        JMP     REXIT          ;EXIT
W:
        CMP     #'W'           ;HEX MODE
        BNE     A1
        STA     MODE           ;SAVE MODE
```

```
        LDA     FLAG1
        AND     #$F0    ;CLR FORMAT FLAGS
        STA     FLAG1
        BSET    HF,FLAG1    ;SET HEX FLAG
        JMP     REXIT       ;EXIT
A1:
        CMP     #'A'    ;ABS BIT PAD ONE
        BNE     P
        STA     MODE        ;SAVE MODE
        LDA     FLAG1
        AND     #$F0    ;CLR FORMAT FLAGS
        STA     FLAG1
        BSET    ABPO,FLAG1 ;SET ABPO FLAG
        JMP     REXIT       ;EXIT
P:
        CMP     #'P'   ;PROMPT FOR REPORT
        BNE     ESCAPE
        BCLR    TE,FLAG1    ;HALT REPORT
        BCLR    TCIE,SCCR2
        BSET    6,STATUS ;SET PROMPT MODE
        JSR     READSW
        JSR     FORM    ;FORMAT AND SEND
                        ;REPORT
        JMP     REXIT
ESCAPE:
        CMP     #'*'   ;SPEC COMMAND MODE
        BNE     OTHER
        BSET    EXTEND,FLAG2
               ;SET EXTENDED COMMAND MODE
        JMP     REXIT       ;EXIT
OTHER:
* SENDS NAK IF BAD CHARACTER RECEIVED

        BCLR    TCIE,SCCR2  ;HALT REPORT
        LDA     #$15        ;NAK
        STA     SCDAT       ;SEND NAK
REXIT:
        RTI             ;RETURN FROM INT

*
* EXTENDED COMMANDS HANDLER
*

RECV40:
        BRSET   ICON1,FLAG2,RECV50
                        ; MSB FOR ICONS
        BRSET   ICON2,FLAG2,RECV60
                        ;LSB FOR ICONS
        BRSET   CURSOR,FLAG2,RECV70
                        ;CURSOR POSITION

        CMP     #'n'    ;SET TO 1200 BPS
        BNE     RECV41
        LDA     #$33
        STA     BAUD
        BCLR    EXTEND,FLAG2;CLR EXT MODE
        BCLR    TCIE,SCCR2  ;STOP REPORT
        BRA     RECVEXIT            ;EXIT
RECV41:
        CMP     #'o'            ;2400 BAUD
        BNE     RECV42
        LDA     #$32
        STA     BAUD
        BCLR    EXTEND,FLAG2
        BCLR    TCIE,SCCR2  ;STOP REPORT
        BRA     RECVEXIT
RECV42:
        CMP     #'p'            ;4800 BAUD
        BNE     RECV43
        LDA     #$31
      STA     BAUD
        BCLR    EXTEND,FLAG2
        BCLR    TCIE,SCCR2
        BRA     RECVEXIT            ;EXIT
RECV43:
        CMP     #'q'            ;9600 BAUD
        BNE     RECV44
        LDA     #$30            ;9600 BPS
        STA     BAUD
        BCLR    EXTEND,FLAG2        ;CLR EXT MODE
        BCLR    TCIE,SCCR2          ;STOP REPORT
        BRA     RECVEXIT
RECV44:
        CMP     #'i'                ;ICON COMMAND
        BNE     RECV45
        BSET    ICON1,FLAG2
        ; NEXT BYTE IS FOR ICONS

        BRA     RECVEXIT            ;EXIT
RECV45:
        CMP     #'c'                ;CURSOR CMD
        BNE     BAD
        BSET    CURSOR,FLAG2
        ; NEXT BYTE IS FOR CURSOR

        BRA     RECVEXIT            ;EXIT

BAD:
* IF BAD CHARACTER RECEIVED, SEND NAK AND
*   EXIT EXTENDED MODE

        BCLR    EXTEND,FLAG2        ;CLR EXT MODE
        BCLR    TCIE,SCCR2          ;STOP REPORT
        LDA     #$21                ;NAK
        STA     SCDAT               ;SEND NAK
RECVEXIT:
        RTI                         ;RET FROM INT

RECV50:
        STA     ICONMSB
                        ; SET AVAILABLE ICONS MSB
        BCLR    ICON1,FLAG2
                        ; CLR 1ST BYTE FLAG
        BSET    ICON2,FLAG2
                        ; SET SECOND BYTE FLAG
        BRA     RECVEXIT     ;EXIT
RECV60:
        STA     ICONLSB
                        ; SET AVAILABLE ICONS LSB
        BCLR    ICON2,FLAG2
                        ; CLEAR 2ND BYTE FLAG
        BCLR    EXTEND,FLAG2 ;CLR EXT MODE
        BRA     RECVEXIT     ;EXIT
RECV70:
        STA     CURPOS       ;SET CSR POS
        BCLR    CURSOR,FLAG2 ;CLR CSR FLAG
        BCLR    EXTEND,FLAG2
        BRA     RECVEXIT
```

```
* FORMAT AND TRANSMIT SUBROUTINE
* THIS ROUTINE FORMATS THE X AND Y POSITION
* AND SWITCH DATA ACCORDING TO THE MODE
* SELECTED ( I.E. MM, MOUSE, ABPO, ETC)
* FORMATTED DATA IS PUT IN 'BUFFER'

FORMAT:
*   CHECK TO SEE IF REPORT IS DIFFERENT
*    BEFORE TRANSMITTING
        JSR     READSW      ;READ SWICHES
        LDA     SWITCHES
        CMP     OLDSW       ;CMP OLD TO NEW
        BNE     FORM        ;SEND IF DIFFERENT
        LDA     XLSB        ; GET X LSB
        CMP     OLDXLSB     ; AND CMP TO OLD
        BNE     FORM        ;SEND IF DIFFERENT
        LDA     XMSB        ;GET X MSB
```

```
        CMP     OLDXMSB   ;AND CMP TO OLD
        BNE     FORM      ;SEND IF DIFFER
        LDA     YLSB          ;GET Y LSB
        CMP     OLDYLSB   ;AND CMP TO OLD
        BNE     FORM      ;SEND IF DIFFER
        LDA     YMSB          ;GET Y MSB
        CMP     OLDYMSB       ;CMP TO OLD
        BEQ     FORM9 ;DONT SEND IF SAME

FORM:
        LDA     SWITCHES      ; UPDATE OLD
        STA     OLDSW         ;SWITCH STOR

        BRCLR   ABPO,FLAG1,FORM1
            ;CHECK FOR BIT PAD ONE FORMAT
        JMP     ABPOPB
                ; DO BIT PAD ONE FORMAT
FORM1:
        BRCLR   MMF,FLAG1,FORM2
             ; CHECK FOR MM SERIES FORMAT

        JMP     MM            ;DO MM
SERIES
FORM2:
        BRCLR   MSF,FLAG1,FORM3
          ; CHECK FOR MOUSE SYSTEMS FORMAT
        JMP     MOUSE
        ; DO MOUSE SYSTEMS FORMAT
FORM3:
FORM9:
        BCLR    TT,FLAG1 ;TRANS TIMR FLG
        RTS              ;RETURN FROM SUB

*
* MOUSE SYSTEMS CORP FORMAT SUBROUTINE
*

MOUSE:
        LDA     #$05   ;5 BYTES IN REPORT
        STA     BYTES
        LDA     SWITCHES ;GET SWITCH INFO
        ORA     #$80
        STA     BUFFER+1
        LDA     XLSB          ;X LSB
        SUB     OLDXLSB
        STA     BUFFER+3   ;STORE X DELTA
        LDA     YLSB       ;Y LSB
        SUB     OLDYLSB
        STA     BUFFER+5      ;STORE Y
DELTA
        CLR     BUFFER+7
        CLR     BUFFER+9
        LDA     #$01          ;PARITY BIT
        STA     BUFFER      ;PAR 1ST BYTE
        STA     BUFFER+2 ;PARITY 2ND BYTE
        STA     BUFFER+4 ;PARITY 3RD BYTE
        STA     BUFFER+6 ;PARITY 4TH BYTE
        STA     BUFFER+8 ;PARITY 5TH BYTE
        JMP     MM5           ;EXIT
```

```
* MM SERIES FORMAT SUBROUTINE (DEFAULT
*  WITH LOGITECH DRIVER)

MM:
        LDA     #$03          ;REPORT 3
BYTES
        STA     BYTES
        LDA     SWITCHES  ;GET SWI STAT
        COMA              ;1=SWITCH CLOSED
        AND     #$0F      ;MASK XTRA BITS
        LSRA                  ;LEFT 3 SWITCHES
        ORA     #$80
        STA     BUFFER+1
        JSR     PARITY        ;CALC EVEN PARITY
        CLRX
        ROLX                  ;PARITY IN X
        INCX                  ;MAKE ODD PARITY
        STX     BUFFER        ;SAVE PARITY
        BSET    4,BUFFER+1    ;ASSUME X DELTA +
        LDA     XLSB          ;CURRENT X LSB
        SUB     OLDXLSB       ;LAST X LSB
        STA     BUFFER+3      ;DELTA X
        LDA     XMSB
        SBC     OLDXMSB       ;FINISH SUB OF X
        BCC     MM2           ;X DELTA POSITIVE
        BCLR    4,BUFFER+1    ;X DELTA NEGATIVE
        COM     BUFFER+3
        INC     BUFFER+3      ;ABS VAL OF DELTA
MM2:
* CHECK IF OVERFLOW  (I.E. DELTA X > 127)
        LDA     BUFFER+3
        BMI     X2BIG         ;DELTA X TOO BIG
                              ;IF MSB SET
        JSR     PARITY        ;CALC EVEN PARITY
        ROLX
        INCX                  ;MAKE PARITY ODD
        STX     BUFFER+2
        BRA     MM3           ;DO Y
X2BIG:
        LDA     #$7F          ;MAXIMUM DELTA
        STA     BUFFER+3      ;DELTA = MAX
        BRSET   4,BUFFER+1,X2BIG1
                              ;DELTA IS POSITIVE
        LDA     OLDXLSB       ;CALC NEW X LSB
        SUB     #$7F
        STA     XLSB          ;X LSB ADJUSTED
        LDA     OLDXMSB
        SBC     #0            ;FINISH SUB OF MSB
        STA     XMSB
        BRA     MM2           ;X FIXED - CONTINU
X2BIG1:
        LDA     OLDXLSB       ;ADJUST X LSB
        ADD     #$7F
        STA     XLSB          ;X LSB ADJUSTED
        LDA     OLDXMSB
        ADC     #$00          ;FINISH ADD OF MSB
        STA     XMSB
        BRA     MM2           ;X COMPLETED
MM3:
* CALCULATE DELTA Y
        BSET    3,BUFFER+1    ;ASSUME DELTA Y +
        LDA     YLSB
        SUB     OLDYLSB       ;CALC DELTA Y
        STA     BUFFER+5
        LDA     YMSB          ;COMPLETE MSB
        SBC     OLDYMSB
        BCC     MM4           ;ASSUMED CORRECT
        BCLR    3,BUFFER+1    ;CLEAR Y + FLAG
        COM     BUFFER+5
        INC     BUFFER+5      ;ABS OF DELTA Y
MM4:
        LDA     BUFFER+5
        BMI     Y2BIG         ;DELTA Y OVERFLOW
        JSR     PARITY        ;CALC ODD PARITY
        ROLX
        INCX                  ;MAKE PARITY ODD
        STX     BUFFER+4      ;STORE PARITY
        BRA     MM5
Y2BIG:
* Y OVERFLOWED SO MUST FIX Y COORDINATES
        LDA     #$7F          ;MAX DELTA ALLOWED
        STA     BUFFER+5      ;DELTA Y SET TO MAX
        BRSET   3,BUFFER+1,Y2BIG1
                          ;GO IF DELTA POSITIVE
```

```
        LDA     OLDYLSB
        SUB     #$7F
        STA     YLSB            ;NEW Y LSB
        LDA     OLDYMSB
        SBC     #$00  ;FINISH SUB OF MSB
        STA     YMSB            ;NEW Y MSB
        BRA     MM4             ;FIXED
Y2BIG1:
        LDA     OLDYLSB
        ADD     #$7F
        STA     YLSB            ;NEW Y LSB
        LDA     OLDYMSB
        ADC     #$0
        STA     YMSB            ;NEW Y MSB
        BRA     MM4             ;Y FIXED
MM5:
        BSET    TCIE,SCCR2
            ;ENABLE REPORT TRANSMISSION
        BCLR    TT,FLAG1

        LDA     XLSB
        STA     OLDXLSB  ;UPDATE PREVIOUS

        LDA     XMSB
        STA     OLDXMSB
        LDA     YLSB
        STA     OLDYLSB
        LDA     YMSB
        STA     OLDYMSB
        RTS

* ABSOLUTE BIT PAD ONE PACKED BINARY
* FORMAT SUBROUTINE

ABPOPB:
        LDA     #$05  ;5 BYTES IN REPORT
        STA     BYTES
        LDA     SWITCHES ;GET SWITCH DATA
        COMA            ;1=SWITCH PRESSED
        ASLA             ;MOVE BITS INTO
        ASLA             ;PROPER POSITION
        AND     #$3C     ;MASK UNWANTED
        ORA     #$80
        JSR     PARITY  ;CALC EVEN PARITY
        RORA               ;PARITY IN MSb
        STA     BUFFER+1
        LDA     #$01    ;9TH BIT IS EXTRA
        STA     BUFFER    ; STOP BIT
        LDX     XMSB          ;X MSB
        LDA     XLSB          ;X LSB
        ASLA           ;MAKE A 6 BITS
        ROLX           ;AND MAKE X 4 BITS
        ASLA
        ROLX
        ASLX           ;SHIFT FOR PARITY
        LSRA
        STX     BUFFER+5
        JSR     PARITY  ;CALC EVEN PARITY
        RORA           ;PUT PARITY IN MSb
        STA     BUFFER+3
        LDA     BUFFER+5
        JSR     PARITY  ;CALC EVEN PARITY
        RORA            ;PUT PARITY IN MSb
        STA     BUFFER+5
        LDA     #$01        ;XTRA STOP BIT
        STA     BUFFER+2
        STA     BUFFER+4
        LDX     YMSB          ;Y MSB
        LDA     YLSB          ;Y LSB
        ASLA             ;MAKE A 6 BITS
        ROLX           ;AND MAKE X 4 BITS
        ASLA
        ROLX
        LSRA            ;SHIFT PARITY BIT
        ASLX
        STX     BUFFER+9
        JSR     PARITY      ;CALC EVEN PARITY
        RORA                ;PUT PARITY IN MSb
        STA     BUFFER+7
        LDA     BUFFER+9
        JSR     PARITY      ;CALC EVEN PARITY
        RORA                ;PUT PARITY IN MSb
        STA     BUFFER+9
        LDA     #$01        ;9TH BIT IS XTRA
        STA     BUFFER+6    ;STOP BIT
        STA     BUFFER+8
        JMP     MM5         ;EXIT
```

```
* TIMER INTERRUPT SERVICE ROUTINE
* THIS ROUTINE SETS TIMER INTERRUPTS TO .5mS,
* MULTIPLEXES LED ICON
* OUTPUT, AND SETS THE REPORT FLAG TO KEEP #
* REPORTS/SEC AS SPECIFIED
* ALL CPU REGISTERS PRESERVED


TIMER:
* LED ICON STUFF START

        BCLR    1,PORTC     ;ICONS OFF
        LDA     LEDNO       ;GET CURRENT LED #
        CMP     #15         ;16TH LED?
        BLO     TIMER1      ;GO IF NOT
TIMER01:
        CLRA                ;START OVER WITH
        STA     LEDNO       ; LED0
TIMER1:
        CMP     CURPOS      ;TEST FOR CURSOR
        BEQ     TIMER5      ;GO SHOW CURSOR
TIMER11:
        CMP     #07         ;IN LSB?
        BHI     TIMER2      ;GO IF IN MSB
        TAX
        LDA     MASK,X      ;GET LED BIT MASK
        AND     ICONLSB     ;CHECK IF LED ON
        BEQ     TIMER4      ;GO FOR LED OFF
        BRA     TIMER3
TIMER2:
        SUB     #08         ;POSITION IN MSB
        TAX
        LDA     MASK,X      ;GET LED BIT MASK
        AND     ICONMSB
        BEQ     TIMER4
TIMER3:
        LDX     LEDNO       ;GET CURRENT LED
        LDA     YADAT,X     ;MAKE LED GREEN
        STA     PORTA       ;SET PORT A
        LDA     YBDAT,X
        STA     PORTB       ;SET PORT B
        BSET    1,PORTC     ;TURN ON LED
TIMER4:
        INC     LEDNO       ;GO TO NEXT LED
        BRA     TIMER9

* CURSOR STUFF
TIMER8:
        LDX     LEDNO
        LDA     RADAT,X     ;MAKE LED RED
        STA     PORTA
        LDA     RBDAT,X
        STA     PORTB
        BSET    1,PORTC     ;TURN ON LED
        INC     LEDNO       ;GO TO NEXT LED
TIMER9:
```

```
* LED ICON STUFF END

        LDA     TSR;DUMMY CHK TIMR STATUS
        LDA     OCLR
        ADD     #125;.25mS INTERRUPT RATE
        TAX
        LDA     OCHR     ;TIMER HIGH BYTE
        ADC     #$00 ;CARRY FROM LOW BYTE
        STA     OCHR     ;STORE HIGH BYTE
        STX     OCLR  ;STORE LOW BYTE AND
                       ;RESET INTERRUPT
        DEC     RPTIMER ;DEC REPORT TIMER
        BNE     TIMER99
        BRCLR   TE,FLAG1,TIMER98
        BSET    TT,FLAG1;TIME TO TRANSMIT

TIMER98:
        LDA     RPTIME
        STA     RPTIMER;RESET REPORT TIMR
TIMER99:
        RTI             ;RETURN FROM INT


* SUBROUTINE TO GENERATE EVEN PARITY
* ON ENTRY, ACCUMULATOR HOLDS 8 BIT VALUE
* ON EXIT, CARRY INDICATES PARITY BIT FOR
* EVEN PARITY
* X ALTERED

PARITY:
        CLRX                    ;CLEAR X
        CLC                     ;CLEAR CARRY
        RORA                    ;BIT0
        BCC     PARITY1
        INCX
PARITY1:
        RORA                    ;BIT1
        BCC     PARITY2
        INCX
PARITY2:
        RORA                    ;BIT2
        BCC     PARITY3
        INCX
PARITY3:
        RORA                    ;BIT3
        BCC     PARITY4
        INCX
PARITY4:
        RORA                    ;BIT4
        BCC     PARITY5
        INCX
PARITY5:
        RORA                    ;BIT5
        BCC     PARITY6
        INCX
PARITY6:
        RORA                    ;BIT6
        BCC     PARITY7
        INCX
PARITY7:
        RORA                    ;BIT7
        BCC     PARITY8
        INCX
PARITY8:
        RORA
        LSRX    ;PARITY BIT IN CARRY
        RTS             ;RETURN FROM SUB


MASK:
        FCB     00000001B   ;BIT MASKS
        FCB     00000010B
        FCB     00000100B
        FCB     00001000B
        FCB     00010000B
        FCB     00100000B
        FCB     01000000B
        FCB     10000000B

* YELLOW/GREEN LED TABLE DATA
*               COL  ROW
*               32103210
YADAT:
        FCB     00011110B           ;LED0
        FCB     00101101B           ;LED1
        FCB     01001011B           ;LED2
        FCB     10000111B           ;LED3
        FCB     00101110B           ;LED4
        FCB     01001101B           ;LED5
        FCB     10001011B           ;LED6
        FCB     00010111B           ;LED7
        FCB     01001110B           ;LED8
        FCB     10001101B           ;LED9
        FCB     00011011B           ;LED10
        FCB     00100111B           ;LED11
        FCB     10001110B           ;LED12
        FCB     00011101B           ;LED13
        FCB     00101011B           ;LED14
        FCB     01000111B           ;LED15

YBDAT:                              ;PORT B MASKS
        FCB     0000B               ;LED0
        FCB     0000B               ;LED1
        FCB     0000B               ;LED2
        FCB     0000B               ;LED3
        FCB     0000B               ;LED4
        FCB     0000B               ;LED5
        FCB     0000B               ;LED6
        FCB     0000B               ;LED7
        FCB     0000B               ;LED8
        FCB     0000B               ;LED9
        FCB     0000B               ;LED10
        FCB     0000B               ;LED11
        FCB     0000B               ;LED12
        FCB     0000B               ;LED13
        FCB     0000B               ;LED14
        FCB     0000B               ;LED15


* RED LED TABLE DATA

RADAT:                              ;PORT A MASKS
        FCB     0001B       ;LED0
        FCB     0010B       ;LED1
        FCB     0100B       ;LED2
        FCB     1000B       ;LED3
        FCB     0001B       ;LED4
        FCB     0010B       ;LED5
        FCB     0100B       ;LED6
        FCB     1000B       ;LED7
        FCB     0001B       ;LED8
        FCB     0010B       ;LED9
        FCB     0100B       ;LED10
        FCB     1000B       ;LED11
        FCB     0001B       ;LED12
        FCB     0010B       ;LED13
        FCB     0100B       ;LED14
        FCB     1000B       ;LED15

RBDAT:                              ;PORT B MASKS
        FCB     1000B               ;LED0
        FCB     0100B               ;LED1
        FCB     0010B               ;LED2
        FCB     0001B               ;LED3
        FCB     0100B               ;LED4
        FCB     0010B               ;LED5
        FCB     0001B               ;LED6
```

```
        FCB     1000B           ;LED7
        FCB     0010B           ;LED8
        FCB     0001B           ;LED9
        FCB     1000B           ;LED10
        FCB     0100B           ;LED11
        FCB     0001B           ;LED12
        FCB     1000B           ;LED13
        FCB     0100B           ;LED14
        FCB     0010B           ;LED15


* 68HC05 VECTOR TABLE

        ORG     $1FF4
        FDB     START
        FDB     SCIISR
        FDB     TIMER
        FDB     START
        FDB     START
        FDB     START

        END     START
```

VITA

Paul Beam was born in Salem, Ohio in May 6, 1966. The son of a minister, his childhood was spent in 4 different states before settling in Tennessee. After graduating from Doyle High School, he enrolled in the University of Tennessee in the Fall of 1984.

While at the University, he served as President of the Christian Student Fellowship as well as being a member of the IEEE and Tau Beta Pi. He graduated with a bachelor's degree in Electrical Engineering in the Spring of 1988.

In the Fall of 1989, he enrolled as a graduate student of the University of Tennessee. He was the electrical team leader for the national championship Methanol Marathon effort launched in conjunction with the mechanical engineering department, and sponsored by DOE and General Motors.

In the Summer of 1990, he began to work with Empruve, Inc. in Knoxville, where he was part of a small team that developed a personal multimedia machine.

Currently, he works for Johnson Bible College as Computer Systems Engineer and as a consultant to Product Development Associates, Inc.