STYLE OF USER DOCOMENTATION FOR MICRO COMPUTERS

BY

ROLF LEE COOK
BBA UNIVERSITY OF MIAMI 1973

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

A MASTER'S REPORT

SUBMITTED IN PARTIAL FULLFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

KANSAS STATE UNIVERSITY
MANHATTAN, KANSAS

- 1983 -

APPROVED BY:

MAJOR PROFESSOR

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

INTRODUCTION

1.1 SCOPE

This report develops a style for user documentation. Using current standards for good technical documentation and many of the attributes of a formal model, this style is presented as an example of desirable structure for user documentation. Its specific focus is a description of how an Atari 850 Interface Module works in an inter-computer communications mode. The following disclaimer must be stated: manufacturer-provided documentation was used as the source of all technical information. The current Atari manual concerning inter-computer communications instructions, the Atari 850 Interface Module Users Manual, was found to be inadequate when applied to a non-trival inter-computer communications requirement due to ambiguity and disorganization. Two examples of this are provided below. Therefore information in the Atari manual has been revised and presented in a more organized manner. This revision is not intended to represent a comprehensive explanation of the full capabilities of the Atari Model 850 Interface Module.

It is focused on a specific function to demonstrate the applicability of style to user documentation.

1.2 PURPOSE

This report presents an example of good user documentation combining the techniques of Software Engineering and Technical Writing.

1.3 BACKGROUND: PROBLEMS OF TECHNICAL DOCUMENTATION

Recently a number of factors have greatly improved the attractiveness of micro computers for both the home enthusiast and the small business owner. These factors include:

1. Significant reductions in price.
2. Increases in overall capability.
3. More prewritten software.
4. More "user friendly" software and operating systems.

In combination these factors result in more and more naive users buying fairly powerful micro computers. The level of sophistication and capability of these personal micro computer systems is approaching that of the "large" main frame computers of just a few years ago.

Herein lies the genesis of a significant problem. In their haste to get the newest and best hardware and software in the market place, manufacturers and vendors give cursory

attention to user documentation. They frequently fail to give the same level of development effort to documentation that is given to designing the hardware or developing the software.

The environment of micro computer documentation may be viewed as a circle. The circumference of the circle is that portion of the user documentation most frequently encountered by the user. That outer portion represents such things as the introductory guidance for setting up the machine, the instructions for using common devices, and the primary language manual used with the system. This crust is firm as the documentation tends to be simple, unambiguous and correct. However when one moves beyond these simplistic areas and attempts to take advantage of the inherant capabilities of the equipment, all too frequently the documentation is disorganized, ambiguous and incorrect. Certainly this is unintentional and results from the fierce competition for resources in a highly competitive industry.

1.4 GOOD TECHNICAL DOCUMENTATION

The Software Engineering discipline is beginning to develop precision, reliability and cost effectiveness in engineered software products. Unfortunately, a review of current articles on Software Engineering reveals a distressing failure to project Software Engineering principles into the

final software domain: user documentation. It is evident that an excellent system, built using the finest Software Engineering tools, is doomed to failure if the user cannot use it because the instruction manual is unintelligable. Similarly a mediocre system that is presented to the user with instructions that are simple, well organized, and correct can be a great success. These obvious truths will become increasingly clear as naive users become a force in the market place.

What is good technical documentation? The answer to this question is highly subjective, with little agreement among Technical Writing authors. Inevitably the answers are presented in terms that are themselves subjective, such as "simple", "well organized", "clear", "complete", and "correct". Howard presents three principles of Information Design <HOWA-81> which seem to encompass the concepts proposed by numerous other Technical Writing authors <TEBE-80>, <FULL-80>, and <BAKE-61>. The three concepts are:

1. Content - defines the breadth and depth of the material in a document.
2. Organization - gives shape and direction.
3. Format - makes the information understandable through language and illustration.

Most authors add a fourth principle, expressed in a variety of terms, which alludes to accuracy or correctness of the information presented.

Technical Writing principles are not far removed from the goals of Software Engineering <ROSS-75>. The verbal descriptions of the Software Engineering goals, modifiability, efficiency, reliability, and understandability, can be reorientated from programs and systems to documentation without changing intent. When modified, the similarity of these goals to the principles of technical writing suggest that Software Engineering concepts are applicable to user documentation. Structured Analysis <ROSS-77> fits with the ideas of organization and format by suggesting procedures that lead to objectivity.

The combination of ideas from Software Engineering and Technical Writing yields six attributes that can be objectively measured. As these ideas represent an amalgamation of ideas from numerous authors, significant literary license has been taken in developing the titles and definitive descriptions.

1. Correctness:
   a) The document contains no conflicting instructions.
   b) The information presented can be tested and produce consistent results.
2. Correlation: Illustrations are supported in the text by reference to objects or keys.
3. Structure: The organization of the document should be hierarchical from the general to the specific, providing direction and bounds.

4. Models:

   a) Physical models should be developed in relevant illustrations and text.

   b) Logical models should be developed in illustrations and text to reflect finite states and transitions.

5. Procedures and Specifications: Each user-initiated procedure should be explained in terms of the parameters and results as they affect the transition of state.

6. Vocabulary standardization:

   a) Technical terms, particularly those for which there is no universally accepted usage, should be defined prior to use.

   b) Terms and names should be used consistently.

The attributes above tend to fall into two categories. Procedures and Specifications, Models, and Structure are generally global and measured in terms of their presence or absence. Correctness, Correlation, and Vocabulary Standardization are measured by countin each instance and comparing success against failure.

Having used the objective attributes, above, the author, reviewer, or reader must determine the threshold of "good" and "bad". A few errors or violations do not necessarily condemn the document. Conversely, a document that has all

the "good" attributes can still be "bad" based on other subjective factors.

The Atari 850 Interface Module Users Manual <ATAR-80> fails to meet most of the objective measures listed above. An example of violating correctness is demonstrated in the power-up or bootstrap sequence instructions. Under the section entitled POWER-UP on page 12 the user reads, "When the Interface Module and a Disk Drive are used in your system, they will both have to boot up before they can be used, so they both have to be powered on before you power on the console.". However, just three pages further the user reads in Table 2.1 that the correct power-up sequence is, "Turn on 810 <Disk Unit>, then console, then Atari 850 <Interface Module>" (comments added between < > by author). On page 83 under the title "Power-on Bootstrapping Operation" the user is told, "The bootstrapping operation is enabled by turning ON the power to the Interface Module before the 400 or 800 computer.". Although a scheme for presentation is given in the introduction, organization of information is disjoint throughout the manual. Detailed discussion of almost every subject takes place in more than one location. For example the Configure BAUD Rate command, the variable values allowed in it, and the impact of those values are discussed in three different locations. The reader must be mystic to understand the title of the

discussions and their relevance to the Configure BAUD Rate command. The manual uses very few illustrations and those are not closely correlated to the nearby text. Very few terms are defined even though their meaning, as used, is not standardly accepted. Further, numerous inconsistencies of use exist.

1.5 SUMMARY

While hardware is getting more complex, the number of naive users is increasing. At the same time the "goodness" of user documentation is not keeping up with the growth of the computer industry. The developing Software Engineering field has principles and tools that are applicable to user documentation and should improve it.

Chapter II

THE ATARI INTER COMPUTER COMMUNICATION MODEL

2.1 INTRODUCTION

The Atari micro computer is capable of inter-computer communication in two sequential modes; through an accoustical connection and through a direct wired connection (RS-232-C compatible). This chapter addresses only inter-computer communications in the sequential direct wired mode and is organized into two sections, one for the definition of terms to be used and the other for a presentation of the inter-computer communications model.

2.2 DEFINITIONS

The definitions of terms presented in this section are a synthesis of several sources or are unique to the Atari manuals. Each definition will follow the procedure of <BARN-74>, that is, DEFINITION = ITEM, CLASS, DIFFERENTIATION.

1. BAUD is the rate of data transmission measured in bits per second. 300 baud is approximately 30 eight bit characters per second.

2. Circuit is the conductor or system of conductors through which an electrical current is intended to flow.
3. Channel is that part of a communications system that connects the message source with the message destination.
4. Interface is the process of causing two or more computing devices (computers or peripherals) to work together.
5. Inter-computer communication is the process of electronicly transfering data between two or more computers.
6. Parallel Input/Output is the process of transfering one or more eight bit characters at a time.
7. Port is the electrical plug receptacle and the associated programs and data structures that facilitate the creation, translation, and reception of electrical signals representing data.
8. Protocol is the predefined circuits, signals, and procedures required to interface computers.
9. RS-232-C is a technical standard of the Electronics Industries Association entitled "Interface Between Data Terminal Equipment and Data Communications Equipment Employing Serial Binary Data Interchange". The standard specifies electrical signal

characteristics and circuit names and defines the function of the signals and control circuits which make up the standard interface.

10. Sequential Input/Output is the process of transfering data one bit at a time usually on a single circuit between two devices.
11. Word is the grouping of 8 bits, normally used to represent a single character. Each character in the Atari character set is represented in a word.
12. Short Word is the use of word lengths of less than 8 bits. The full ASCII character set can be represented in 7 bits; therefore, many manufacturers choose to use short woords for data transmission.

2.3 MODELS

Operations with microcomputers require an understanding of the transitions between finite states. User understanding of these states and transitions is gained by integration of physical and logical models. The fully developed integration of these models leads the user to a level of knowledge necessary to apply state transition procedures meeting his needs.

2.3.1 PHYSICAL MODELS

In its normal configuration for use the Atari system consists of the console, disk unit, interface module, and printer. These devices are physically linked by wire cables as shown in Figure 1. Operating instructions for the console, disk unit, and printer are contained in applicable Atari instruction manuals. Those instructions will not be repeated here unless they are specifically applicable to inter-computer communications.

2.3.1.1 SERIAL I/O PORT

The serial I/O port at KEY 2 in Figure 1 supports all input and output operations between the Atari 800 computer and its peripheral devices (except the TV monitor). It is a 13 pin electrical plug receptacle which has the circuits shown in Figure 2. For a detailed explanation of the use of these circuits and the electrical standards, refer to Chapter III of the Atari Hardware Manual and Chapter 9 of the Atari Operating System Users Manual.

2.3.1.2 PARALLEL PRINTER PORT

Data for the printer is sent to the interface module, buffered, then sent to the printer using the 15-pin electrical connector plug receptacle, at KEY 3 in Figure 1, in a parallel data format. Figure 3 shows the circuits

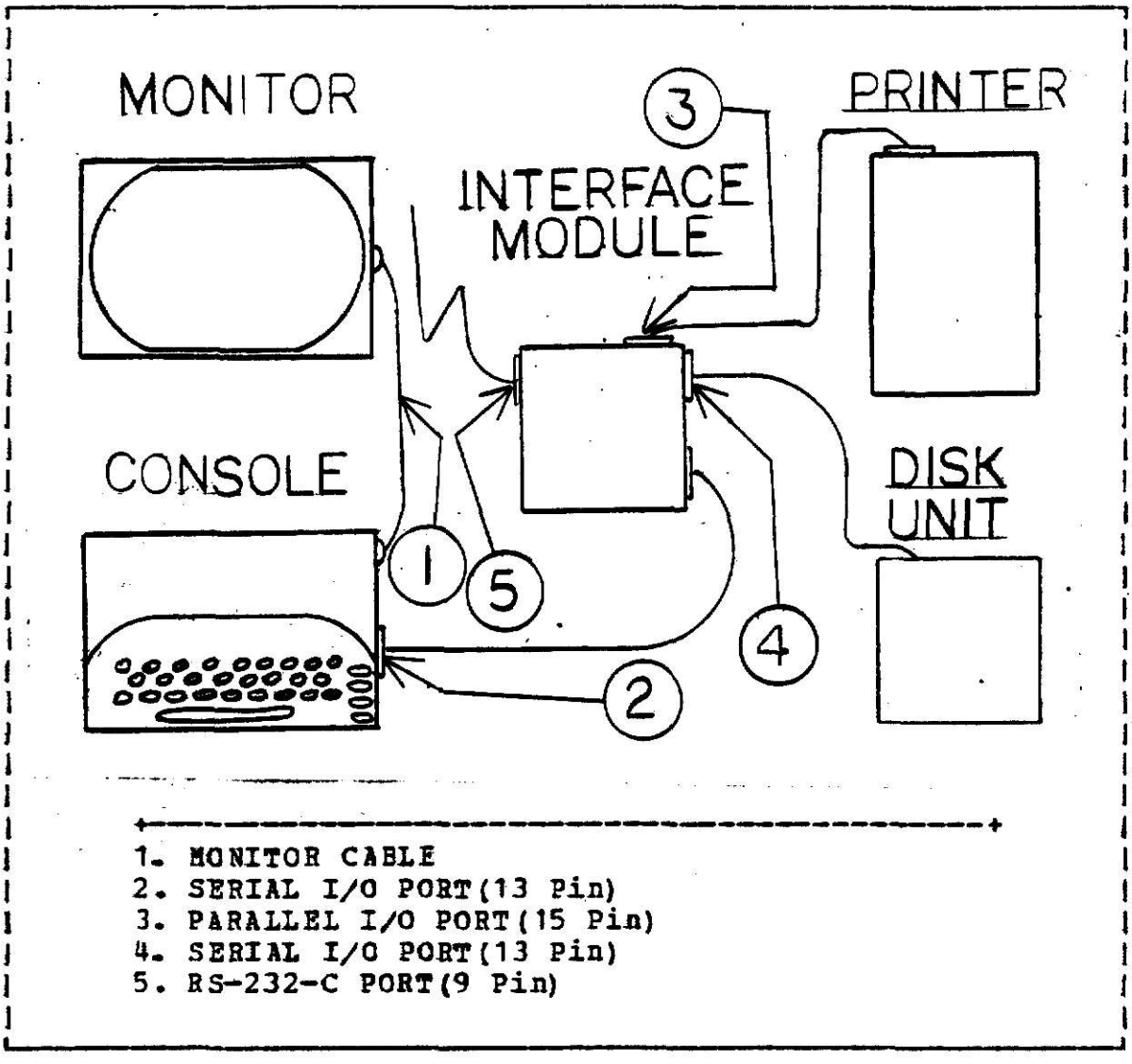

Figure 1: ATARI INTER COMPUTER COMMUNICATIONS CONFIGURATION

used. For a full explanation of the electrical standards refer to Appendix 11 of the Atari Interface Module Users Manual.

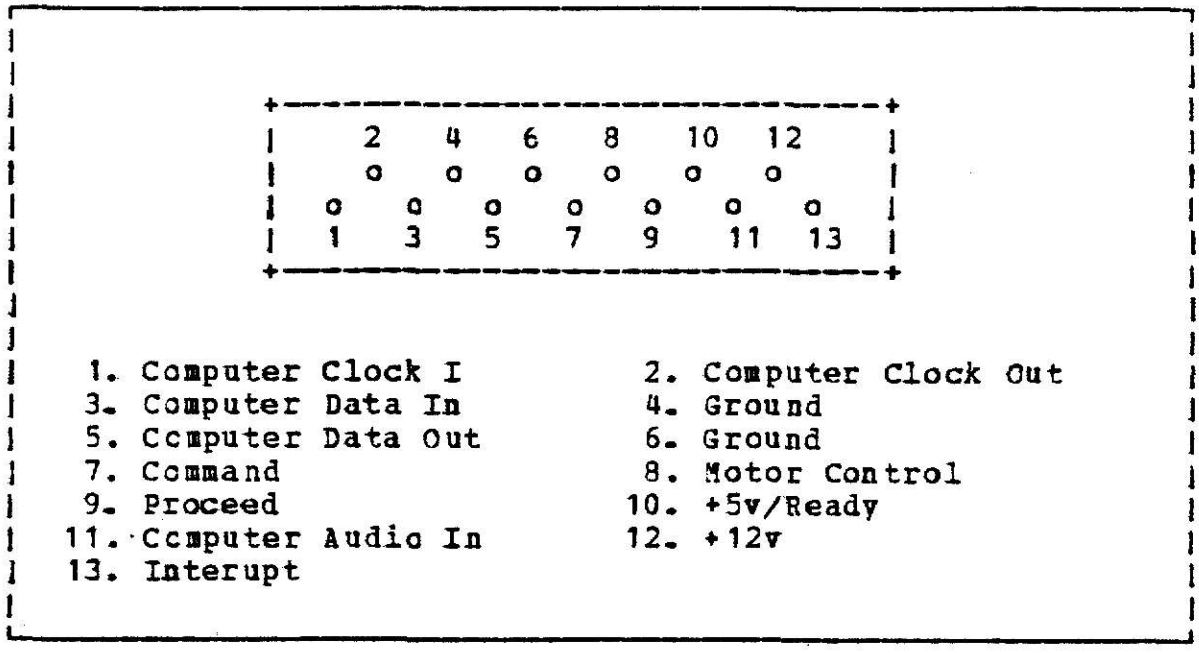

Figure 2: SERIAL I/O PORT

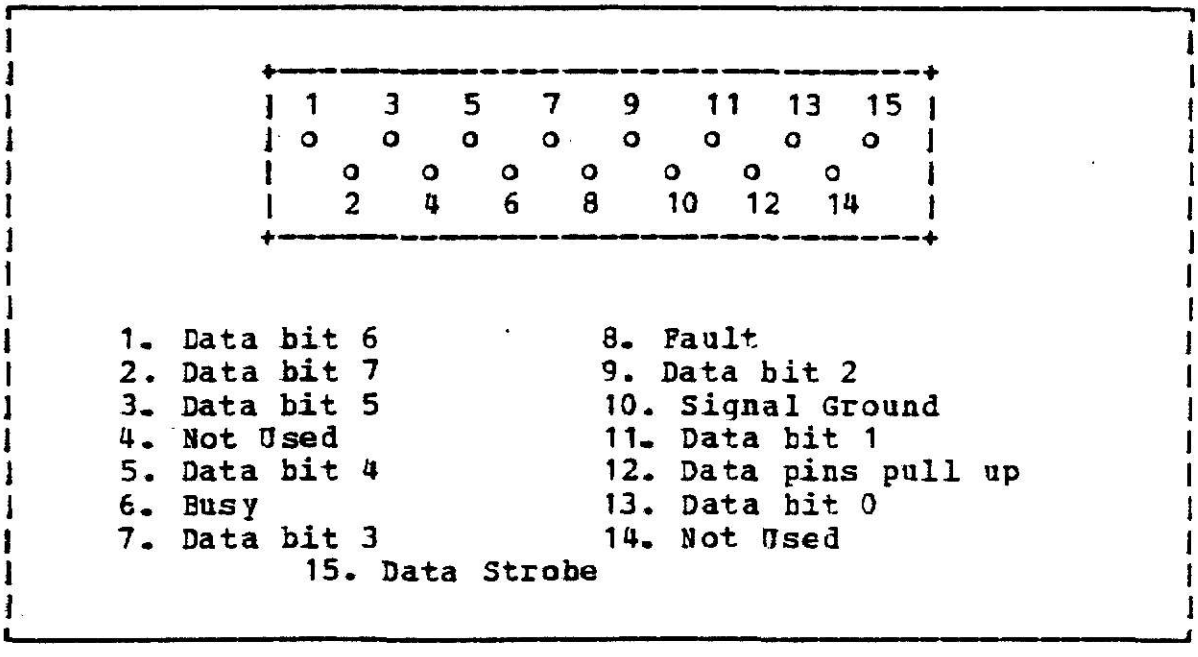

Figure 3: PARALLEL I/O PORT

2.3.1.3 RS-232-C COMPATABLE PORTS

The RS-232-C compatible port at KEY 5 in Figure 1 above is a 9 pin electrical plug receptacle which is designed to meet the essential electrical and circuit requirements of the standard, allowing communication with other computers that also meet the standard. Three different circuit protocols are available and shown in Figure 4. For a detailed explanation of the electrical standards and circuit directional requirements see Appendix 1 of the Atari Interface Module Users Manule.

Although port 4 is RS-232-C compatible it will also interface with devices that require a 20mA loop. Figure 5 shows this capability. A full explanation of connecting this port with other devices is found in Appendix 11 of the Atari Interface Module Users Manual.

2.3.1.4 BOOTSTRAPPING

There is more than one computer in Figure 1 above. The Atari 850 Interface Module as well as the Atari disk unit are themselves computers. Each contains a micro processor, a program in ROM, and Input/Output capabilities. Further, each contains information required for inter-computer communications . This information is transfered to the Atari 800 Computer. When powered up, the computer polls the system to determine which of the peripherals are available. Since

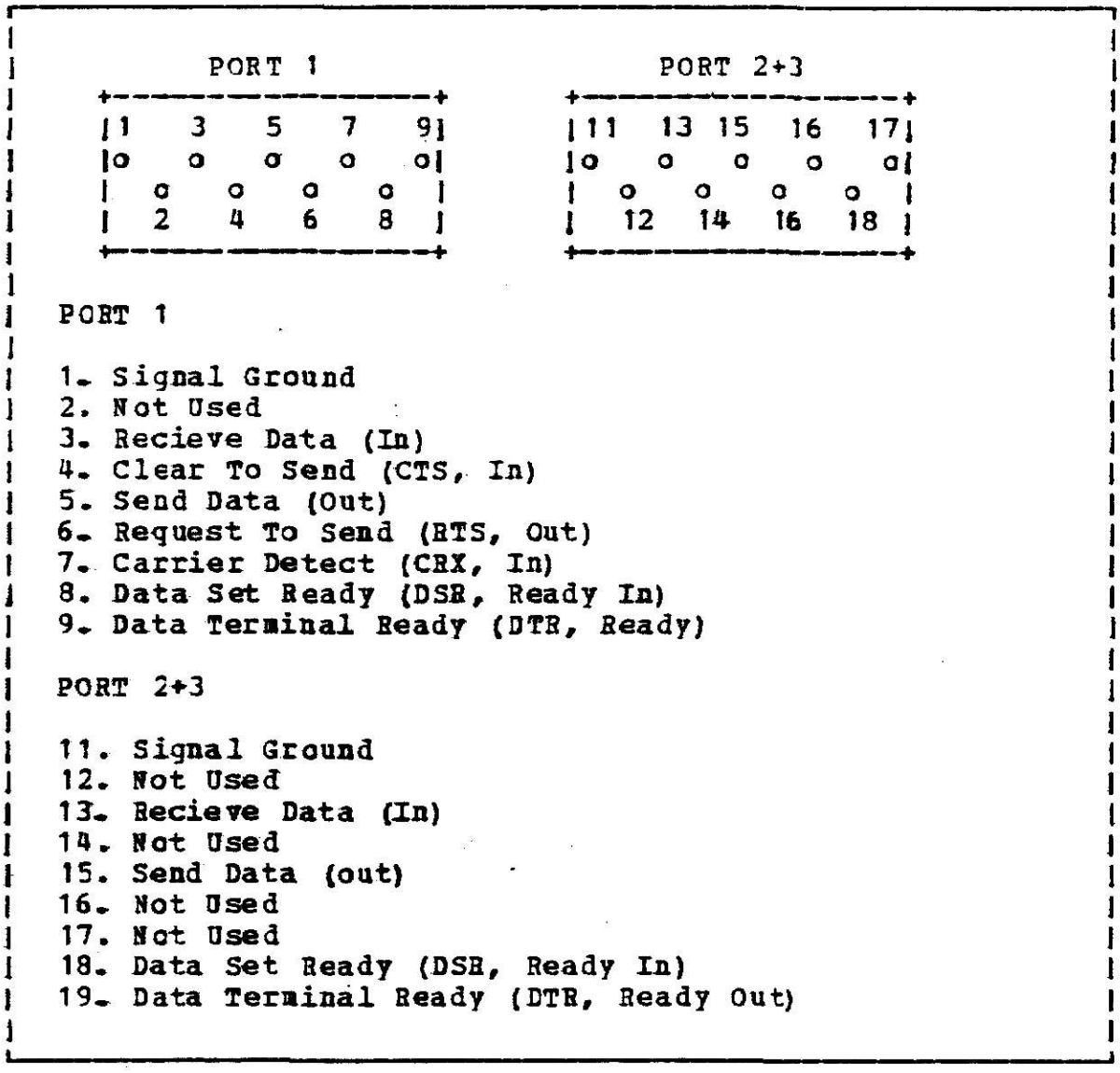

Figure 4: RS-232-C PORTS

the peripherals contain information to allow general operation in a variety of configurations, it is important to bootstrap in the correct sequence.

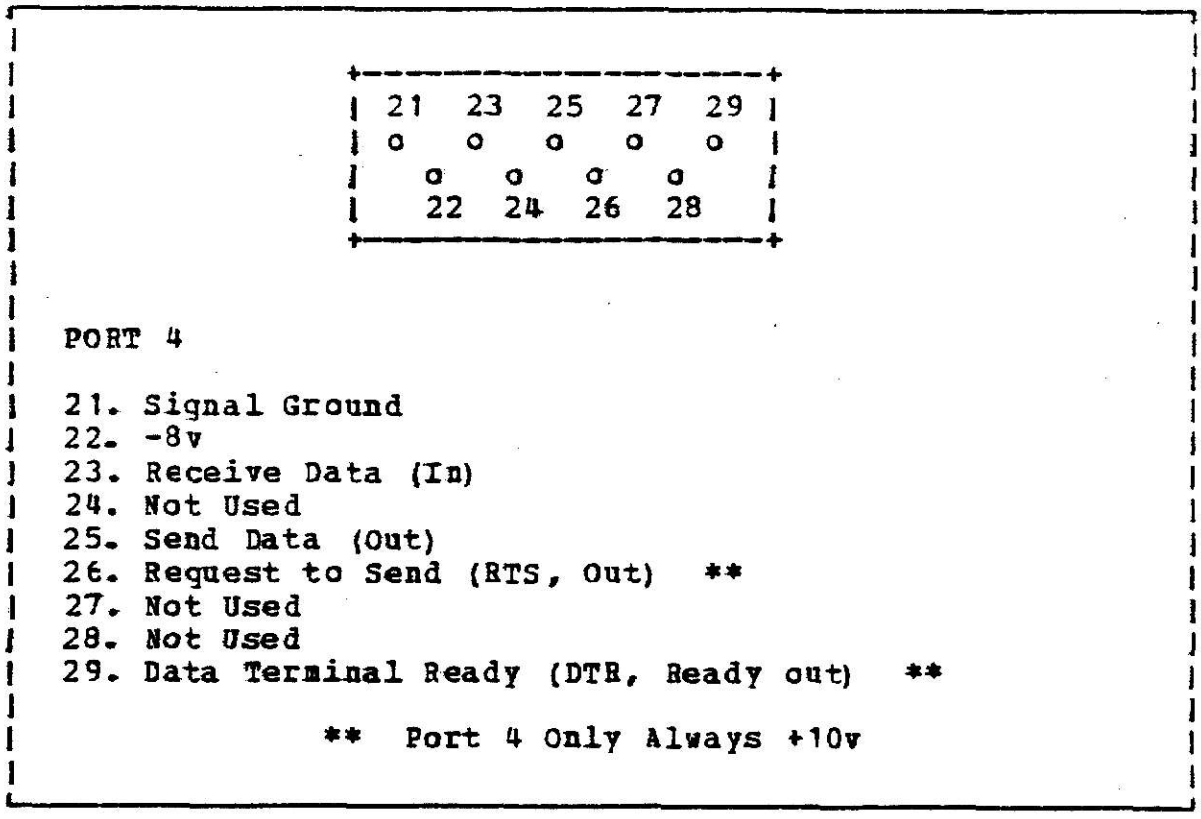

Figure 5: RS-232-C PORTS

It is generally correct to apply power to any peripheral prior to applying power to the console. However, if the Disk Operating System (DOS) is DOS I, the disk unit may not be used while the interface module is being used for inter-computer communications . Therefore the disk unit must not be powered up before the console. This means the disk unit is not bootstrapped and cannot be addressed by the computer. With DOS I, rebootstrapping is required between use of the disk unit and inter-computer communications . If

the DOS is DOS II then the disk unit may be used during inter-computer communications and there is no requirement to rebootstrap, either before or after an inter-computer communication session. However the program AUTORUN.SYS must be resident on the disk used during bootstrap. AUTORUN.SYS contains operating instructions that are used to control inter-computer communication when the disk unit is in the system. Table 1 shows the correct bootstrap sequence for the system.

TABLE 1

BOOTSTRAP SEQUENCE

DEVICE	SEQUENCE	
	DOS I	DOS II
Disk Unit	NOT ALLOWED	1
Control Module	1	2
Console	2	3
Printer	NO EFFECT	NO EFFECT

When the console is powered up predefined RAM allocations exist. Figure 6 shows these allocations.

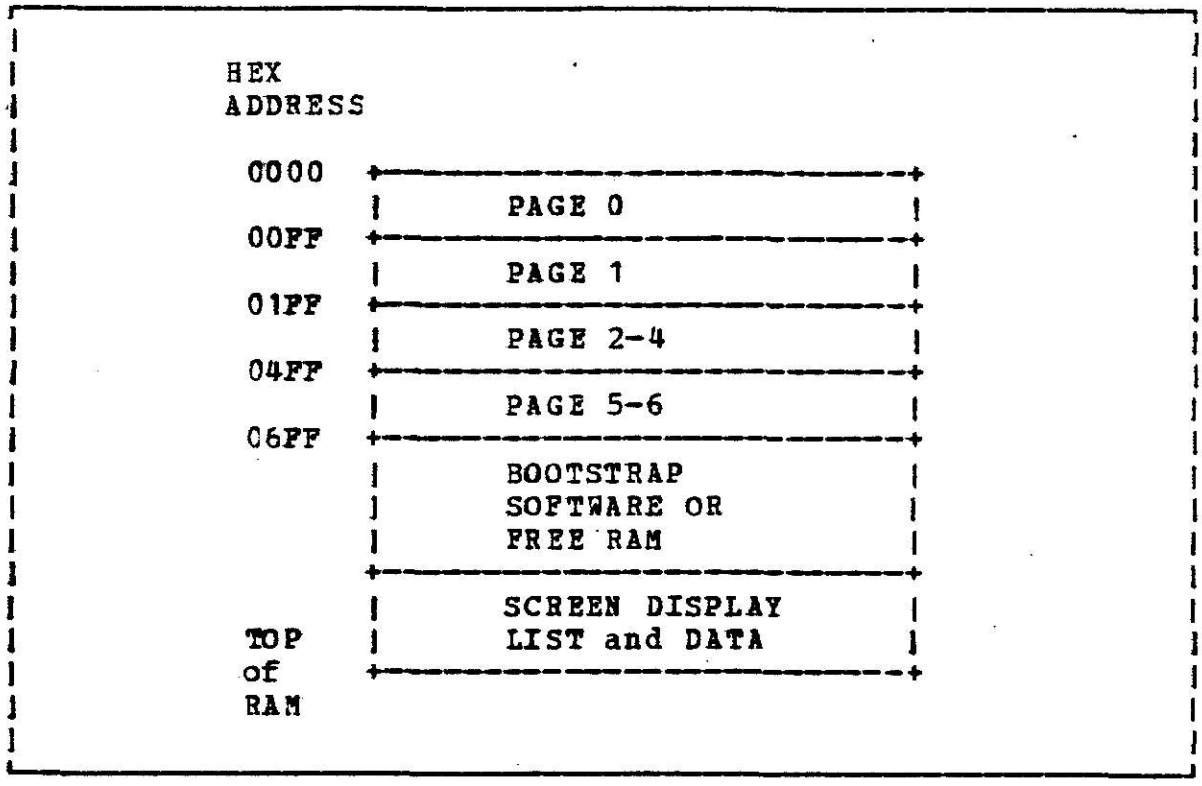

Figure 6: MEMORY MAP

Page 0 RAM is used by the OS, Device Handlers, and the Floating Point Package.

Page 1 RAM is used as the hardware stack region and for interupt processing.

Pages 2-4 are used for OS database variables, OS working tables, and data buffers.

Pages 5-6 are available for user workspace except location 057E through 05FF which are used by the Floating Point Package.

Although Pages 0 through 4 contain locations that are, at times, available for user work areas, extreme caution should be exercised before these areas are used. Users should be thoroughly familiar with the contents of the Atari Operating System Users Manual prior to attempting use of any location in Page 0 to 4 as a work area.

When the Console is powered on, the Operating System (OS) resident in ROM causes a sequence of actions to be performed. A partial list of these actions is shown in Table 2. The last entry in Table 2 is the most important to inter-computer communication. This step in the bootstrap sequence begins the actions that will prepare the system to accomplish inter-computer communication.

TABLE 2

OS POWER UP SEQUENCE

DETERMINE THE HIGHEST RAM ADDRESS
CLEAR ALL RAM TO ZEROS
ESTABLISH ALL RAM INTERRUPT VECTORS
FORMAT THE DEVICE TABLES
INITIALIZE CARTRIDGE(S)
SET UP THE SCREEN
BOOT THE CASSETTE IF PRESENT
CHECK CARTRIDGE SLOTS FOR DISK BOOT INSTRUCTIONS
BOOT THE DISK IF PRESENT
TRANSFER CONTROL TO THE CARTRIDGE, DISK BOOTED PROGRAM, CASSETTE BOOTED PROGRAM, OR BLACKBOARD PROGRAM

When the disk unit is present and bootstraped, OS transfers control to the DOS II area which in turn calls program AUTORUN.SYS. AUTORUN.SYS establishes the fixed and default values for the RS-232-C port device tables then

calls the interface module to load and initialize the four port device handlers. The default values for the port handlers and device tables are then returned to the interface module. This series of actions is shown in Figure 7.

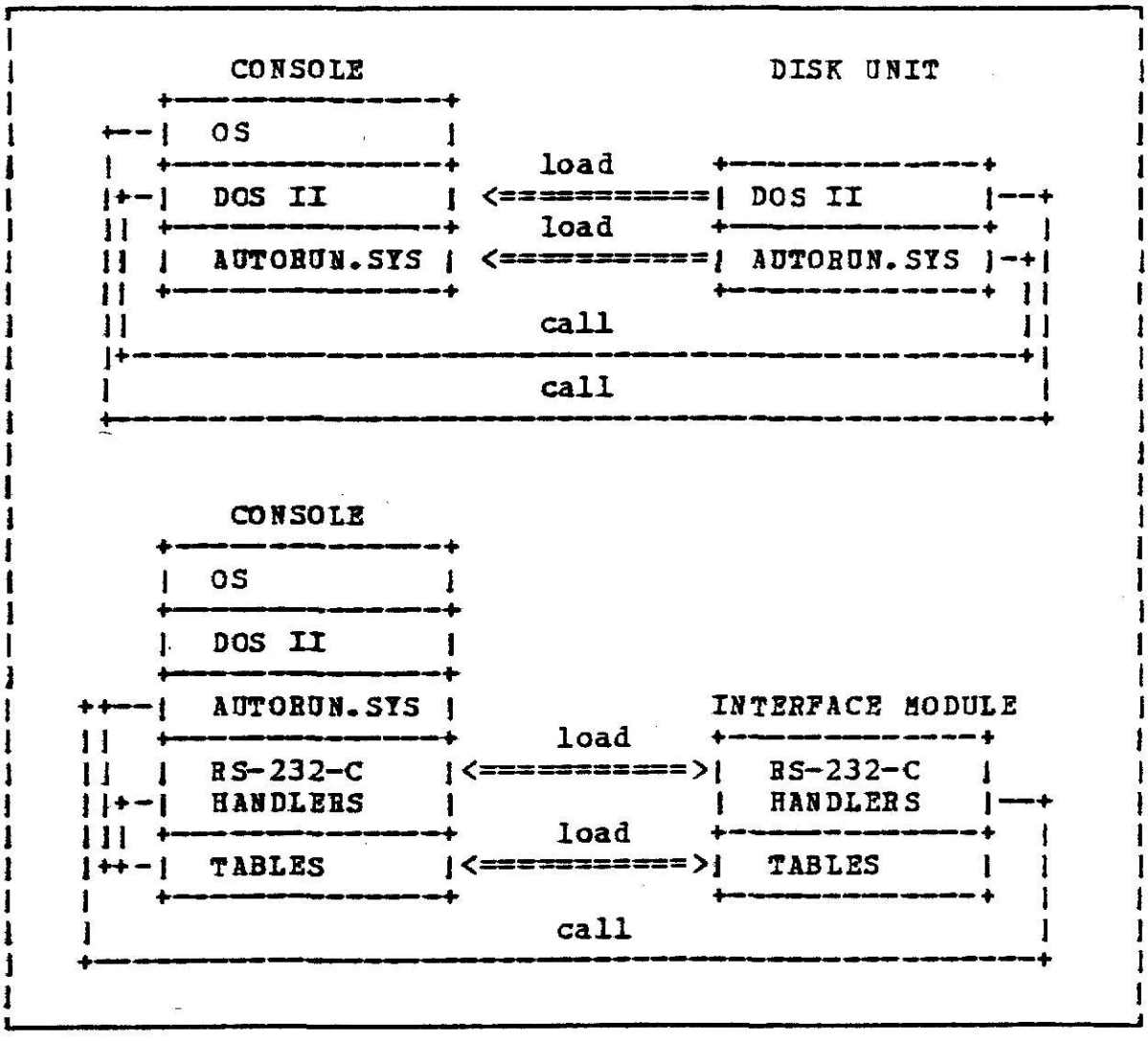

Figure 7: PORT INITIALIZATION WITH DOS II

If the disk unit is not present the OS calls the interface module to load and initialize the port handlers and device tables. This series of actions is depicted in Figure 8.

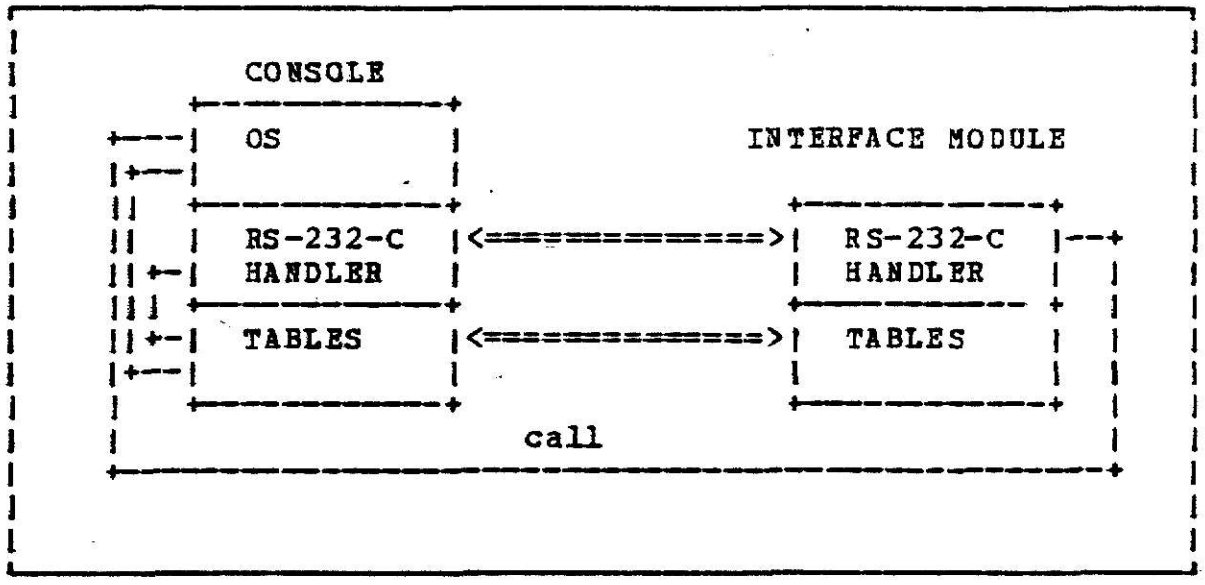

Figure 8: PORT INITIALIZATION WITHOUT DISK UNIT

Figure 9 shows the internal communications processes and data structures as they exist after the computer has been powered up and bootstrap completed.

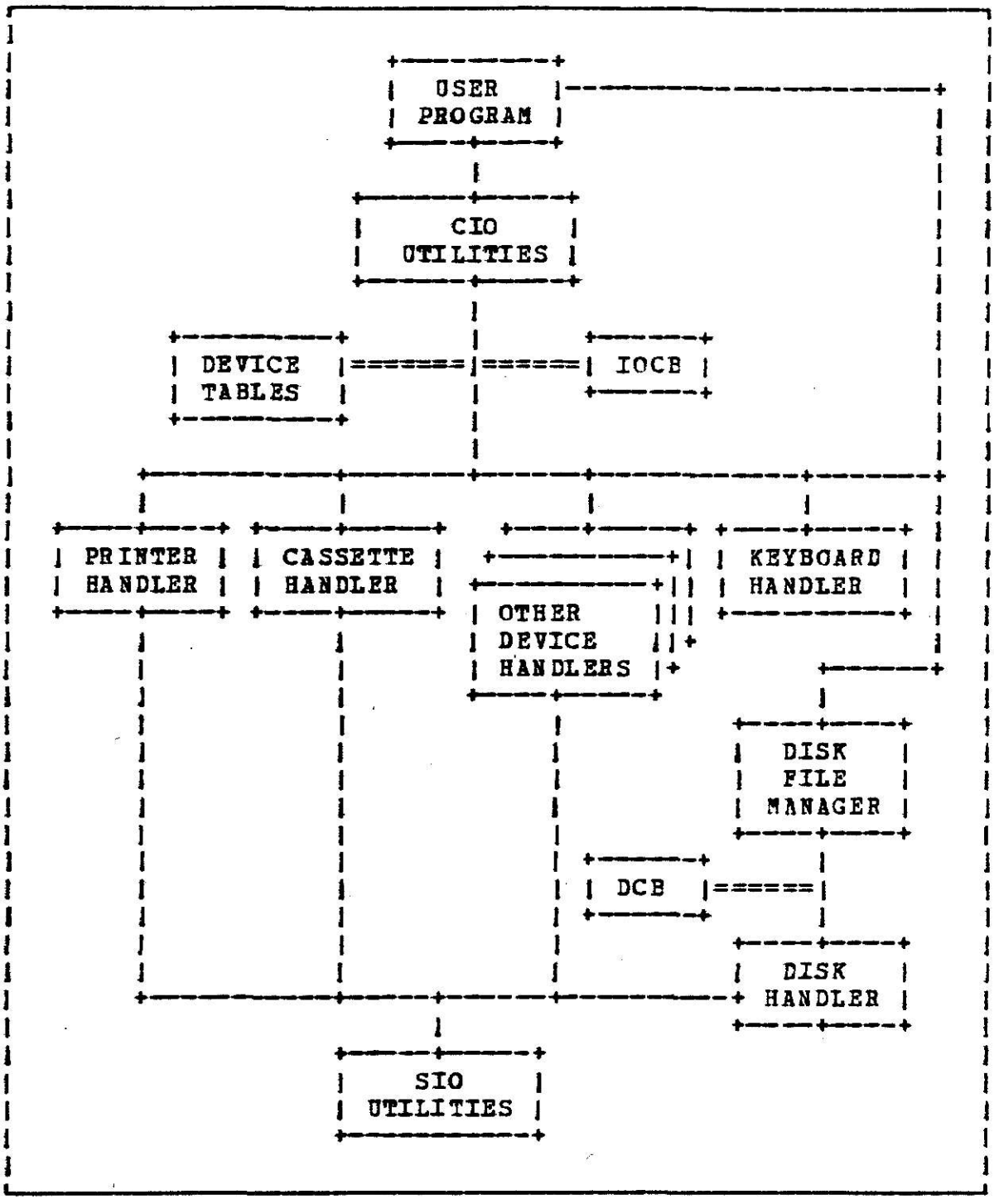

Figure 9: INITIAL STATE

All user programs communicate with the peripheral devices through the CIO utilities that are part of the OS. The CIO utilities use the Input/Output Control Block (IOCB) and device tables to provide the necessary device-unique parameters to the device handlers which supervise the Sequential Input/Output (SIO) utilities in performing the actual transfer of data and commands. Most peripheral devices have internal handlers and tables to accept or return the data as required. This includes the four RS-232-C compatible ports in the interface module.

A device table is initialized for each peripheral device. Each table contains the device name and the address of the unique handler vector table created for the device during bootstrap.

The handler vector table contains the addresses for primitive utilities oriented to each device that perform I/O functions.

Eight IOCBs are formated during bootstrap. One is assigned to each device as it is OPENed. For example IOCB 0 is OPENed and assigned to the Screen Editor by OS during bootstrap. IOCB is a 16 byte block that contains the fields shown in Table 3.

A final view of the physical model of the Atari system as it is bootstrapped and ready for the user to start the process of configuring for inter-computer communications is in Figure 10.

TABLE 3

The INPUT/OUTPUT CONTROL BLOCK

NOUN NAME	DATABASE VARIABLE NAME	HEX ADDRESS	DESCRIPTION
Handler I.D.	ICHID	0340	Index to Device Table. Set by CIO on OPEN command, reset to null on CLOSE.
Device Number	ICDNO	0341	Set by CIO on OPEN command to distinguish between multiple devices of same type.
Command Byte	ICCMD	0342	This byte is set by the I/O command to be executed i.e. PUT, GET, INPUT, PRINT,and STATUS.
Status	ICSTA	0343	This byte holds the status after each CIO action. The high order bit is 1 on error with the remaining bits holding the error condition code.
Buffer Address (low)	ICBAL	0344	Initially set to the 32 byte system buffer. Can be modified by user.
Buffer Address (high)	ICBAH	0345	
Put Address (low)	ICPTL	0346	Initially set to "IOCB Not OPEN" routine in CIO. Used to support Basic on OPEN.
Put Address (high)	ICPTH	0347	

TABLE 3

CONTINUED

Buffer Length	ICBLL	0348	Set to 32 on boot.
Byte Count	ICBLH	0349	Set to 0 on boot. Incremented by CIO on I/O commands.
Auxiliary Information	ICAX1	034A	Set on OPEN command. Specifies direction of I/O.
	ICAX2	034B	Contains device specific information.
	ICAX3 : ICAX6	034C : 034F	Area used by device handlers in processing I/O.

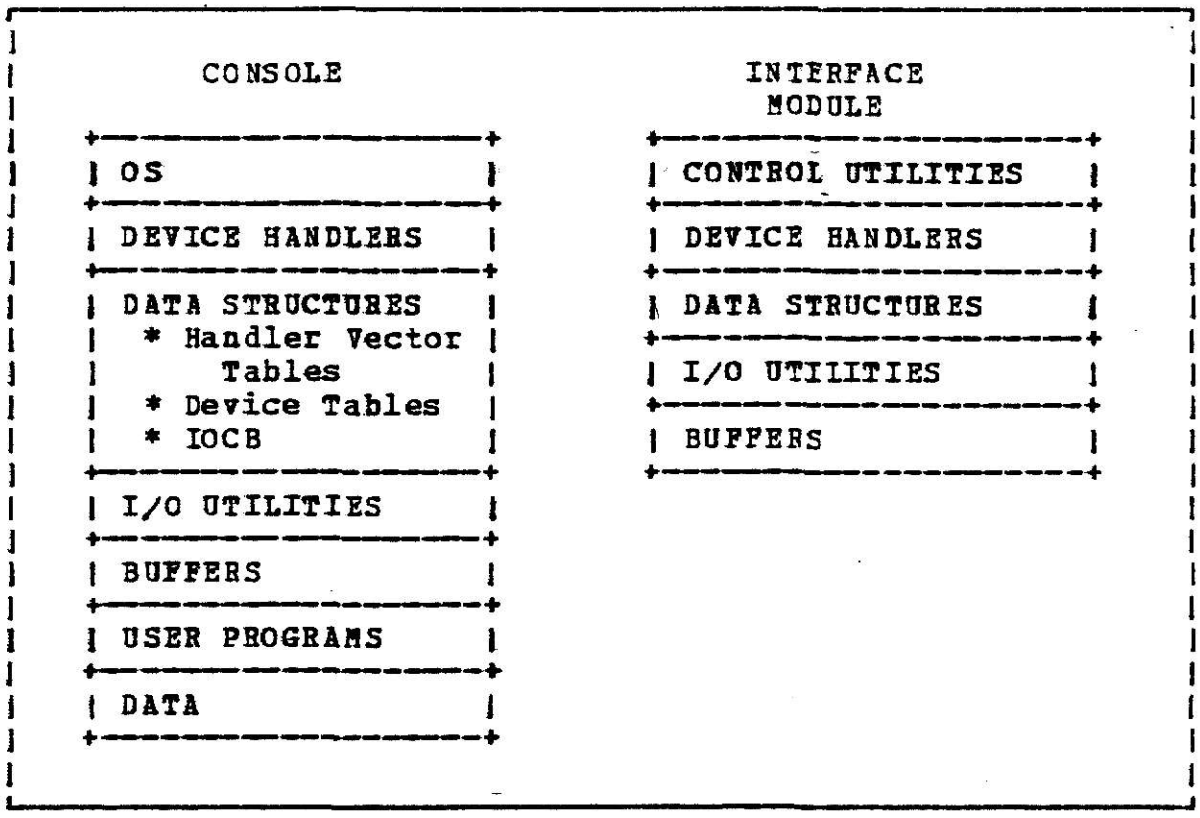

Figure 10: SYSTEM VIEW

2.3.2 THE LOGICAL MODEL

Preparing the Atari computer to interact with another computer is an extremely complex task. The easiest way to explain these preparations is to logically develop from the physical model of the system toward a final state which will allow inter-computer communication. Computers are finite state machines in that they work in discrete steps transitioning from one state to the next based on the current state and the next input. The key to success is

controlling the sequence of state changes to accomplish the desired task. Therefore the logical model must show how to control these state changes.

2.3.3 Initial RS-232-C Port State

During bootstrap the RS-232-C port handlers are initialized and returned to the interface module along with a set of external device handling tables. The initial state of these tables is shown in Table 4.

TABLE 4

INITIAL STATE of VARIABLES

300 BAUD
8 BITS PER WORD
1 STOP BIT PER WORD
INPUT PARITY NOT CHECKED
OUTPUT PARITY BIT SET TO ZERO
LINE FEED OFF
LIGHT TRANSLATION
DTR AND RTS NOT SET

2.3.3.1 PORT CONFIGURATION COMMANDS

Figure 11 contains an abbreviated reference for the configuration commands. Detailed information is contained in the sections that pertain to each command.

2.3.3.2 PORT FUNCTION SEQUENCE

The sequence of functions shown in Figure 12 is fixed and cannot be directly changed by the user. Yet the sequence of these functions significantly influences how the user must approach control of input and output for inter-computer communications.

2.3.3.3 ATARI BASIC COMMANDS

STATE	COMMAND					
	XIO 32	XIO 34	XIO 36	XIO 38	OPEN	XIO 40
Force S/B	CMD					
Set DTR		AUX1				
Set RTS		AUX1				
Set XMT		AUX1				
BAUD			AUX1			
Word Size			AUX1			
In Circuit			AUX2			
Translatien				AUX1		
In Parity				AUX1		
Out Parity				AUX1		
Append LF				AUX1		
Won't trans				AUX2		
I/O Mode					CMD	
Start Con I/O						CMD

Figure 11: PORT CONFIGURATION COMMANDS

```
OUTPUT
  1. TRANSLATION
  2. PARITY
  3. TRUNCATION

INPUT
  1. EXPANSION
  2. PARITY
  3. TRANSLATION
```

Figure 12: PORT FUNCTION SEQUENCE

Figure 13 shows the syntax of the Atari Basic commands that configure the RS-232-C ports.

Assertion

```
COMMAND NAME, IOCB#, AUX1, AUX2, "DEVICE and NUMBER:"

EXAMPLE:  XIO 36, #1, 136, 25, "R1:"
          OPEN, #1, 13, 0, "R1:"
```

Figure 13: CONFIGURATION COMMAND SYNTAX

1. COMMAND NAME = XIO plus (32, 34, 36, 38, or 40) or OPEN.
2. IOCB# = # plus 0 through 7.
3. AUX1 = sum of parameter values from appropriate Figures or 0.
4. AUX2 = sum of parameters from appropriate Figures or 0
5. Device = "R (for RS-232-C ports).
6. Number = 1 through 4 followed by :".

Specifications

The XIO commands and the OPEN command are used to convey one or more parameters used by the system to transition from the initial states for inter-computer communications. The IOCB number will be used by CIO to assign values to the IOCB and handler vector table. The AUX1 and AUX2 values are used to set internal tables or are passed to the interface module

for setting its tables. The device number must be one of the four RS-232-C ports in the interface module. Several of the available IOCBs may already be assigned during the bootstrap process. For example IOCB #0 is assigned to the screen editor by OS. The Atari users manual for each peripheral will tell which IOCB is used for that device (some manuals call the IOCB a Channel). The AUX1 and AUX2 values are the sum of one or more values selected from a menu applicable to each command. The individual selections may be added by the user or Basic will add the values, example; ,136, or,100+30+6,. Care must be taken to establish and continue the relation between the IOCB and the device. That is, if IOCB #3 is used for port "R4:" then all XIO commands and the OPEN command must continue this relation. The IOCB to device and device to IOCB relation is singular.

2.3.3.4 XIO 36 BAUD Rate, Word Size, Stop Bits, Input Circuits

Syntax

```
XIO 36, IOCB#, AUX1, AUX2, "R:"
```

Assertions

1. XIO 36 must be used before XIO 40.

2. IOCB may be OPEN.

3. IOCB not OPEN to another device.

4. System Reset will not change states established by XIO 36.

5. AUX1 = BAUD Rate + Word Size + Stop Bits.

6. AUX2 = Input Circuits.

7. BAUD Rate = Figure 14.

BAUD RATE VALUE	BAUD RATE	BAUD RATE VALUE	BAUD RATE
0	300	8	300
1	45.5	9	600
2	50	10	1200
3	56.875	11	1800
4	75	12	2400
5	110	13	4800
6	134.5	14	9600
7	150	15	9600

Figure 14: BAUD RATE

8. If OPEN = concurrent mode then BAUD rate less than or equal 300.

9. Word Size = Figure 15.

10. If Word Size less than 8 then output = Block Output mode and input = concurrent (half duplex).

WORD SIZE VALUE	WORD SIZE
0	8 bits
16	7 bits
32	6 bits
48	5 bits

Figure 15: WORD SIZE

11. If Word Size less than 8 then high order bits truncated for output and filled with 1s for input.
12. Stop Bits = Figure 16.

WORD SIZE VALUE	STOP BITS/WORD
0	1
128	2

Figure 16: STOP BITS SENT PER WORD

13. Incoming Circuits = Figure 17.
14. DTR not supported on port 4.
15. CTX and CRX not supported on ports 2, 3, and 4.

Specifications

CIRCUIT MONITOR VALUE	TO MONITOR
0	None
1	CRX
2	CTS
3	CTS, CRX
4	DSR
5	DSR, CRX
6	DSR, CTS
7	DSR, CTS, CRX

Figure 17: MONITOR INCOMING CIRCUITS

XIO 36 is used to set the BAUD Rate, Word Size, number of Stop Bits, and Incoming Circuit ready monitoring values in the interface module tables. Data can be received and sent at discrete rates ranging from 45.5 to 9600 BAUD. The baud rate set must meet the protocol established between both computers. The Atari system can communicate with systems that use from 5 to 8 bit words. Sending and receiving short words may require translation. For output the high order bits are truncated to meet the word size specification, while for input the high order bits are set to 1 to Meet the Atari word size requirement. The Atari system will apply either one or two stop bit to each word of output. The stop bits are added to the character bits by the interface module

prior to transmission. Any combination of one to all three of the incoming circuits can be monitored.

2.3.3.5 XIO 38 TRANSLATION and PARITY

Syntax

```
XIO 38, IOCB#, AUX1, AUX2, "R#:"
```

Assertions

1. XIO 38 must be used before XIO 40.
2. Values set to interface module are not changed by System Reset.
3. AUX1 = Translation + Input Parity + Output Parity + Append Line Feed.
4. AUX2 = Won't Translate Character.
5. Translation = Figure 18.
6. If output and Translation = light then Atari EOL changed to ASCII CR and high order bit set to 0.
7. If input and Translation = light then ASCII CR changed to Atari EOL and high order bit set to 0.
8. If input and Translation = heavy and word = ASCII (decimal 32 to 124) then high order bit set to 0 and

TRANSLATION VALUE	TO GET
0	LIGHT ATARI/ASCII TRANSLATION
16	HEAVY ATARI/ASCII TRANSLATION
32	NO TRANSLATION

Figure 18: TRANSIATION

word sent to SIO else word set to Won't Translate Character and sent to SIO.

9. If output and Translation = heavy and word = ASCII then high order bit set to 0 and word sent else word changed to Won't Translate Character and sent.
10. If input and Translate = heavy and Won't Translate Character = 0 and word not equal ASCII then word not sent to SIO.
11. If output and Translate = heavy and Won't Translate Character = 0 and word not equal ASCII then word not sent.
12. If Translate = none then words are received and sent as formed.
13. If Word Size less than 7 and Translate = light or heavy then word = garbage.
14. Input Parity = Figure 19.
15. Input Parity disabled if WordSize less than 8 bits.

INPUT PARITY VALUE	TO GET
0	IGNORE, NO CHANGE
4	CHECK ODD, CLEAR
8	CHECK EVEN, CLEAR
12	NO CHECK, CLEAR

Figure 19: INPUT PARITY OPTICN

16. Output Parity = Figure 20.

OUTPUT PARITY VALUE	TO GET
0	DO NOT CHANGE PARITY BIT
1	SET OUTPUT PARITY ODD
2	SET OUTPUT PARITY EVEN
3	SET HIGH ORDER BIT TO 1

Figure 20: OUTPUT PARITY OPTICN

17. Output parity disabled if word size less than 8 bits.

18. Append Line Feed = Figure 21.

19. If Append Line Feed = ON then interface module sends ASCII line feed character immediately after each ASCII CR.

APPEND LINE FEED VALUE	TO GET
0	DO NOT APPEND LF
64	APPEND LF AFTER CR

Figure 21: APPEND LINE FEED OPTION

20. Append Line Feed disabled for input.

Specifications

The XIO 38 command is used to set values in the interface module tables to implement code conversion (translation), parity, appending line feed, and establishing a won't translate character. Internally the Atari system uses an 8 bit character code which includes the ASCII character set as a subset. This results in most non-Atari devices not recognizing the full Atari character set. Translation will convert the Atari unique characters to the ASCII character set. Three levels of character translation can be selected.

1. No Translation results in the interface module receiving the 8 bit Atari character from SIO and sending that part of the character that remains when the word size option of the XIO 36 command is

invoked. On input the word will be padded with 1s to fill 8 bits, if required.

2. Light Translation is applicable only if 8 bit words are being sent. On output the high order bit (7) is set to 0 and the Atari End of Line (EOL) character, decimal 155, is converted to the ASCII CR, decimal 13. On input the high order bit is set to 0 and the ASCII CR converted to an Atari EOL.

3. Heavy Translation assumes little correlation between ASCII and the interface character set. On output the high order bit is set to 0, all characters with a decimal value between 32 and 124 are sent, the Atari EOL is coverted to an ASCII CR and sent, and the remaining characters are not sent. On input the high order bit is set to 0. Characters that match the ASCII character set are sent to SIO. If the character is an ASCII CR it is converted to an Atari EOL. Otherwise the character is converted to the "Won't Translate Character".

If the received character set is totally different from ASCII the user should accept the character without translation and devise his own translator. Parity checking must be consistant with the protocol. However parity checking is disabled if short words are being received or sent. This is because bit 7 will have been set to 1 when the

short word was filled to 8 bits. The ASCII line feed (LF) character may be added after each ASCII CR (translated from Atari EOL). The "won't translate character" will be inserted in the input instead of a non-ASCII character when heavy translation is used. The decimal value of a single ASCII character, the "won't translate character", will be entered in the AUX2 field of the XIO 38 command. This provision applies only to Heavy Translation. With Heavy Translation and no "won't translate character" the non-ASCII input will not be sent to SIO.

2.3.3.6 XIO 34 Control of Outgoing Circuits

Syntax

```
XIO 34, IOCB#, AUX1, AUX2, "R#:"
```

Assertions

1. XIO 34 must be used before XIO 40.
2. Values set to interface module are not changed by System Reset.
3. AUX1 = DTR + RTS + XMT.
4. DTR, RTS, and XMT = Figure 22.

DTR VALUE		TO GET
0	.....	NO CHANGE FROM CURRENT SETTING
128	.....	DTR OFF
192	.....	DTR ON
RTS VALUE		**TO GET**
0	.....	NO CHANGE FROM CURRENT SETTING
32	.....	RTS OFF
48	.....	RTS ON
XMT VALUE		**TO GET**
0	.....	NO CHANGE FROM CURRENT SETTING
2	.....	SET XMT TO SPACE (0)
3	.....	SET XMT TO MARK (1)

Figure 22: VALUES FOR DTR, RTS, XMT

5. If OPEN = Block Output mode and XMT = space then BREAK is sent.

Specifications

In addition to meeting a variety of protocol requirements in hardware the Atari system allows control of the outgoing circuits DTR, RTS, and XMT. These circuits can be controled individually at each of the RS-232-C ports that has the circuit available. SEE FIGURES 4 and 5 above. The XIO 34 command is used to set table values in the interface module

and may be used after the port has been OPENed in the Block Output mode(to be discussed later). This command allows the user to send a BREAK by changing the state of the XMT circuit. The normal resting state of XMT is MARK (RS-232-C Standard). An extended space state is interpreted to be a BREAK signal. Use of this convention is usually based on a conditional situation such as the when buffer is full. The protocol in effect dictates the use of this convention. The AUX2 field of the XIO 34 command is not currently used, placing any value in AUX2 other than 0 may result in undesirable results.

2.3.3.7 OPEN COMMAND

Syntax

```
OPEN IOCB#, AUX1, AUX2, "R#:"
```

Assertions

1. OPEN is used by CIO to set IOCB bytes Handler I.D., Device Number, Put Address (low), and Put Address (high). See Table 3.
2. Values set by the OPEN will be reset by System Reset.

3. When OPEN is invoked IOCB may not be OPEN to another device.

4. AUX1 = Figure 23.

ENTER IN AUX1	TO USE THE PORT FOR	MODE
5	INPUT ONLY	CONCURRENT
8	OUTPUT ONLY	BLOCK
9	OUTPUT ONLY	CONCURRENT
13	INPUT and OUTPUT	CONCURRENT

Figure 23: I/O MODE VALUES

5. AUX1 is used by CIO to set I/O direction indicators the IOCB Auxillary Information byte.

6. If mode = concurrent then BAUD Rate less than or equal 300 and no other peripherals may be used until port is CLOSEd.

7. If mode = concurrent then Basic I/O commands LIST, SAVE, LOAD, and ENTER may not be used.

8. If mode = Block Output then no input is received.

9. AUX2 = 0.

10. The device field is used by CIO to select I/O primitives for the handler vector table.

Specifications

CIO uses the values in the OPEN command to set the Handler I.D., Device Number, and Auxillary Information bytes of the IOCB and establist the primitive I/O utilities that are assigned to the Handler Vector Table. Other configuring information is sent to the interface module to set table values. In essence the OPEN command causes the system to recognize the RS-232-C ports as devices. The OPEN command may not be used to an IOCB that is already OPEN. Likewise two IOCBs may not be OPENed to the same device. The AUX1 field of the OPEN command is used to set the direction of I/O operations in the Auxillary Information byte (ICAX1) and causes device specific information to be placed in the second byte of that field (ICAX2). A value other than 0 in the AUX2 field may result in unpredictable and undesirable results.

2.3.3.8 FORCE SHORT BLOCK

Syntax.

```
XIO 32, IOCB#, AUX1, AUX2, "R#:"
```

Assertions

1. XIO 32 does not set values in any tables or blocks.
2. XIO 32 forces the device handler to release a data block prior to normal release.
3. XIO 32 is used in Block Output mode only.
4. AUX1 = 0.
5. AUX2 = 0.

Specifications

In the Block Output mode the device handler causes SIO to send 32 byte blocks of data to the interface module. While the interface module sends that block, the computer waits and other device handlers have access to SIO. Therefore other devices such as the printer or disk unit may be used while the RS-232-C port is OPEN in the Block Output mode. Block Output mode causes the device handler to place 32 bytes in the Page 4 memory buffer area. SIO is called to send this buffer to the interface module on three conditions.

1. The buffer fills up.
2. The IOCB is CLOSEd
3. An Atari EOL is placed in the buffer.

The XIO 32 command may be used to override these normal conditions by forcing the contents of the buffer to be sent at the point the command is invoked. This is normally done as a conditional event, such as sending a few characters at

a time to ensure the receiving device is not overloaded. Block Output mode is used for output only. No input may be received in this mode. Neither the AUX1 or AUX2 fields of the XIO 32 command are currently used. Values other than 0 in the AUX1 and AUX2 fields may result in unpredictable and undesirable results.

2.3.3.9 CLOSE COMMAND

Syntax

CLOSE IOCB#

Assertions

1. If IOCB OPENed in concurrent mode then IOCB must be CLOSEd before other peripherals can be used.
2. CLOSE does not change the state of parameters in the interface module.

Specifications

Peripheral devices such as the printer and the disk unit may be used when the port is OPEN in Block Output mode without CLOSEing the port. In concurrent Mode the port must be CLOSEd before other devices can be used. Further it is

important to CLOSE the concurrent OPEN before any other CLOSE occures. Failure to do this can cause unpredictable results.

2.3.3.10 XIO 40 START CONCURRENT I/O

Syntax

```
XIO 40, IOCB, AUX1, AUX2, "R:"
```

Assertions

1. XIO 40 must be used after IOCB is OPEN and mode = concurrent.
2. Concurrent I/O is terminated by System Reset or CLOSE.
3. XIO 40 sets values in the interface module tables.
4. After XIO 40 external devices, other that one OPEN port, may not be used.
5. AUX1 = 0.
6. AUX2 = 0.
7. If OPEN and mode = concurrent then device handler sends each character as its read to the interface module for transmission.

8. When XIO 40 is invoked the interface module buffer is used for input.
9. If OPEN and mode = concurrent and XIO 40 invoked then BAUD Rate must be equal or less than 300.

Specifications

XIO 40 changes the state of the interface module by changing the use of the internal data buffer. Unlike block output mode, concurrent mode sends each character to the interface module as it is read. The interface module then processes the characters one at a time. When input is expected, that data is stored in a 32 word buffer until the interface module is asked to sent it to SIO. If more data arrives than will fit in the buffer, the oldest data is overwritten. To preclude this the user program must process fast enough to clear the buffer before it fills.

2.3.3.11 BASIC I/O COMMANDS

This section is intended to address only those aspects of the Basic I/O commands that are related to inter-computer communications using the Atari 850 Interface Module. For a complete explanation of Atari Basic I/O the user should refer to the Atari Basic Users Manule.

A review of the buffers and data flow involved in the two I/O modes supported by the interface module is in order. Two

buffers are available; however, their use is determined by the I/O mode selected in the AUX1 field of the CPEN. Figure 24 illustrates the use of the buffers in the Block Output mode.

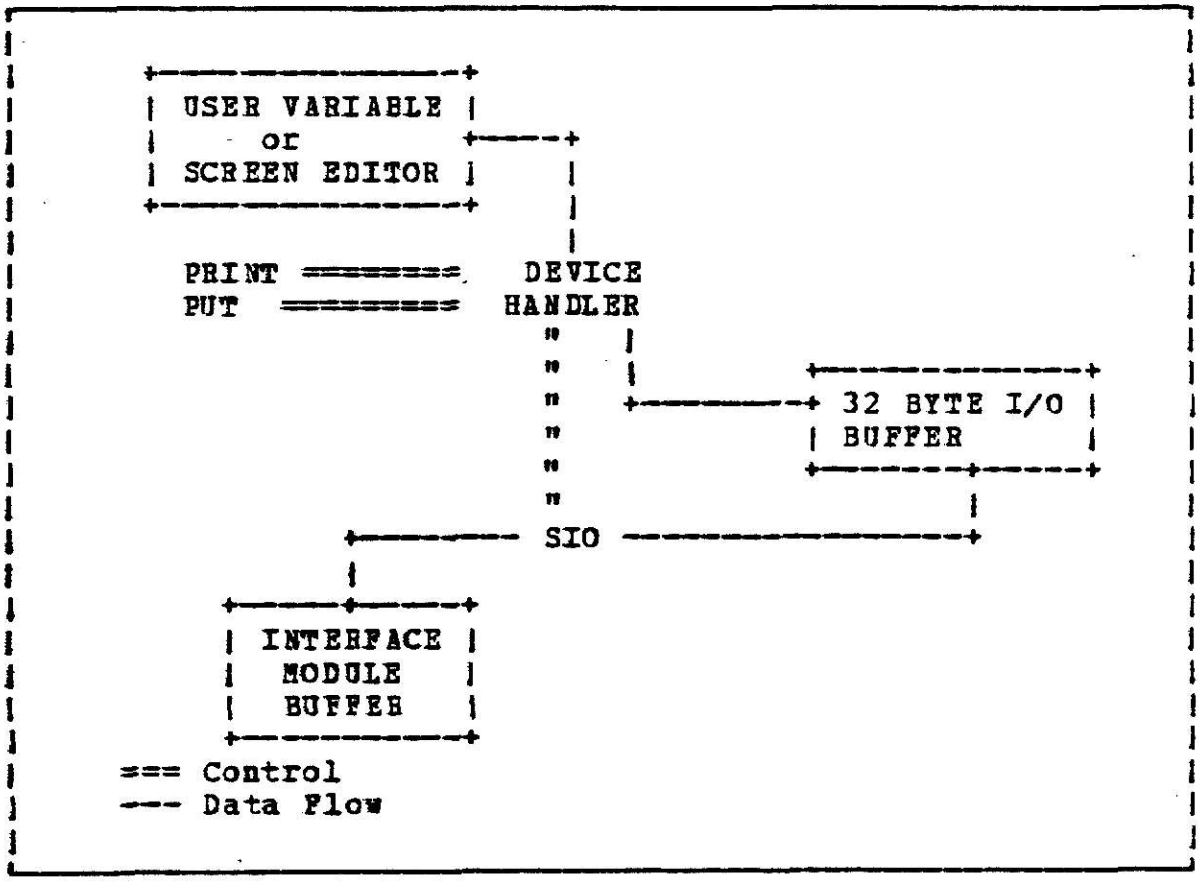

Figure 24: BLOCK OUTPUT I/O BUFFERS

Figure 24 shows the device handlers taking the data from a user defined variable or the screen editor and moving it to a 32 byte buffer located in page 4 memory (see Figure 6).

The buffer is allocated to support the RS-232-C port when the OPEN command is invoked. The IOCB bytes, Buffer Address (low) and Buffer Address (high) (see Table 3), point to the specific area allocated. When the device handler detects one of the following conditions, it calls SIO to pass the buffer contents to the interface module.

1. An Atari EOL is encountered.
2. The buffer is full.
3. The IOCB is closed.
4. The XIO 32 command is invoked.

The interface module accepts and stores the data in the interface module buffer prior to sending it on to the other computer. Input is not supported in the Block Output mode.

PUT causes a single 8 bit word to be placed in the I/O buffer. The buffer contents are not sent to the interface module until one of the release criteria listed above is met.

PRINT causes a single line of data ending with an Atari EOL to be moved to the I/O buffer. The device handler will take care of necessary interupts if the buffer fills before the EOL is encountered. The primitives that implement the Print algorithm require an EOL. Otherwise, an I/O error will occur.

Figure 25 illustrates the use of buffers and the data flow in the concurrent mode.

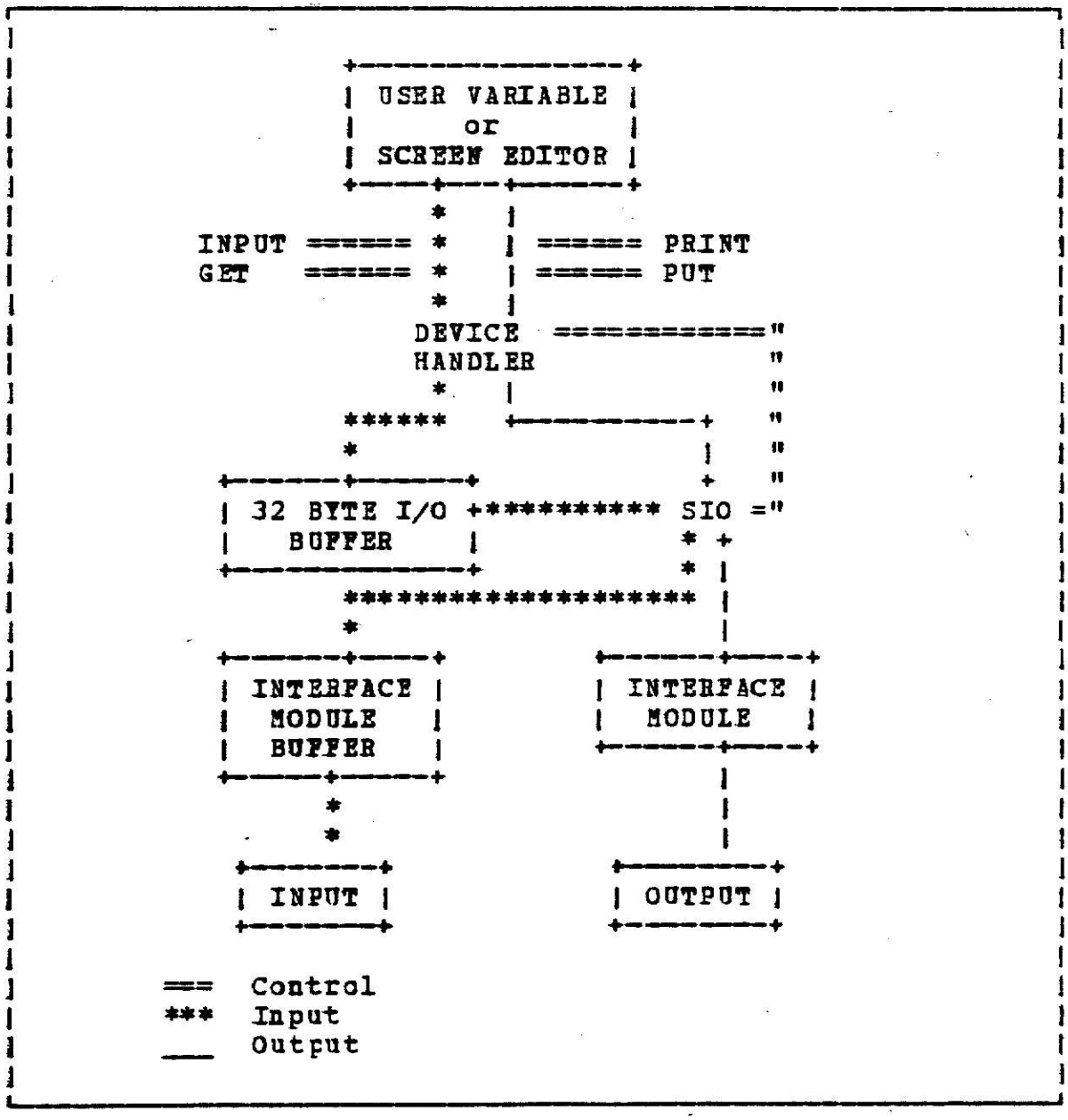

Figure 25: CONCURRENT I/O BUFFERS

In concurrent mode the interface module buffer and the I/O buffer are dedicated to recieving input from the other computer. Therefore, output is handled differently than in

the block output mode. In concurrent mode the output data is sent one word at a time through the interface module to the other computer, passing through the chain, device handler, SIO, interface module, and other computer, without being placed in a buffer.

Input from the other computer is stored in the interface module buffer until called. It then moves through the chain, interface module buffer, to SIO, to I/O buffer, to device handler, to user variable or screen editor. When the other computer sends more data than the interface module buffer will hold, there is no automatic provision to prevent loss as the oldest data in the buffer is overwritten.

PUT and GET cause a single word (8 bits) to be transfered. The difference is that in concurrent mode PUT causes the word to be sent directly to the other computer.

PRINT and INPUT are line oriented, but PRINT causes each word to be sent to the other computer until the EOL is encountered. Both commands require an EOL or an I/O error will occur.

Syntax

```
GET IOCB#, user defined variable
PUT IOCB#, arithmetic expression
INPUT IOCB#; user defined variable
PRINT IOCB#; user defined variable
PRINT IOCB#, user defined variable
```

Assertions (Global)

1. IOCB must be OPEN to direct I/O commands to a port.
2. If OPEN = concurrent mode then I/O buffer and interface module buffer dedicated to input.
3. If OPEN = Block Output mode then I/O buffer and interface module buffer dedicated to output.

Assertions (Output)

1. Basic output commands are PRINT and PUT.
2. If OPEN = Block Output mode and data stream longer than 32 bytes then device handler prevents loss of data.
3. If OPEN = Block Output mode then one of four release criteria must be satisfied.
   a) Buffer is filled.
   b) Atari EOL is encountered.

c) IOCB is closed.

d) XIO 32 command is invoked.

4. If OPEN = concurrent then I/O buffer and interface module buffer are not used.
5. If OPEN = concurrent then each 8 bit word is sent to interface module individually.
6. If OPEN = Block Output mode then INPUT and GET commands will cause I/O error.

Assertions (Input)

1. Basic input commands are INPUT and GET.
2. If OPEN = concurrent mode then input data can be lost in the interface module buffer.

Assertions (PRINT)

1. PRINT = line of data ending in Atari EOL.
2. If PRINT syntax = semicolon (;) then Atari EOL sent to interface module.
3. If PRINT syntax = comma (,) then Atari EOL not sent to interface module.

Assertions (INPUT)

1. INPUT = line of data ending in Atari EOL.
2. User defined variable = string variable or numeric variable.

3. If user defined variable = numeric then input data = digits, sign (optional), decimal point (optional), and exponent (optional).

Assertions (GET)

1. GET = transfer of one 8 bit word.

Assertions (PUT)

1. PUT = transfer of one arithmetic expression.
2. Arithmetic expression = 1 to 255.

Specifications

The output commands, PRINT and PUT, cause data to be transfered from the Atari computer to the interface module. In Block Output mode, 32 byte blocks of data are transfered. In concurrent mode both commands result in sending one byte at a time. The essential difference in concurrent mode output is that PUT must be invoked for each arithmetic expression (byte) transfered, while PRINT causes a line of data to be transfered one byte at a time until an EOL is encountered. String data must be converted to a decimal equivalent for PUT to be used.

The input commands, INPUT and GET, are used only in the concurrent mode. When INPUT is used the data type received must match the user defined variable. That is, Character

data cannot be INPUT to a numeric variable. Further INPUT requires that the input data stream contain Atari EOLs. That means the ASCII CRs must be converted to EOLs by translation in the interface module. If the input data stream cannot meet this requirement then the GET command should be used.

Chapter III

A SAMPLE IMPLEMENTATION

The physical and logical models of Chapter 2 will now be used to develope a brief Atari Basic program to demonstrate the applicability of the system to inter-computer communications. A Perkin Elmer model 8/32 will be used as the far end of this network. Preparatory actions will be described in the section Protocol followed by an Implementation section and a Comments section.

3.1 PROTOCOL

The Perkin Elmer serves a large number of customers so negotiation of the protocol is not possible. The protocol currently in effect must be used. The following list details the requirements to be met.

1. The ASCII character set is used.
2. The Perkin Elmer reflects a copy of each data character received.
3. The BAUD rate is set to 300.
4. The Perkin Elmer requires a DSR signal from all remote sites.
5. 7 bit words are used.

6. 2 stop bits are used.

7. Each Perkin Elmer transmission will contain lead and trail padding characters. Example: _ _ - data - _ _.

8. Circuit signal requirements are RS-232-C compatible.

The protocol requirements are within the capability of the Atari system; implementation is possible.

3.2 IMPLEMENTATION

Having determined that the protocol requirements can be met, implementation becomes a matter of configuring the Atari system to perform within the stated limits. This will be discussed in three parts; Physical Environment, Configuration of Ports, and Execution.

3.2.1 PHYSICAL ENVIRONMENT

Physical preparation of the system consists of ensuring that the required incoming and outgoing circuits are properly matched. Figure 26 shows the relations needed.

ATARI		PERKIN ELMER
DTR	<======>	DSR
Receive Data	<======>	Send Data
Send Data	<======>	Receive Data
Signal Ground	<======>	Signal Ground

Figure 26: CIRCUIT MATCHING

3.2.2 CONFIGURATION OF PORTS

The default or initial state for the ports is shown in Table 4 above. While the parameters BAUD rate, append line feed, and translation meet the protocol requirements, the remaining parameters will require a state change. The Atari Basic commands that will make the needed changes are XIO 34 and XIO 36. These commands are shown in Figure 27 below.

```
XIO 34, #2, 192 + 3, 0, "R1:"
XIO 36, #2, 16 + 128, 0, "R1:"
```

Figure 27: EXAMPLE CONFIGURATION COMMANDS

The AUX1 value of the XIO 34 command sets DTR = ON and XMT = MARK. This will cause the interface module to send the expected signal to the Perkin Elmer on DTR. The Atari users manuals do not tell the initial state of the XMT circuit so it is set to MARK to be sure. The AUX1 value of the XIO 36 command changes Word Size to 7 and Stop Bits to 2, meeting the protocol. Translation is left on Light to handle the CR and EOL conversions.

Translation requires some explanation as it will be used in conjunction with short words. A review of the interface module input and output function sequence is in order. When

the interface module receives a 10 bit word from from the Perkin Elmer, the start bit and two stop bits are stripped and the 7 data bits expanded by setting the high order bit to 1. Parity, the next step is disabled because the word size is less than 8. Translation sets the high order bit to 0, checks for a CR, changes it to an EOL, then sends the word to SIO. When the interface module receives the word from SIO the opposite sequence occurs. Translation checks for an EOL, changes it to CR, then sets the high order bit to 0. Again parity is disabled because of word size. The high order bit is truncated and the word is sent. Therefore, Translation can be used even though short words are being sent.

3.2.3 EXECUTION

A brief Atari Basic program is listed below to demonstrate the use of the port configuration commands by sending a "Sign on" message and receiving a responce.

```
10  DIM OUTREC$ (32)
20  LET OUTREC$ = "SIGNON NAME,21,CS720"
30  XIO 34, #2, 192 + 3, 0, "R1:"
40  XIO 36, #2, 16 + 128, 0, "R1:"
50  OPEN #2, 8, 0, "R1:"
60  PRINT #2; OUTREC$
```

```
 70  XIO 32, #2, 0, 0, "R1:"
 80  CLOSE #2
 90  OPEN #2, 13, 0, "R1:"
100  INPUT #2; OUTREC$
110  CLOSE #2
120  PRINT OUTREC$
130  STOP
```

The trival program above has little value other than demonstrating the port configuration commands. The program configures the port, lines 30 and 40, to meet the protocol discussed above. The IOCB is OPENed to the port, line 50, in Block Output mode. The "signon" message is output, lines 60 and 70, and forced to the interface module. The port is CLOSEd, line 80, and reOPENed in concurrent I/O mode, line 90. The Perkin Elmer responce is captured, line 100, and the port CLOSEd, line 110. That portion of the responce that was not overwritten is printed to the monitor, line 120.

3.3 COMMENTS

The example does not work. Although it demonstrates the use of the Atari configuration commands, it fails. It appears that there are undocumented faults in the Atari OS which cause the failure.

BIBLIOGRAPHY

<ANSI-77> American National Standards Institute, "American National Standard for Writing Abstracts", IEEE Transactions on Professional Communications, Vol. PC-20, No. 2, December, 1977, pp. 252-254.

<ATAR-80> ATARI 850 Interface Module Owner's Manual, Atari, Inc., Sunnyvale, California, 1980.

<ATAR-82a> ATARI Personal Computer System Hardware Manual, Atari, Inc., Sunnyvale, California, 1982.

<ATAR-82b> ATARI Personal Computer System Operating System User's Manual, Atari, Inc., Sunnyvale, California, 1982.

<BARN-74> Barnett, Marva T., Elements of Technical Writing, Delmar Publishers, Albany, New York, 1974.

<DEGI-80> De Gise, Robert F., "A Systems Approach to Business Writing", IEEE Transactions on Professional Communications, Vol. PC-23, No. 2, June, 1980, pp. 77-78.

<DOLL-78> Doll, Dixon R., Data Communications Facilities, Networks, and Systems Design, John Wiley and Sons, New York, New York, 1978.

<EMBE-55> Emberger, Meta Riley and Hall, Marion Ross, Scientific Writing, Harcourt Brace and Co., New York, New York, 1955.

<FITZ-78> FitzGerald, Jerry and Eason, Tom S., Fundamentals of Data Communications, John Wiley and Sons, New York, New York, 1978.

<FULL-80> Fuller, Don, "How to Write Reports That Won't Be Ignored", IEEE Transactions on Professional Communications, Vol. PC-23, No. 2, June, 1980 pp. 79-81.

<GOLD-81> Goldfarb,Stephan M., "Writing Policies and Procedures Manuals", Journal of Systems Management, Vol. 32, No. 4, April, 1981, pp. 10-13.

<HALP-76a> Halpern, M., "Documentation", Encyclopedia of Computer Science, Mason/Charter Publishers, Inc., New York, New York, 1976, pp. 503-517.

<HALP-76b> Halpern, M., "Bootstrap", Encyclopedia of Computer Science, Mason/Charter Publishers, Inc., New York, New York, 1976, pp. 173-174.

<HOWA-81> Howard, Jim, "What is Good Documentation", BYTE, Vol. , No. , March, 1981, pp. 141-150.

<MICH-68> Michaelson, Herbert B., "Structure, Content, and Meaning in Technical Manuscripts", Technical Communications, Vol. 15, Second Quarter, 1968, pp. 15-18.

<PEAR-76> Pearcy, T. and Pine, M., "Channel", Encyclopedia of Computer Science, Mason/Charter Publishers, Inc., New York, New York, 1976, pp. 200-206.

<ROSS-75> Ross, Douglas T., et al, "Software Engineering: Process, Principles, and Goals", Computer, Vol. , No. , May, 1975, pp. 17-27.

<ROSS-77> Ross, Douglas T., "Structured Analysis: A Language for Communicating Ideas", IEEE Transactions on Software Engineering, Vol. SE-3, No. 1, January, 1977, pp. 16-34.

<SCHW-77> Schwartz, Mischa, Computer Communications Network Design and Analysis, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

<SIPP-76> Sippl, Charles J., Data Communications Dictionary, Van Nostrand Reinhold Co., New York, New York, 1976.

<STRU-79> Strunk, William Jr. and White, E.B., Elements of Style, MacMillan Publishing Co., New York, New York, 1979.

<TEBE-80> Tebeaux, Elizabeth, "What makes Technical Writing Bad? A Historical Analysis", IEEE Transactions on Professional Communications,

Vol. PC-23, No. 2, June,1980, pp. 71-76.

<VAND-82> Van Duyn, J., The DP Professional's Guide to Writing Effective Technical Communications, John Wiley and Sons, Englewood Cliffs, New Jersey, 1982.

<VALD-79> Valdur, Silbey, "Documentation Standardization", Data Management, Vol. 17, No. 4, April, 1979, pp.32-34.

<WEIS-82> Weiss, Edmund H., The Writing System for Engineers and Scientists, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982.

STYLE OF USER DOCUMENTATION FOR MICRO COMPUTERS

BY

ROLF LEE COOK
BBA UNIVERSITY OF MIAMI 1973

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

AN ABSTRACT OF A MASTER'S REPORT

SUBMITTED IN PARTIAL FULLFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

KANSAS STATE UNIVERSITY
MANHATTAN, KANSAS

- 1983 -

ABSTRACT

This report developes a simple yet formal model of inter-computer communications for an Atari micro computer system. The model is presented as an example of desirable structure for technical documentation. Finally the model is used to illustrate an example Atari configuration for inter-computer communication with a Perkin Elmer 8/32.