This is the technical specification for the Apple /// computer's RS-232 Serial driver.
This information originated in the files RS232ERS.TEXT and RS232ERS1.TEXT

Printed by David T. Craig -- 10 April 1998

EXTERNAL REFERENCE SPECIFICATION

APPLE III ON-BOARD RS-232 DEVICE DRIVER

Version 1.3

April 20, 1981

Author:

Jim Trezzo

RS-232 Driver ERS SIGNOFF SHEET Date: April 20, 1981

_____
Engineering Manager

_____
Product Manager

_____
Publications Manager

_____
NPR Manager

RS-232 Device Driver

1. Project Identification

1.1 Product name: Apple III On-Board RS-232 Device Driver

1.2 Project number: I-154

1.3 Related documents

- Marketing Requirements Document
Pete Sinclair - 7/31/80

- Engineering Project Authorization
Jim Trezzo - 8/22/80

- Product Requirements Document
Jim Trezzo - 9/12/80

- Product/Marketing Requirements Document
Jim Trezzo / Pete Sinclair - 12/12/80

- Detailed Project Schedule
Jim Trezzo - 1/23/81

- SOS ERS Appendices (SOS System Calls)
Bob Etheredge / Don Reed - 9/30/80

- SARA PASCAL ERS
Al Hoffman / Jim Jatczynski - 4/15/80

- Apple Business BASIC Reference Manual

- Apple III - Standard Device Drivers

1.4 Product abstract

- Intended use

This driver complements the RS-232 port printer driver already written for the Apple III by providing bidirectional communications capabilities. This driver is required in order to send and receive data or set the device characteristics (baud rates, parity, etc.) of the on-board serial port.

Because of these reasons and the needs of future application software, it is desirable to have a standard SCP installable SOS device driver which fully utilizes the on-board serial port.

- Target market

The Serial Port, which this driver supports, is a general purpose device. The RS-232 Driver will therefore be found in use in most

market areas, including Business, Professional, Scientific, Educational and Hobby.

- Typical user

There will be three classes of users who will require this driver. The primary users will wish to interface their system to a central computer for high speed data processing or information transfers. Many of our Fortune 500 customers will be using this mode. A second group of users will use the driver to communicate with the variety of nationwide data banks now forming. A good example of such a data bank is the Dow Jones service Apple is now working with. (Such customers will be using their own modem and not Applephone III which will not use this driver). The final group are those users who want to connect a terminal to the Apple III for data entry, an RS-232 device (bar code reader) for data capture or to otherwise communicate between Apple systems. To such users, this driver is indispensable.

- Relation to other Apple products

The System Configuration Program (SCP) will be used to build the SOS Driver file, which contains all peripheral device drivers required by SOS. SCP is used to add, edit and delete individual drivers including the RS-232 driver file.

With SCP the default parameters of the device ".RS232" may be edited. This allows you to specify the baud rate, parity type, number of start and stop bits, etc. that will be set when a SOS OPEN is performed for this device.

The RS-232 Driver is a standard SOS device driver and as such can only be used through a set of SOS System Calls. The language facilites, both BASIC and Pascal, allow an Application or a Utility program access to much of the functionality provided in the device driver.

If full functionality is needed, an Assembly Language module will be required to perform the SOS System Calls.

When terminal emulation programs, such as the VT-100 Emulator for the Apple II, are implemented on the Apple III, they will use the RS-232 Driver through SOS System Calls.

1.5 Use environment

1.5.1 Hardware environment

- Host machine

An Apple III hardware configuration on which SOS will operate.

- Memory requirements, options

The RS-232 device driver will require approximately 1.7K bytes of memory from the device driver space of SOS, this includes an input and an output buffer of 256 bytes each.

- Mass storage requirements

A single disc drive system with adequate memory will be capable of using this device driver. The SCP utility can be run on this system to add, delete or modify the driver file.

- Peripheral requirements

The On-Board RS-232 Port is an integral part of the Apple III computer. Therefore no additional hardware is required beyond the external RS-232 device (terminal, modem, etc.) and its cable.

- Other hardware requirements

The Apple III computer is classified as Data Terminal Equipment (DTE) under the EIA RS-232-C Interface Standard.

This standard allows a DTE device to connect to Data Communications Equipment (DCE) with a standard cable and connectors. When the Apple III is interfaced to a DTE device, a modem eliminator will be required (supplied with the Apple III). Most terminals (printers and crt's) are DTE devices. Modems and Multiplexers are examples of DCE devices.

1.5.2 Software environment

- Operating system

This driver will function as part of the SOS software environment and is accessed via device management system calls to the SOS Kernel.

All SOS requests from the user or interpreter software are handled by SOS and routed to the specified device driver request handler when appropriate. The device driver performs the request and returns control to SOS. SOS then restores the user environment and returns control to the user.

The device driver is not stand alone software.

There is no direct communication between the user and the driver. All communication is through SOS, even hardware interrupts go through the Interrupt Receiver of SOS first.

- SOS System Call Facility

The following SOS System Calls are supported by this driver;

D_READ
D_WRITE
D_STATUS
D_CONTROL
OPEN
CLOSE

- Pascal Interface

Peripheral interaction in Apple III Pascal is based on the BIOS concept, similar to the Apple II version.

Units 7 and 8 and the names REMIN and REMOUT are assigned to interface with the On-Board RS-232 port.

The characteristics are similar to the Apple II Remote port.

A new routine, UNITSTATUS, is used to control devices and to examine their status.

The format is;

UNITSTATUS(UNITNUM: INTEGER; VAR STATUSWORDS; CONTROL: INTEGER);

UNITNUM is the desired unit, i.e. "7" for RS-232 Port.

STATUSWORDS may be any variable; generally it is a record or array of control or status information.

The CONTROL parameter has the following format:

bit 0=0 for output channel, 1 for input channel
(either if there is only one channel)
bit 1=0 for status operation, 1 for control operation
bits 2-12 = device defined control or status parameter
bits 13-15 reserved

For a status operation, the status bits (bits 2-12 of CONTROL) usually select the subcomponent of the device whose status is to be checked, and STATUSWORDS is a record of status information passed back to the user.

For a control operation, the control bits (bits 2-12 of CONTROL) usually select the type of control operation to be performed on the selected device, and STATUSWORDS is a record or array of control information.

- BASIC Interface

The device name ".RS232" is used in the BASIC OPEN statement. The standard BASIC I/O operations are available for this device. INPUT or GET can be used to read characters from the driver. PRINT is used to write to the driver.

- Assembly-Language Interface

When maximum control of the functionality within the RS-232 Driver is required, the SOS System Calls should be used directly by an application program. Either as an assembly-language subroutine, callable from both BASIC or Pascal, or as a stand alone assembly-language program. In many cases, dramatic execution speed improvements can be realized through the use of assembly-language code.

Detailed descriptions of the calling conventions and parameter lists are documented in the SOS ERS Appendices.

- Utilities

The System Configuration Program (SCP) is required to install the driver into the SOS environment. SCP is also used to modify the configuration parameters contained in the Device Configuration Block of the driver. The Utilities diskette contains the SCP program.

- Terminal Emulator

A Terminal Emulator program, such as the VT-100 Emulator written for the Apple II computer, allows an Apple computer to act as a conversational terminal on a large timesharing system.

The central computer views the Apple as a recognizable terminal attached to one of its asynchronous communication ports.

Typically some file transfer capability is available with an emulator program. This capability allows an Apple user to edit programs and files locally on the Apple, and then transfer these file to a central computer for processing. The user of this type of program has all the convenience and capability of a Personal Computer, but additionally saves connect time charges to the central timesharing system.

A Terminal Emulator program for the Apple III could be written in a high level language such as Pascal. The functionality of the RS-232 Device Driver would be available to the high level language either through the language interface to SOS or through direct SOS Calls from Assembly Language Subroutines. By writing an emulator within the SOS environment, development time could be greatly reduced and the facilities of SOS would be available to the emulator user.

2. Functional specifications

2.1 General overview

- Terms and concepts

The On-Board RS-232 Device Driver is designed to allow application programs to access and control the built-in RS-232 serial port (not the serial card) of the Apple III.

Through standard SOS calls, the application program is able to both send and receive character streams. The port characteristics are resettable via SOS calls. With this driver and the appropriate application software, a terminal or an RS-232 device (e.g., bar code reader) can be attached to the system. Additionally the system can connect to a central computer, timesharing system or data communication network as an intelligent terminal.

The RS-232 driver is a SOS compatible standard device driver which supports most of the hardware features of the On-Board RS-232 serial port.

Device characteristics (baud rates, parity, etc.) are settable via SOS calls or are specified with the System Configuration Program (SCP). All useful baud rates available on the serial port, up to 9600 baud, are supported.

The driver is interrupt driven so as to allow some interleaved processing. The driver supports the XON/XOFF communications protocol, which is used on many popular terminals including the VT-100 and DECWRITER III from DEC. The protocol is also used on Telenet and Tymnet, two of the larger data communication networks.

Optionally, the ENQ/ACK communication protocol is supported, which is used on the 2645 terminal from HP.

The driver is able to create timing delays which are sometimes required on unbuffered terminals after a carriage return, line feed or form feed character is sent.

In order to utilize this driver properly, the input/output philosophy should be understood. There are both an input and an output buffer contained within the driver code. Each buffer has a capacity of 255 characters. When the RS-232 device driver is opened (OPEN call), a reset is performed on the driver. Parameters are initilized to their values as specified in the device configuration block (set up by SCP). The buffers are cleared and the receive interrupt of the RS-232 port is enabled. Any characters received are stored in the input buffer until they are retrieved with a device read (D_READ) SOS call.

Several different actions are taken for a D_READ, depending on device parameter states. In the simple case, a read request is issued to SOS with the "bytes" parameter equal to "n". Both the "is_newline" and the "imed_read" device parameters are zero. Characters are transfered from the input buffer of the driver to the user buffer until "n" characters have been transfered.

If there are less than "n" characters in the driver buffer, the read operation will wait until enough characters are received to satisfy the requested amount "n".

If the "is_newline" parameter was set (a value of [80] hex), the read request would terminate early if a newline character (character contained in the "newline" parameter) was retrieved. BASIC usually sets "newline" to carriage return (ASCII CR - [0D] hex).

If the immediate read parameter ("imed_read") was set (a value of [80] hex), a read request would retrieve all characters contained in the driver buffer at the time the request was made. Requested count satisfied or newline character received would terminate the read request before the buffer was emptied. If the driver buffer was empty, the "bytes_read" parameter would be set to zero and control would immediatly be returned to the user.

Combinations of values for the driver control parameters allow the RS-232 port driver to be configured for many different uses.

A write request (D_WRITE SOS call) transfers characters from the user buffer to the driver output buffer. If the driver buffer can hold all the characters to be written, control will be returned to the user almost immediatly and the characters will be transmitted by the driver on an interrupt basis while the user program is free to perform other tasks. If there are more characters to be written than there is room in the driver buffer, the write request will wait until room becomes available.

SCP is used to install the driver into SOS and modify the device configuration parameters.

The data set and modem control signal handling conform to Bell 103 and 212 type conventions.

This driver only supports the built-in serial port of the Apple III computer.

The SOS operating system is required for use of this driver, it is not a stand alone device handler.

Data loss may occur at the higher baud rates, if other high speed interrupt devices are operating simultaneously or the SOS interrupt handling conventions are not adhered to.

- Resident code size

Current size estimates indicate that the code portion of the device driver should be 1.2K bytes of memory. Any data buffer space contained within the driver would be in addition to this estimate.

- Resident data capacity

The driver code uses both an input and an output buffer for character storage. These buffers are contained within the code space of the driver. An additional .5K bytes (two 256 byte buffers) are added to the driver size as a result of the buffers. This brings the total estimated driver size to 1.7K bytes.

- User-critical response times

The performance goal is no data loss at 9600 baud operation. If large ammounts of data are to be sent to the Apple, the user program must empty the driver's input buffer quickly enough to prevent the 255 character buffer from overflowing. If this is a problem for the user program, a communications protocol should be used to control character transmission to the Apple.

2.4 Compatibility requirements

- Call sequences

The SOS Device System Calls as described in the SOS ERS Appendices will be supported.

- Communication protocols

The XON/XOFF communications protocol may be selected by running the SCP utility. The communication protocol parameter (contained in the device configuration block) must have a value of [80] hexadecimal in order to enable XON/XOFF. Control character one and control character two must be set to XOFF (ASCII DC3 - [13] hex) and XON (ASCII DC1 - [11] hex) respectively, in the device control block.

If enabled, this protocol will control both the input and output buffers of the driver. If the input buffer is too full (buffer count has reached the maximum buffer level), the XOFF character (ASCII DC3 - [13] hex) is transmitted. The computer sending to the Apple responds to this control character and suspends transmission until it receives an XON character (ASCII DC1 - [11] hex). When the input buffer count becomes less than the minimum buffer level, the driver sends the XON character signaling that transmission may resume to the Apple. The maximun buffer level and minimum buffer level may be adjusted by modifying the device configuration block with SCP.

The output buffer is controled in a similar manner by this protocol. When an XOFF is received by the Apple, output will be suspended (within one or two character times) until an XON character is received by the Apple.

The Enquiry-Acknowledge (ENQ/ACK) protocol is available as an alternate protocol. This software handshake method is also selected through the SCP utility, but behaves differently than the XON/XOFF handshake. The communications protocol parameter in the device configuration block must have a value of [40] hexadecimal in order to enable ENQ/ACK. Control character one and control character two must be set to ENQ (ASCII ENQ - [05] hex) and ACK (ASCII ACK - [06] hex) respectively in the DCB. The data block length parameter must also be specified in the DCB.

The ENQ/ACK handshake depends on the two communicating devices using the same data block size. If an HP 2645 terminal was connected to an Apple III, the data block length would be 80 decimal. The Apple would send an ENQ character to the terminal (asking if there was room for 80 characters) and wait until the terminal responded with an ACK character (yes, there is room). The Apple would then transmit up to 80 characters from the device driver's output buffer. When there are more characters to be sent, the Apple will first send an ENQ character. After an ACK is received, the data block can be sent. This exchange would be repeated each time the Apple had data to send.

The ETX/ACK handshake (similar to the ENQ/ACK handshake) may be specified by following the ENQ/ACK instructions and substituting ETX (ASCII ETX - [03] hex) for ENQ as control character one in the DCB. This handshake is used by some serial printers.

Either ENQ/ACK, XON/XOFF or no protocol may be selected as a device characteristic.

- Hardware handshake

The Hardware handshake may be selected by the SCP utility. When enabled, the Data Set Ready (DSR) and Data Carrier Detect (DCD) signals on the Apple III RS-232 connector are monitored by the device driver. If either signal becomes "false", output will be suspended from the driver until both signals are "true".

If the input buffer is too full (buffer count has reached the maximum buffer level), the Request to Send (RTS) signal on the RS-232 connector will become "false". RTS will remain "false" until the input buffer count becomes less than the minimum buffer level. RTS will then become "true", indicating to the transmitting device that room is available in the input buffer.

When two Apple IIIs are connected, the modem eliminator (required to interface two Apple IIIs) will translate the RTS signal from one RS-232 port to a DCD signal on the other port. This allows the Apple IIIs to communicate using the Hardware handshake.

3. Interfaces provided

3.1 Program interface

All communication between a user program or language interpreter takes place through SOS System Calls. This section describes all SOS calls which are available in the RS-232 device driver. Refer to the SOS ERS Appendices for further information.

D_READ (dev_num:iv1, buf:ip, bytes:iv2, block_num:iv2, bytes_read:ov2)

This function performs a device read on the RS-232 port. Since this port is a character device, "block_num" is ignored. Characters are read from the input buffer contained within the device driver (255 character capacity) and moved into the user buffer which "buf" points to. The parameter "bytes" specifies the capacity of the user buffer. If the read was successful, the number of characters read will be returned in "bytes_read".

The system call GET_DEV_NUM may have to be performed if the value "dev_num" is not known.

D_WRITE (dev_num:iv1, buf:ip, bytes:iv2, block_num:iv2)

This function performs a device write on the RS-232 port. Characters to be written are moved from the user buffer, pointed to by "buf", to the output buffer contained within the device driver (255 character capacity).

D_STATUS (dev_num:iv1, status_code:iv1, status_list:op)

The Device Status function returns RS-232 port status information. The parameter "status_code" specifies the status function code to be performed. The following functions are available:

D_STATUS 0 : No Operation.

D_STATUS 1 : Retrieve device driver control parameters. Device driver control parameters are moved into the buffer "status_list". The first byte of the buffer must contain the number of bytes available for the status table control parameters. When the status call completes, the first byte will be replaced by the actual number of bytes used (fifteen).

There are 15, one byte control parameters. The first 12 have the same format as the device configuration block (see section 3.2 for description). The next parameter is "imed_read". A value of [80] hex indicates that immediate read mode is enabled. A zero value indicates that the mode is disabled. The next parameter is "ACIA_status". This parameter contains the ACIA (Asynchronous Communication Interface Adapter) status from the last interrupt processed. The bits of this parameter have the following meanings:

bit 0=1 for parity error, 0 for no error
bit 1=1 for framing error, 0 for no error

bit 2=1 for overrun has occurred, 0 for no overrun
bit 3=1 for receiver data register full,
0 for not full
bit 4=1 for transmitter data register empty,
0 for not empty

bit 5=1 for Data Carrier Detect false, 0 for true
bit 6=1 for Data Set Ready false, 0 for true
bit 7=1 for interrupt has occurred, 0 for no interrupt

The last parameter is "status_latch" and contains selected status bits which are "latched" since the last D_STATUS 1 or D_CONTROL 0 request.

In other words, if any parity error occured since the last D_STATUS 1 or D_CONTROL 0 request, the parity error bit would still be equal to one. The bits of this parameter have the following meanings:

bit 0=1 for parity error, 0 for no error
bit 1=1 for framing error, 0 for no error
bit 2=1 for overrun has occurred, 0 for no overrun
bit 5=1 for Data Carrier Detect false, 0 for true
bit 6=1 for Data Set Ready false, 0 for true
bit 7=1 for input character lost due to full buffer

D_STATUS 2 : Get newline character. The "is_newline" and "newline" bytes are returned in the buffer "status_list".

D_STATUS 3 : Retrieve driver buffer information. Eight bytes of buffer information are returned in "status_list".

Byte	Description
1-2	Size of output buffer [low byte | high byte]
3-4	Number of bytes in output buffer [low byte | high byte]
5-6	Size of input buffer [low byte | high byte]
7-8	Number of bytes in input buffer [low byte | high byte]

D_CONTROL (dev_num:iv1, control_code:iv1, control_list:ip)

The Device Control function specifies which control function is to be performed on the device driver. The parameter "control_code" selects the control function desired. The following functions are available:

D_CONTROL 0 : Resets the device handler and clears the buffers. This function will clear the driver output buffer immediatly. Any characters not yet transmitted will be lost. The current control parameter values are used for the reset.

D_CONTROL 1 : Load device control parameters into the driver from the buffer "control_list". Control parameters are in the same format as "D_STATUS 1". The first byte of the buffer must contain the exact number of bytes as set by status request-1 (fifteen). The last parameters ("ACIA_status" and "status_latch") are not loaded from "control_list". A "D_CONTROL 0" is performed in order to reset the control parameters.

D_CONTROL 2 : Load newline list. The "is_newline" and "newline" bytes are loaded into the driver from the buffer "control_list".

D_CONTROL 3 : Transmit Break. This control function will cause the transmit line of the RS-232 port to go to a zero state or "SPACE" condition for a specified number of 233 millisecond intervals. A one byte "control_list" specifies the number of intervals the "break" will remain on for. This value parameter can range from 1 to 100 (maximum "break" of 23.3 seconds). Normally this value should be equal to one. A "break" signal will not be transmitted until the driver output buffer has been emptied.

OPEN (pathname:ip, ref_num:ov1, open_list:ip, length:iv1)

The OPEN function must be performed before any other function can be used. The device control parameters are setup based on the device configuration block values (set by SCP). The "is_newline", "newline" and "imed_read" control parameters are set to zero.

The parameter "pathname" must conform to the SOS pathname conventions as outlined in the SOS ERS Appendices. A valid example of a pathname for this device is ".RS232". The value "ref_num" must be saved for use in the CLOSE. The remaining parameters, "open_list" and "length", are not used with this device.

CLOSE (ref_num:iv1)

This function closes the device. The value of "ref_num" returned during the OPEN must be used here. The CLOSE function will wait until the driver output buffer has been emptied before control is returned to the user.

3.2 System Configuration Program interface

The System Configuration Program (SCP) is used to both install the RS-232 driver into the SOS.DRIVER file of the boot diskette and modify the parameters in the device configuration block of the driver. The Standard Device Drivers manual contains a complete description of how to use SCP.

The device configuration block (DCB) of this driver contains twelve parameters. These twelve bytes may be modified with SCP to reconfigure the device characteristics of the RS-232 port.

The following paragraphs explain the function of each byte of the DCB.

Byte 0 [06] : Data rate. This value controls the RS-232 port communications speed. There are nine possible speeds (baud is the unit of measure):

Value	Speed
03	110 baud
04	134.5 baud
06	300 baud
07	600 baud
08	1200 baud
09	1800 baud
0A	2400 baud
0C	4800 baud
0E	9600 baud

Byte 1 [22] : Data format. This value determines the arrangement of bits in the data stream of the RS-232 port. There are nine standard formats.

Value	Format
22	7 bits, odd parity
26	7 bits, even parity
2A	7 bits, mark parity
2E	7 bits, space parity
00	8 bits, no parity
42	6 bits, odd parity
46	6 bits, even parity
4A	6 bits, mark parity
4E	6 bits, space pariy

Bytes 2-4 [00] [00] [00] : Delay counts.

These values determine the amount of time, measured in character times, that the driver is to wait after it sends either a carriage return character (ASCII CR, value 13), a line feed character (ASCII LF, value 10) or a form feed character (ASCII FF, value 12). This delay would allow a hard copy device, attached to the RS-232 port, time to position the printing element to the next print position after it receives a carriage return, line feed or form feed.

The normal delay amount is zero, which produces no delay. The delay amount can range from 0 to 255 and must be specified in hexadecimal (base 16).

Byte 2 is the carriage return delay, byte 3 is the line feed delay and byte 4 is the form feed delay.

Byte 5 [00] : Communications Protocol.

Bit 7 specifies whether the XON/XOFF communications protocol mode is set. A value of [80] enables XON/XOFF.

Bit 6 specifies whether the ENQ/ACK communications protocol mode is set. A value of [40] enables ENQ/ACK. Only values of [00], [80], or [40] are valid. Bits 7 and 6 should not both be set.

Byte 6 [13] : Control Character One.

If the XON/XOFF communications protocol is to be used, this byte should contain the XOFF character (ASCII DC3 - [13] hex). If the ENQ/ACK communications protocol is to be used, this byte should contain the ENQ character (ASCII ENQ - [05] hex). If neither protocol is enabled, this parameter is ignored.

Byte 7 [11] : Control Character Two.

If the XON/XOFF communications protocol is to be used, this byte should contain the XON character (ASCII DC1 - [11] hex). If the ENQ/ACK communications protocol is to be used, this byte should contain the ACK character (ASCII ACK - [06] hex). If neither protocol is enabled, this parameter is ignored.

Byte 8 [DF] : Maximum Buffer Level.

This value, 223 decimal or [DF] hexadecimal,
is used only in XON/XOFF protocol mode. It specifies the input buffer level which when reached will cause the driver to transmit the XOFF character. Since the input buffer can hold 255 characters, there will be room for 32 more characters in the buffer when XOFF is sent. This reserve amount allows the sender some time to respond to the XOFF before suspending transmission to the Apple.

Byte 9 [84] : Minimum Buffer Level.

This value, 132 decimal or [84] hexadecimal, is used only in XON/XOFF protocol mode. When the input buffer count is less than this value, the XON character is sent from the Apple, signaling the sender to resume transmission to the Apple.

Byte A [50] : Data Block Length.

This value, 80 decimal or [50] hexadecimal, specifies the number of characters that can be transmitted (or received) between ENQ/ACK handshakes. This parameter may be set to any value from 1 to 255 ([01] to [FF] - hex) allowing the RS-232 driver to work with devices which use block sizes other than 80. This parameter is only used in ENQ/ACK communications protocol mode.

Byte B [00] : Hardware Handshake Mode. Bit 7 specifies whether hardware handshake mode is enabled. Valid values for this parameter are [80], hardware handshake mode enabled, and [00], hardware handshake mode disabled. If this mode is specified (byte B set to [80]) then byte 5 should be set to [00] (no communications protocol).

Appendices

A. Staging plan for future releases

If and when a Universal Serial Card for the APPLE III is developed, this driver could be adapted to handle one or more units within the device driver itself.

An alternate approach would be to develop a separate driver for a Serial Card and utilize much of the code and functionality of this driver.

B. Estimated resource requirements

B.1 Staffing requirements

LAB - One person full time for 6-9 months

NPR - One person for 3-5 months

PUBs - One person for 2-3 months

B.2 Equipment requirements

Apple III - 2 Disk Drives, 12" Monitor

Apple II - 3 Disk Drives with SOROC terminal and communications interface.

Pair of Bell compatible 103-Type (300 baud) modems. *

Pair of Bell compatible 212-Type (1200/300 baud) modems. *

Access to pair of telephones for modem attachment.

Break-out box and RS-232 cables.

DEC VT-100 *

DECWRITER III *

HP 2645 **

* This equipment can be borrowed from the MIS group for testing.

** These terminals will be needed for testing, since support of their technical features is desired.

B.3 Testing requirements

Possible rental of terminals for testing (HP 2645). Possible rental of computer time for testing on data communication networks (Telenet and Tymnet) and data banks (Dow Jones, Source, etc.).

B.4 Publication requirements

Either a separate manual, similar to the Apple III - Standard Device Drivers manual or another chapter in that manual.

B.5 Critical dependencies

No special dependencies exist at this time.

C. Design history

C.1 Preliminary ERS, Version 1.0 (original document 16-DEC-80)

Final ERS, Version 1.1 (9-FEB-81)

Final ERS, Version 1.2 (4-MAR-81)

###