MCM COMPUTERS LIMITED

COMMUNICATIONS SUBSYSTEM

REFERENCE MANUAL

HEAD OFFICE

MCM COMPUTERS LTD.
6700 Finch Avenue West
Suite 600
Rexdale, Ontario
M9W 5P5

MANUFACTURING FACILITY

MCM COMPUTERS LTD.
133 Dalton Street
Kingston, Ontario
K7L 4W2

To order the Communications Subsystem Reference Manual, use the number below:

Manual No. 0180023

Revision No. AB

Date March 1979

TABLE OF CONTENTS

APPENDICES

1.0 GENERAL DESCRIPTION

1.1 INTRODUCTION

This subsystem is responsible for handling all explicit (that is, Quad or Quad Quote) I/O for the *APL* system. It contains built in drivers for a DIABLO or QUME printer, and an EIA asynchronous communications interface. Both drivers are controlled by user-accessible tables which reside in the *APL* workspace, and can be modified to support almost any EIA compatible asynchronous device, including ASCII and IBM 2741 devices, with both *APL* and non-*APL* character sets. Communication with other computer systems is also possible. Synchronous communication protocol is not supported.

1.2 DIVERTED I/O

For the purposes of this description, SINK is the logical device to which output is sent, and SOURCE is the logical device from which input is obtained.

In *MCM/APL*, the following types of output can occur:

1. Data assigned to Quad (Quad Output)
2. Quad input prompt
3. Quad prime input prompt
4. Data left unassigned at the end of a line of immediate execution or user defined function (Implicit Output)
5. Error Messages
6. Function editing display

Normally, all output is directed to the MASTER SINK on the MCM/900, (the built-in CRT display). However, it is possible to divert output of the first type (that is, Quad Output), to a device attached to the MCM/900's Omniport. A device to which output is so directed is called a DIVERTED SINK.

Similarly, the following types of input can occur:

1. Literal (Quad-Prime) Input
2. Quad Input
3. Input requested by the system when no function is active (Implicit Input)
4. Function editing Input

Normally, all input is obtained from the MASTER SOURCE (on the MCM/900, the built-in keyboard). However, it is also possible to divert input of the first type (that is Quad-Prime Input), to a device attached to the Omniport (which becomes a diverted source).

In addition, if the device is known to the system to be able to accept prompts, the prompt normally output on the built-in display will be diverted to this device.

The diversion of I/O is accomplished with the system functions ⎕*IN* and ⎕*OU*, (see the description of the Direct Device Access Functions Sections 2.7 - 2.9). In general, ⎕*IN A* (or ⎕*OU A*) causes the device whose omniport address is *A* to become the current source (or SINK) device. Normal devices may have any address in the range 1 through 199. Address 0 refers to the MASTER device, and addresses above 199 are reserved for special purposes. The address of a built in RS232 is set to 102.

N.B. The system is not able to cope with the situation where two devices on the omniport have the same address.

1.3 I/O FORMATTING

All diverted output is formatted, and all diverted input is de-formatted.

1.3.1 RULES FOR OUTPUT

1. All numeric output is converted to character form. That is, in all cases, the expression ⎕←*X* is equivalent to the expression ⎕←⍕*X*, except that the former takes less time and space.
2. Boundaries between subarrays of rank ≥2 when output are marked by a single blank line. (See examples in Appendix).
3. Trailing space truncation may be enabled or disabled. If disabled, no special action is taken. If enabled, trailing spaces on each line are removed before further output formatting.

4. Characters whose output code is MNEMONIC (¯1; see DEVICE CONTROL), are converted to their equivalent mnemonic form (see appendix). If no such representation exists, or if the code for the escape character ("$") is MNEMONIC, a *COMM TABLE ERROR* will result.

5. Lines whose width after mnemonic conversion exceeds the width declared for the device are broken at the right-most space, left parenthesis, or right parenthesis to the left of the right margin of the device. If no such character exists, the line is broken at the right margin. Lines broken in this fashion have a Continuation Character (see Continuation, Page 11) appended to them, and the next line is preceded with the continuation indent declared for the device.

   NOTE: The maximum legal device width is 133.

6. Each character is translated to its declared output code before transmission. Legal output codes are 0 through 127, MNEMONIC (¯1) and OVERSTRIKE (¯2). Characters whose output code is OVERSTRIKE are transformed to their equivalent overstrike form (see Appendix), and the resulting triplet is re-translated prior to output. If any of the resulting triplet translates to a code outside the range 0 to 127, a *COMM TABLE ERROR* will result.

7. Each physical line output causes the device line counter to be incremented (see Form Feed, page 11). If this causes line counter to reach the page size declared for the device, the line counter is reset to 0 and the page counter is incremented.

1.3.2 RULES FOR INPUT

1. Each character received is translated to its declared system character code before further processing.

2. Visual fidelity is maintained. This means that if the input device has an integral display, the line returned to the system is that which appears on that display at the time that the input operation is terminated. Devices without an integral display are treated as if they had one.

3. The rightmost character in the line corresponds to the rightmost character (including spaces) input from the device, subject to margin restrictions.

4. Backspaces received at the left margin are ignored. Non-control characters received at the right margin are ignored, and the rightmost valid character is replaced with the system canonical Bad Character. The right margin is fixed at 133.

5. Overstriking may be enabled or disabled (see ROW 0, page 10). If disabled, a character received at a position containing another character replaces that character.

   If overstriking is enabled, the following rules apply:

   a) Any non-control character replaces a blank

   b) Blanks replace no other character

   c) Any character replaces itself

   d) Non-blank, non-control characters form their appropriate overstrike (see appendix). If no such valid overstrike exists, the result is the system canonical Bad Character.

6. Idle characters are ignored. Input Codes which do not translate to recognizable characters are replaced by the system canonical Bad Character.

7. An END-OF TRANSMISSION character received from a prompted device causes the line to be truncated immediately to the left of the current position. The remainder is re-transmitted on a new line as a prompt for further input. For non-prompted devices, all input prior to the EOT is discarded.

8. A NEWLINE character terminates the input operation.

9. Mnemonic processing may be enabled or disabled (see DEVICE CONTROL). If disabled, no special action is taken. If enabled, an escape character ("$") followed by two characters causes them to be converted to the appropriate character. If no such valid mnemonic exists, the result is the system canonical bad character.

10. If, after mnemonic processing, a continuation character is found at the right end of the line, the system returns to the input device for the continuation (NOTE: At this time, the right margin position is reduced by the logical length of the previous accumulated input.) Any input may consist of an arbitrary number of continuations.

11. If a prompt is provided for the input, and the device is capable of accepting it, it is transmitted to the device according to the rules for output, and the device is left positioned at the logical position specified for the operation (i.e. the left argument to ⍞, if one exists, or else at the right end of the output). If the prompt was broken (see Output rule 5) at a point to the right of the requested position, the device is left positioned at the left margin. The prompt is then treated as if it were input from the device (see examples in Appendix).

1.4 DEVICE CONTROL

Device Control is achieved via two sets of tables, one each for the SOURCE and SINK devices. These tables reside in the workspace, and may be modified by the user. For information on table access, see the description of the Table Access System Functions, Section 2.11. The MASTER DEVICES do not use these tables.

Each table consists of three segments: a CONTROL segment, an INPUT segment, and an OUTPUT segment. The INPUT and OUTPUT segments are simply translate tables. Each output character is translated to the corresponding code in the OUTPUT segment. Permissible codes are 0 through 127, ¯1 and ¯2. Non-negative codes are output verbatim (except for Shift Control; see EIA INTERFACE). A code of ¯1 indicates that this character has no direct representation on the device, and must be output as a MNEMONIC (see Output Format rule 3). A code of ¯2 means that this character must be output as an OVERSTRIKE (see Output rule 5).

Each input code is translated to the corresponding character in the INPUT segment. Control characters are represented by character values less than zero. Each control character corresponds to a row in the CONTROL segment. The number of the row referred to is the absolute value of the control character.

The Control segment is used to specify properties of the device which cannot be specified in either the INPUT or the OUTPUT segment.

Control table rows are numbered starting at 0, and are of varying length. Each row except 0 refers to action to be taken in transmitting a control character. In most cases, this information is peculiar to the device in question, and is discussed there. However, certain elements are used by the system as a whole. The structure of the control table is as follows:

ROW	LENGTH	USAGE
0	4	General Device Information
1	3	Continuation
2	4	Newline
3	1	Backspace
4	1	Idle
5	3	Carriage Return
6	1	Form Feed
7	1	Shift Down
8	1	Shift Up
9	2	End of Transmission
10	2	Beginning of Transmission

0 - GENERAL DEVICE INFORMATION

The value of the first element of row 0 is:

(32×*PROMPT*) + 2 ⊥ *TRUNCATE*, *OVERSTRIKE*, *MNEMONIC*

WHERE:

PROMPT	1 if input prompts are to be issued to the device.
TRUNCATE	1 if trailing spaces are to be truncated on output.
OVERSTRIKE	1 if overstriking is enabled on input.
MNEMONIC	1 if Mnemonics are enabled on input.

The definition of the remaining 3 elements is dependent on the type of device being used. See the description of the device in question for definition (EIA - Section 3, Hytype - Section 4, Centronics - Section 5).

1 - CONTINUATION

The structure of row 1 is:

WIDTH, CHAR, INDENT

WHERE:

WIDTH is the physical width to be used for the device (maximun 133).

CHAR is the code for the continuation character: 0 through 127 means an actual device code; 128 means no character; 129 means mnemonic; 130 means overstrike.

INDENT is the number of spaces which precede the text of continuation lines (maximum 15).

6 - FORM FEED

Row 6 is used for paging control. The first element is the number of physical lines to be printed on a page. When this limit is reached the number of Form Feed characters specified by the second row element is issued, and the line counter is reset to zero. If the limit is 0, page control is disabled.

The remaining rows are unique to the application and are described in the following sections.

I/O INTERRUPTION

An I/O operation may always be interrupted with a HARD INTERRUPT on the main keyboard (*CONTROL*, *SHIFT*, '→'). However this may leave the I/O device in an unknown state, from which it must be reset by the user prior to any further usage of the device. For this reason, interruption in this fashion should be avoided wherever possible.

2.0 SYSTEM FUNCTIONS

The Communications Subsystem contains the following system functions:

2.1 CHARACTER I/O

⎕ ← *X*	OUTPUT
AV ← *SI* ⍞ *AV*	INPUT

2.2 OMINPORT ACCESS

NIV ← ⎕*IN* *NIV*	Set SOURCE
NIV ← ⎕*OU* *NIV*	Set SINK
NIS ← ⎕*BI* ⍳0	Direct SOURCE input
NIS ← ⎕*BI* *NIS*	Direct SOURCE output
NIS ← ⎕*BO* ⍳0	Direct SINK input
NIS ← ⎕*BO* *NIS*	Direct SINK output
NIS ← ⎕*YA* *NIV*	Locate device ADDRESS

2.3 TABLE ACCESS

YD ← *YD* ⎕*YI*[*SI*]*YR*	Access SOURCE table segment
YD ← *YD* ⎕*YO*[*SI*]*YR*	Access SINK table segment
NIS ← *TS* ⎕*YR* *NAME*	READ Complete Table
NIS ← *TS* ⎕*YW* *NAME*	WRITE Complete Table
TS ⎕*YX* ⍳0	EXPUNGE Complete Table

2.4 MISCELLANEOUS

X ← ⎕*Y X*	Convert Character/Numeric
NS ← ⎕*DL NS*	Delay *NS* Seconds
NIV ← ⎕*PC NIV*	Read and Set Print Counters

WHERE:

X may be of any shape or type

NS is a numeric scalar

NIS is an integer scalar

NIV is an integer vector

AV is a character vector

SI is a scalar index

YR is a table segment reference array

YD is a table segment data array

TS is "*I*" or "*O*"

NAME is a character vector which represents the name of an *APL* variable.

.. indicates that the enclosed item may or may not be present.

Notes on the following descriptions:

1) Wherever conformability requires a one-element vector, a scalar is acceptable.

2) The symbol "↔" means "is identical to".

2.5 CHARACTER OUTPUT

SYNTAX

R←⎕←B

DOMAIN

B may be character or numeric

CONFORMABILITY

No restriction

RESULT

If a result is requested, B is returned as R.

OPERATION

B is formatted and output to the current Sink device according to the Output Formatting rules (Section 1.3.1).

2.6 CHARACTER INPUT

SYNTAX

R←A⍞B

DOMAIN

B must be character
A must be an index ≤ ⎕IO+132

CONFORMABILITY

0 = ρρA
1≥ρρB
(×/ρB)≤ 132

RESULT

R is character vector
(PR) ≤ 132

OPERATION

If the current SOURCE device is to be prompted, B is output to the device according to the Output Formatting rules (Section 1.3.1). If A is present, the cursor (or carriage) is left positioned at the logical position of the character B[A] (note: origin dependence). If (×/ρB)<1+A-⎕IO, B is extended on the right with spaces. If A is absent, its value is assumed to be (×/ρB)+⎕IO-1.

If the current SOURCE device is not interactive, no prompt is output. A line of input is then obtained from the device, and any prompt output is treated as if it were part of the input. The input is processed according to the Input Formatting Rules, and the resulting character vector is returned as R.

2.7 DIRECT DEVICE ACCESS

Basically, six operations are available:

1) Address device

2) Read device code

3) Send command to device

4) Read device status

5) Send data to device

6) Read device data

The function of these operations is achieved with four *APL* system functions - ⎕*OU*, ⎕*IN*, ⎕*BO*, ⎕*BI*. All operations cause the device in question to be addressed, and its Answer-Back Code to be obtained. The Answer-Back Code is hard wired 8-bit response which returns information about the class of the device. The Answer-Back Code of a device is made up as follows: Let *ABC* be the code received via ⎕*IN* or ⎕*OU* (See Section 2.8), and suppose we set:

C ← 2 2 2 32 ⊤ *ABC*

Then the elements of *C* are interpreted as follows:

ELEMENT	*ABC* BITS	MEANING
C[1]	7	1 = Input Device
C[2]	6	1 = Output Device
C[3]	5	1 = Prompt valid for Input
C[4]	4 - 0	Device Type

The following device types have so far been assigned:

TYPES	DEVICE
1	MCM/EIA Interface (RS232C)
2	MCP-132 Printer
3	PMR-200 Card Reader
4	ATU-100 External Cassette Drive
5	DDS-500 or 1000 Diskette Drive
6	MCP-713, 712, 709 etc. Printers
7	CRT Display

In communicating with the Omniport, the 8 bits of the Omniport are represented in the system by their base 2 value. Bit 7 on the Omniport is the most significant bit.

2.8 SYSTEM FUNCTIONS ⎕OU, ⎕IN

These functions are identical in operation except that ⎕OU (short for ⎕OUT) sets the system SINK, while ⎕IN sets the system SOURCE.

SYNTAX

R ← ⎕*IN B* (Source)
R ← ⎕*OU B* (Sink)

CONFORMABILITY

1 = ρρ*B*
2 ≥ ρ*B*

RESULT

1 = ρρ*R*
3 = ρ*R*

OPERATION

The current SOURCE/SINK is set to the device whose interface address is *B*[1]. If *B* is empty, the SOURCE/SINK device is left unchanged. The device is selected, and the answer-back code from it is returned in *R*[2]. *R*[1] contains the device address. *R*[3] contains the current device status. If *B*[2] is present, it is output as a command to the device.

If the addressed device is not present, the answer-back code, status, and data (received via ⎕*BI* or ⎕*BO*, see below), will all be zero.

The command status codes are peculiar to the device being accessed. Refer to the documentation for the device in question. The sequence of events is:

1. Device Addressed
2. Answer-back code obtained
3. Command (if any) issued
4. Status obtained
5. Device de-addressed

2.9 SYSTEM FUNCTIONS ⎕BO, ⎕BI

These functions are used to output and input data via the omniport in binary form directly to or from the device currently addressed.

SYNTAX

R ← ⎕*BI B* or ⎕*BO B*

CONFORMABILITY

0 = ρρ*B* or 0↔ρ*B*

DOMAIN

The elements of *B* must be integers in the range 0 to 255.

RESULT — $0 = \rho\rho R$ if B is empty, otherwise $\rho R \leftrightarrow \rho B$

OPERATION — If B is empty, the operation is a request for input, and R is input data from the current SOURCE/SINK device.

Otherwise B must be a scalar, representing data which is output to the current SOURCE/SINK device. In this case, if a result is requested, B is returned as R.

The sequence of events in both cases is:

1. Device Addressed
2. Answer-back obtained
3. Data transferred
4. Device de-addressed

2.10 DEVICE ADDRESS LOCATION

SYNTAX — R← ⎕YA B

DOMAIN — B must be numeric, in the range 0 to 255.

CONFORMABILITY — $1 = \rho\rho B$

$(1 \leq \rho B) \wedge (3 \geq \rho B)$

RESULT — $0 = \rho\rho R$

R is the address of the device located, or 0 if no device was found.

OPERATION — $1 \leq \rho B$:

Only devices whose answer-back code is B[⎕IO] are examined.

$2 \leq \rho B$:

The answer-back code ABC is masked as follows before being compared to B[⎕IO].

ABC ← 2⊥((8ρ2)⊤B[1+⎕IO]) ∧ (8ρ2)⊤ABC

$3 = \rho B$:

Only devices with addresses B[2+⎕IO] or higher are examined.

2.11 DEVICE TABLE SEGMENT REFERENCE

SYNTAX

R←⎕*YI*[*I*]*B* (SOURCE)
R←⎕*YO*[*I*]*B* (SINK)

OPERATION

The SOURCE device table is accessed with ⎕*YI*. The SINK device table is accessed with ⎕*YO*. The segment requested is indicated by the origin-dependent index *I* as follows:

```
            0 - CONTROL segment
I ← ⎕IO+    1 - INPUT translate segment
            2 - OUTPUT translate segment
```

2.11.1 CONTROL SEGMENT (*I*=⎕*IO*)

DOMAIN

Elements of *B* must be integers in the range 0 to 10

CONFORMABILITY

1≥ρρ*B*

RESULT

(ρ*R*) ↔ (ρ*B*), ⌈/*N* [*B*+⎕*IO*]

Where *N* is a vector of control row lengths such that for any row *J*, *N*[*J*] is the length of row *J*. That is, since the rows of the Control Table are not all the same length, it is necessary to pick a result row length which is large enough to accommodate the largest of the control segment rows indicated by *B*.

The result *R* is integer.

OPERATION

Each row of *R* is the row of the control table indicated by the corresponding element of *B*. Short rows are padded on the right with zeroes.

NOTE: *B* is not origin dependent.

2.11.2 INPUT TRANSLATE SEGMENT (*I*=⎕*IO*+1)

DOMAIN

The elements of *B* must be integers in the range ¯10 to 127.

CONFORMABILITY

1≥ρρB

RESULT

(ρ*R*)=ρ*B*
R is integer.

OPERATION

Each element of *R* is the system code value to which the device code represented by the corresponding element of *B* translates on input.

NOTE: For control characters, the system code value is a negative number whose absolute value is the corresponding row number of the control segment (Section 1.4). For non-control characters, the code value is that given by the conversion function ⎕Y (Section 2.4).

2.11.3 OUTPUT TRANSLATE SEGMENT (I=⎕IO+2)

DOMAIN — B must be character

CONFORMABILITY — 1≥ρρB

RESULT — (ρR)=ρρB
R is integer

OPERATION — Each element of R is the device code value to which the corresponding character in B translates on output.

If the character is to be represented as a mnemonic, the translate code is ¯1. The translate code for overstrike representation is ¯2.

The tables of valid mnemonics and overstrikes is built into ROM and cannot be altered by the user.

2.12 DEVICE TABLE SEGMENT MODIFICATION

SYNTAX — R←A ⎕YI[I]B (SOURCE)
R←A ⎕YO[I]B (SINK)

RESULT — If a result is requested, A is returned as R. Refer to the description of the monadic forms of these functions.

2.12.1 CONTROL SEGMENT (I=⎕IO)

DOMAIN — The elements of A must be integers in the range 0 to 255.

CONFORMABILITY — (ρA)=(ρB),⌈/N[B+⎕IO]
where - N is as for the monadic form.

OPERATION — For each element J of B, row J of the control table is replaced with the first N[J+⎕IO] elements of the corresponding row of A. The remaining elements in the row are ignored. If B contains duplicates, the operation is not defined.

2.12.2 INPUT TRANSLATE SEGMENT (*I*=⎕*IO*+1)

DOMAIN — The elements of *A* must be integers in the range ¯10 to 127. Values other than system character code values or control code values are assumed to be the system canonical bad character code (108).

2.12.3 OUTPUT TRANSLATE SEGMENT (*I*=⎕*IO*+2)

DOMAIN — The elements of *A* must be integers in the range ¯2 to 127.

2.12.4 INPUT and OUTPUT TRANSLATE SEGMENTS

CONFORMABILITY: — (ρ*A*)=ρ*B* or (ρρ*A*)=0

OPERATION — The translate value indicated by each element of *B* is set to the corresponding element of *A*. If *A* is a scalar, it is extended. If *B* contains duplicates, the operation is not defined.

2.13 DEVICE TABLE READ

SYNTAX — *R*←*A* ⎕*YR* *B*

DOMAIN — *B* must be a character vector representing a valid *APL* name. *A* must be the character scalar '*I*' or '*O*'.

RESULT — *R* is an integer scalar.

OPERATION — The variable named in *B* is replaced with the complete contents of the device table indicated by *A*, ('*I*' for the SOURCE table, '*O*' for the SINK table.

If *B* does not name a variable or an undefined object, a *RANGE ERROR* is issued.

NOTE: The type of the resulting data specified to the variable is such that no function other than *NULL* (∘) will accept it as valid data. Any attempt to do so will result in a *DOMAIN ERROR*. ⎕*NC* returns a 5 for this data.

	The result is the device answer-back code associated with the table at the time it was created. This code is the device answer-back code with either the OUTPUT or the INPUT and PROMPT bits masked off. The OUTPUT bit is masked off for the SOURCE table, and the INPUT and PROMPT bits are masked off for the SINK table.

2.14 DEVICE TABLE WRITE

SYNTAX	*R* ← *A* ⎕*YW* *B*
DOMAIN	*B* must be a character vector representing a valid *APL* name. *A* must be the character scalar '*I*' or'*O*'.
RESULT	*R* is an integer scalar.
OPERATION	As for ⎕*YR*, except that a device table is created with the data in the variable named by *B*. If *B* was not created by ⎕*YR*, a *RANGE ERROR* is issued. The result is the same as for ⎕*YR*.

2.15 DEVICE TABLE EXPUNGE

SYNTAX	*A* ⎕*YX* *B*
DOMAIN	*B*←→⍳0 *A* must be the character scalar '*I*' or '*O*'.
OPERATION	The table indicated by *A* is unconditionally destroyed.

2.16 CHARACTER/NUMERIC CONVERSION

SYNTAX	*R*←⎕*Y* *B*
DOMAIN	*B* may be character or numeric. If *B* is numeric it must be integer data in the range 0 to 255.
CONFORMABILITY	No restriction.
RESULT	(⍴*R*)←→⍴*B*

OPERATION:	If B is character, R is numeric. Each element of R is the value of the system code for the corresponding character in B.

If B is numeric, R is character. Each element of R is the character for which the corresponding element of B is the system code value. Codes which do not correspond to valid system characters are mapped into the system canonical bad character (Code 108).

Control characters are not valid system characters.

See Appendix for system code values.

2.17 DELAY

SYNTAX: *R*←⎕*DL B*

DOMAIN: B must be numeric, in the range 0 to 25.5

CONFORMABILITY: 0=ρρ*B*

RESULT: $R = B$

OPERATION: A delay of at least B seconds occurs before the function returns its result. In most cases, the delay will be approximately equal to B. However, the delay timing is suspended while the control key on the main keyboard is held down. Timing resolution is 0.1 sec.

2.18 PRINT COUNTERS

SYNTAX: *R*←⎕*PC B*

DOMAIN: B must be integer in the range 0 to 255.

CONFORMABILITY: 1=ρρ*B*
2≥ρ*B*

RESULT: 2↔ρ*R*

OPERATION: The print counters form a two-element vector. The first element is a page counter, and the second is a line counter.

Each time a physical line is output to the current SINK DEVICE (if output is diverted), or to the current SOURCE DEVICE, if its address is the same as that of the current SINK, the line counter is incremented.

Whenever the end of a page is reached (or whenever the line counter reaches 256, if paging is off), the line counter is reset to zero, and the page counter is incremented. The page counter is reset when it reaches 256.

If *B* is two elements, they are used to set the current page and line counter. If *B* is one element, the page counter only is set. If *B* is empty, the counters are unaffected.

The result is the page and line counter after setting according to *B*.

3.0 EIA INTERFACE

This interface, consisting of a device driver in the communications subsystem and an omniport EIA board, allows the user to communicate with almost any asynchronous device which is compatible with the EIA RS-232-C specification. A teletype current loop is also provided.

The standard tables provided by the system allows communication with a 300 baud ASCII half-duplex terminal using the APL/ASCII typewriter pairing overlay character set. In order to divert output to such a device, the user need simply connect the device to the system and type:

⎕*OU* ⎕*YA* 65 95

or for input:

⎕*OU* ⎕*YA* 129 159

The following documentation describes handling of EIA devices in a non-standard fashion:

3.1 DIRECT ACCESS

Before any communication can occur, the omniport EIA board must be informed of the protocol to be used for data transfer. This is achieved with the command byte. The least significant 6 bits of the command byte are control data. The most significant two bits of the command byte indicate the usage of the control data:

0(00) - General control

1(01) - Serial word control

2(10) - Least significant data rate bits

3(11) - Most significant data rate bits

- Bits 0-6 Command Data

3.1.1 DEVICE STATUS BYTE

Bit 7 - Read Overrun

Bit 6 - Read Parity Error

Bit 5 - Read Framing Error

Bit 4 - Device Powered and Ready

Bit 3 - Receive Carrier Off

Bit 2 - Break Received

Bit 1 - Transmit Buffer Empty

Bit 0 - Receive Data Available

Bits 3 and 4 are static conditions. All other bits are set and reset by events.

Bit 0 is set when a complete serial data word has been received, and reset when the resulting data byte is read.

Bit 1 is set when the transmit buffer is ready to accept a byte for transmission, and reset when a data byte is output to it. If the interface is connected to a modem, this bit is forced low wherever the CLEAR TO SEND line from the modem is false.

Bit 2 is set when the RECEIVE DATA line stays in the SPACE condition for a time determined by a setting on the board. This time is factory set to 150 msec., and may be set anywhere in the range 15 to 200 msec.

Bit 3 is true when the EIA interface is connected to a modem, and the modem does not detect a receive carrier. When the interface is connected to a terminal, this bit is tied to bit 4.

Bit 4 is true whenever the interface is powered and is receiving either a DATA SET READY (from a modem or DATA TERMINAL READY (from a terminal).

Bit 5 is set whenever the line is in a SPACE condition at the STOP position of a received serial data word.

Bit 6 is set whenever the parity of the received serial data word is incorrect.

Bit 7 is set whenever a serial data word is received with Bit 0 true. If this happens, the data in the buffer is replaced with the new data.

3.1.2 COMMAND 0 - GENERAL CONTROL

Only the least significant 3 data bits are defined. They are used as follows:

Bit 0 - Master Reset. Resets all latches.

Bit 1 - Event reset. Resets status bits 0, 1, 2, 5, 6 & 7.

Bit 2 - Transmit Break. This sets the TRANSMIT DATA line in a SPACE condition for a time determined by a setting on the board. This time is factory set to 200 msec, and may be varied from 15 to 200 msec.

EXAMPLE:

To transmit a BREAK, execute the statement:

```
⎕OU (1↑⎕OU⍳0),4
```

To execute a MASTER reset execute:

```
⎕OU (1↑⎕OU⍳0),1
```

3.1.3 COMMAND 1 - SERIAL WORD CONTROL

The least significant 6 data bits are used as follows:

Bits 1 & 0	The number of data bits in a serial word, minus 5. Thus a 6-bit data word is indicated by 01.
Bit 2	If 1, indicates even parity.
Bit 3	If 1, indicates no parity bit (in this case, bit 2 must be set to zero.
Bit 4	If 1, indicates two stop bits (one is normal).
Bit 5	If 1, indicates that the REQUEST TO SEND line to the device is to be set true.

3.1.4 COMMANDS 2 & 3 - DATA RATE

The data rate is determined from the Baud Rate B as follows:

```
R ← ¯1+⌊+.5+25000÷B
```

This is then represented as an 8 bit binary word. The least significant 4 bits are output with command 2, and the most significant 4 bits are output with command 3:

```
ADDR← 1↑⎕OUι0

X ← 16 16 ⊤ R

⎕OU ADDR, 8 64 ⊥ 2,¯1↑X

⎕OU ADDR, 8 64 ⊥ 3,1↑X
```

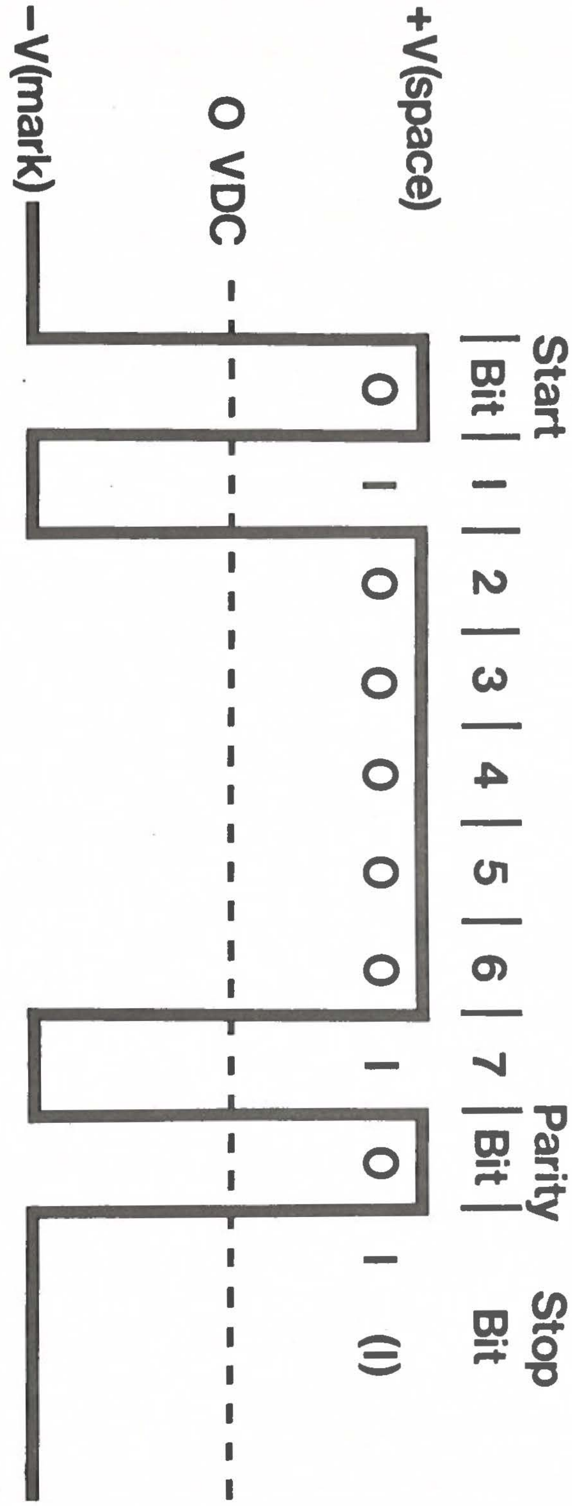
3.1.5 DATA TRANSFER

The most recently received data byte may be read with ⎕*BI*ι0 (or ⎕*BO*ι0).

A data byte *B* may be transmitted with ⎕*BO B* (or ⎕*BI B*).

Following is a time diagram of a serial data word. The least significant data bit is the first bit in the string. The parity bit is last.

Timing Diagram of Serial Data Word



RS-232 C Encoded ASCII "A"

3.2 DEVICE TABLES

Direct access to an EIA device is rather clumsy and usually unnecessary. Considerable flexibility may be obtained through modification of the device tables.

Devices with an *APL* Character Set whose codes differ from the standard set can be supported simply by specifying the appropriate codes in the INPUT and OUTPUT translate segments.

Devices with non-*APL* Character Sets (such as standard ASCII) can be supported by representing characters which the device does not have by mnemonics. Even a Teletype 33 or EIA compatible card reader can be supported in this fashion.

Protocol differences are handled via the CONTROL segment. The function of each row of the control table is described below in detail.

3.2.1 CONTROL SEGMENT FUNCTIONS

ROW 0 - GENERAL INFORMATION

This row has four elements, the first of which was described in the general documentation under Device Control (Section 1.4). The elements are:

XX,MISC,WORD,RATE

RATE The data rate for the device is set by *RATE*. The value of this element is identical to R as discussed under Data Rate in the discussion of Direct Access. The default is 82 (300 baud).

WORD The least significant 5 bits of the binary word represented by *WORD* are identical to the corresponding bits discussed under Serial Word Control. If bit 6 is 1, parity errors are ignored on input. Otherwise, they are translated into a Bad Character. The default value is 70 (ignore parity, even parity, 7 data bits).

MISC The 8 bits of the binary word represented by *MISC* are used as follows:

Bit 7 If 1, echo received data. (used for full duplex terminals) Default = 1.

Bit 6 If 1, this device is a shifting device (see Shift Control under rows 7 and 8) Default = 0

Bit 5 If 1, Break is enabled on output. Default = 1.

Bit 4 If 1, and Bit 5 = 1, Break on input is treated as an EOT (see EOT under row 9). Default = 1.

Bits 3-0 After an EOT (see row 10) is transmitted to the device, the first *N* characters received are ignored where *N* is the binary word represented by these 4 bits. This is used to dump acknowledgment of the transmitted EOT which may precede input text. Default = 0.

ROW 1 - CONTINUATION

This was discussed under Device Control in the general documentation (Sectiom 1.4). Its form is:

WIDTH, *CHAR*, *INDENT*

The default value is 72, 130 (overstrike),6.

ROW 2 - NEWLINE

This row has four elements as follows:

TIME_OUT, *IDLES*, *CODE*, *CODE*

At the end of each physical line output, the two *CODES* are transmitted. If a code value is 128, its transmission is suppressed. Following the outputs, the number of *IDLE* characters (see row 4) determined by the following algorithm are transmitted:

$$IDLE\_COUNT \leftarrow 1 + \lfloor CARRIAGE\_POSITION \div IDLES$$

If *IDLES* = 0, no Idles are transmitted. The default row is 0,0,13 (carriage return), 10 (linefeed). A newline received during input terminates the input operation. The default input code is 13 (carriage return), 10 (linefeed). A newline received during input terminates the input operation. The default input code for *NEWLINE* (See Input Translate Table) is 13 (carriage return).

If *TIME_OUT* ≠ 0, and an interval longer than *TIME_OUT* seconds elapses between the receipt of any two characters during an input operation, the receipt of a newline is simulated. The maximum value for *TIME_OUT* is 255 seconds.

ROW 3 - BACKSPACE

This row has only one element, the value of which is the code transmitted for backspace. If the code value is 128, transmission is suppressed. (Default = 8).

ROW 4 - IDLE

Similar to Backspace, above, but used when idle characters are required. (Default = 0). Idle Characters received on input are ignored.

ROW 6 - FORM FEED

This row has four elements, the first two of which were discussed under Device Control in the general documentation (Section 1.4). The form is:

LIMIT, *COUNT*, *CODE*, *CODE*

When the page limit is reached, the two *CODE*'s are transmitted. If a code value is 128, its transmission is suppressed. The transmission is repeated *COUNT* times. The default row is 0 (paging off), 0, 10 (linefeed), 0 (idle). Form Feed is meaningless on input.

ROW 7 & 8 - SHIFT DOWN & UP

These rows have one element each. Devices with fewer than 7 data bits usually expand their character set with shift control. In effect, bit 6 of the data byte is set by Shift Up, and reset by Shift Down. On output, if row 0 indicates that this is a shifting device (IBM 2741 or equivalent), a data byte which has bit 6 different from the current setting causes a shift code to be transmitted. If the code given is 128, transmission is suppressed (Default = 128).

ROW 9 - END OF TRANSMISSION

Prior to each input operation, after the prompt (if any) has been output, an EOT is issued to the device. This "turns the line around", informing the device that input is requested. If the device precedes its inupt data with an EOT acknowledgment, row 0 specifies the number of characters in the acknowledgment, which are ignored.

This control row consists of two elements:

CODE, *CODE*

When an EOT is issued, the two *CODES* are transmitted. If a code is 128, its transmission is suppressed.

The default row is 0, 7 (idle, bell).

An EOT received during an input operation initiates an editing operation. If the device cannot accept prompts, all previous input for the current physical line is discarded.

If the device can accept prompts, previous input at and to the right of the current position is discarded. The remainder of the current physical line is re-transmitted to the device, preceded by a BOT and a newline, and followed by an EOT. The input operation is then resumed.

The default input code is 4 (EOT).

ROW 10 - BEGINNING OF TRANSMISSION

At the end of each input operation, a BOT is issued to the device. This again "turns the line around", informing the device that output follows.

The form of this row is identical to that for row 9

The default row is 0,0 (idle).

BOT on input is meaningless.

3.3 INPUT INTERRUPTION

As with all operations, input may be interrupted with a Hard Interrupt (*CONTROL* '→' on the main keyboard). The operation may also be terminated, however, with a Soft Interrupt or Attention (*CONTROL* '←'). This causes a newline input to be simulated, and processing proceeds as it would if an actual newline were input from the device.

If a "*HARD*" interrupt is used, the interface should be reset (⎕*OU ADDR*,|) before attempting to use it again.

3.4 EIA INTERFACE INSTALLATION INSTRUCTIONS

The EIA interface is selected for output via:

⎕*OU* 102

The response returned should be 102 193 8 or 102 225 8.

This indicates that the EIA interface (address 2) has been selected for output, and that it is ready, but not yet connected to any device. From this point on, the procedure is different for each device to which the interface is to be connected.

3.4.1 PRINTERS AND TERMINALS

1. Plug the small adapter box into the top connector on the rear of the MCM/900.

2. Place the "PROMPT" switch in the "ON" position, and on the MCM/900 type:

   ⎕*OU* ⍳0

   This should respond 102 255 8. (If it responds 0 0 0 repeat ⎕*OU* 102 and verify responses).

3. Connect the terminal (or printer) to the EIA adapter box supplied. Place the "DTR" (Data Terminal Ready) switch in the "X" position. (In this position, DTR is supplied by the terminal).

   Turn on the terminal. Obtain the device status with:

   ⎕*OU* 102 2

   This should respond with 102 225 18. If it responds 102 225 8, move the "DTR" switch to the "1" position. (This forces DTR to be true, since it has been determined that the terminal does not supply the signal).

4. If the terminal is a 300 Baud ASCII device with *APL* typewriter pairing character codes, it should now be ready to accept output. To check this, type:

   ⎕←⍳9

   The terminal should then display the numbers 1 through 9. If it does not, special support is required. Contact your local MCM representative for details.

5. If the terminal has a keyboard, it can also be used for input to the MCM/900. To do this, type:

⎕*IN* 102

(102 can be substituted with whatever device address you are using). The response should be 102 225 18. Then type:

⍞ '*HELLO*'

The terminal should print "HELLO" and ring its bell, indicating that it is awaiting input. If you then type:

12345 (RETURN)

on the <u>terminal</u> keyboard, the terminal should echo the "12345", do a Return and Linefeed, and the MCM/900 should display

*HELLO*12345

3.4.2 <u>MODEMS AND ACOUSTIC COUPLERS</u>

1. Ensure that nothing is plugged into the top connector on the rear of the MCM/900 and type:

⎕*OU* ⍳0

This should respond 102 193 18. (If the response is 0 0 0 repeat ⎕*OU* 102.

2. Plug the modem (or acoustic coupler) into the top connector. On the MCM/900, type:

⎕*OU* ⍳0

The response should be 2 193 24, indicating that the modem is ready but has not detected a line carrier.

3. Ensure that if the modem has "ECHO", "COPY", or "ANSWER" switches, they are all off. Make the telephone or hard-wire connection. This should result in an audible tone from the modem. Type

⎕*OU* ⍳0∘⎕*IN* ⍳0

The response should now be 102 193 18, indicating that the modem has a carrier and is ready to transfer data. (If the last number in the response is odd, the system to which you have just connected has tried to send data which the system has ignored.)

4. If the system to which you are connected uses a 300 Baud ASCII/APL typewriter-pairing overlay character code, you are now ready to talk to it. Type:

   ⍞''∘⎕←'-- (something the system expects)--'

   This should transmit the quoted character string to the external system, and display the first line of its response to the MCM/900. If it does not, special support is required. Contact your local MCM representative for details. (Note - If the external system does not respond, the MCM/900 will wait indefinitely for it to do so. Type CONTROL/← or CONTROL/SHIFT/→ to escape from this situation.)

3.5 LARGE BUFFER EIA DRIVER

The large buffer EIA driver allows an MCM computer to receive up to several thousand characters at one time. The user should have some experience with *MCM/APL* and with the built-in, one-line EIA driver. For the rest of this section "Large Buffer EIA Driver" is referred to as simply "LB Driver".

3.5.1 GENERAL DESCRIPTION

The LB Driver is very similar to the built-in one line EIA driver, with the following exceptions:

SIZE: The built-in driver returns only a single line, while the LB driver returns a character matrix whose size is specified by the user, via the device control tables.

ECHO, PROMPT: Since the LB driver is designed to receive information from other computers or from terminals that transmit an entire block of data at a time, neither echoing nor prompting are supported.

NEWLINE: When the one-line driver receives a NEWLINE character it terminates the input operation and returns the line to the user. When the LB driver receives a NEWLINE it advances to the start of the next row of the result, and continues to receive data.

EOT: The one-line driver takes an EOT (End Of Transmission) character as a request for editing. This is useful only if the input is coming from a person sitting at a keyboard. The LB driver takes an EOT as the signal that the transmitted block is complete and returns the result to the *APL* user.

3.5.2 THEORY OF OPERATION

The LB driver is similar to the built-in, one-line driver, but returns an array of characters, rather than a single line. The size of this array is specified by the user, via the control table, and may be any size as long as it meets the following restrictions:

1) The maximum allowable width (number of columns) is 132. If you request a result wider than 132 columns, a *WIDTH ERROR* is issued.

2) There must be enough workspace available for the result. If the result is to be written to tape or disk or if AVS is in active use, then the result should not be bigger than about 3900 characters.

When the driver is invoked by Quote-Quad it starts at the upper left corner of the result matrix and places characters as they are received in the first row. The driver will advance to the next row when:

1) The last character of the first row has been filled, or

2) A NEWLINE character is received.

The driver will continue to receive characters in this manner until:

1) The last character of the last row is filled, or

2) An End-Of-Transmission (EOT) character is received, or

3) A soft (control/arrow) or hard (control/shift/arrow) interrupt is issued from the keyboard, or

4) There is a pause between characters that is greater than the limit specified. This will occur only if you have specified a time limit, since the default time is infinity.

3.5.3 LOADING AND USING THE DRIVER

The LB driver is supplied on tape or disk as a complete I/O table named 'LBY'. The following sequence outlines the steps necessary to setup the driver:

1) Make sure that the device to be used is connected to the EIA interface and that the prompt switch is OFF.

2) Select the EIA interface for input via ⎕*IN X* where *X* is the omniport address of the EIA interface.

3) Read the driver into memory with (group) ⎕*XR 'LBY'*.

4) Set up the device table in the machine using: *'I'* ⎕*YW 'LBY'*.

The driver is now ready for use, and is invoked by issuing a quote-quad (⍞''). The driver is supplied with the same default settings as the standard EIA driver (except for ECHO and PROMPT, which are off), and the standard result size is 24 by 80.

3.5.4 CHANGING THE RESULT SIZE

The size of the result returned by the LB driver is controlled by the fifth row of the control segment of the input table. To find out what the current result size setting is use:

```
⎕YI[⎕IO] 5
```

To change the result size, specify the desired size (rows, columns) as the left argument. For example, to have a result of 20 rows by 120 columns:

```
20 120 ⎕YI[⎕IO] 5
```

3.5.5 NOTES ON EOT

The sending terminal or computer MUST transmit an EOT at the end of a block of data. The default code is 4, which is the standard ASCII definition of EOT. If the terminal or computer you are using sends some other character at the end of a block, then the input translate table must be changed so that the driver will recognize that character as an EOT. For example, if your device sends an ASCII BEL (07) at the end of a block, then:

```
¯9 ⎕YI[⎕IO+1] 7
```

would specify that an input code of 7 (BEL) represents a logical EOT operation. The index (⎕*IO*+1) indicates that we are accessing the input-translate segment of the control table, the negative symbol (¯) indicates that the code represents a control operation rather than a simple translation, while the 9 indicates that the particular control operation is the one covered by row 9 of the control-segment of the device control table: namely EOT.

SECTION 4.0 HYTYPE/QUME INTERFACE

This interface, consisting of a driver in the communications subsystem and an omniport HYTYPE/QUME interface board, (PI-20), enables the system to transmit to a DIABLO HYTYPE or QUME Q30 or Q45 printer. These are directly controlled mediumspeed printers with interchangeable type fonts and high quality print.

The answer-back code for these devices is 66 (output + Type 2). In order to divert output to such a printer, the user need only type:

⎕*OU* ⎕*YA* 66

after connecting the printer to the system and loading it with paper. This assumes that the printer has an *APL* print wheel and 120 column width paper.

In order to have any other type of access to these printers, the user must either modify the device table or use the omniport access functions. Both are described below.

4.1 DIRECT ACCESS

It is possible to drive these printers directly using the omniport access functions:

⎕*OU* and ⎕*BO*

The printer interface requires a 3 bit command, and thirteen bits of data. The least significant data Bit is called Data 1/2 (2*0-1) and the most significant bit is called Data 2048 (2*12-1). The most significant 8 bits of this field are output as an omniport command byte using ⎕*OU*, and least significant 8 bits are the data byte output via ⎕*BO*, following the ⎕*OU* command.

The 3 bit command forms a binary word whose value is 0 through 7. The meaning of these commands is as follows:

0 - Set Printer Flag
1 - Print Character
2 - Move Carriage
3 - Toggle Ribbon States
4 - Feed Paper
5 - Enable Platens
6 - Form Feed
7 - Restore Printer

The operation of these commands will be described using the following functions:

```
        ∇     C COMMAND DATA;ADDR
   [1]        DATA←32 256⊤⌊2×DATA
   [2]        ADDR←1↑⎕OU⍳0
   [3]        ∘⎕OU ADDR, 8 32⊥C,1↑DATA∘⎕BO 1↓DATA
        ∇
        ∇      R←STATUS
   [1]        R←(8⍴2)⊤''⍴2↓⎕OU⍳0
        ∇
```

0 - SET PRINTER FLAG

Set the printer flag to the state indicated by Data 1048. All other data bits are ignored. The function of the flag is determined by the printer. It is intended to be used to turn the printer on and off.

Thus for a printer which is equipped with this option, the following will turn it on:

0 COMMAND 1048

1 - PRINT CHARACTER

The character whose code is represented by the binary word in Data 1 through Data 64 is printed. If the printer is equipped with variable ribbon step, the ribbon is stepped according to the binary word in Data 128 through 512. All other data bits are ignored. No carriage motion occurs.

Ribbon step
| 512 256 128 | 64 32 16 8 4 2 1

Thus to print an *'A'*, whose code is 97 on an *APL* print wheel:

1 *COMMAND* 97

2 - MOVE CARRIAGE

The carriage is moved the number of increments (60ths of an inch) represented by Data 1/2 through data 512. If Data 1024 is 0, the carriage is moved to the right. If Data 1024 is 1, the carriage is moved to the left. Data 2048 is ignored. Any attempt to move the carriage past either end stop will result in a check condition.

Thus to move a distance *N* inches (left if *N* is negative):

2 *COMMAND* (1024**N*<0)÷ 60×|*N*

NOTE: HYTYPE I's ignore Data 1/2.

3 - TOGGLE RIBBON STATES

There are two state bits which affect ribbon position: UP/DOWN, and RED/BLACK. If DATA 2048 is 1, the UP/DOWN state is toggled. If Data 1024 is 1, the RED/BLACK state is toggled. All other data bits are ignored.

The ribbon state is normally UP and BLACK. Thus, in order to drop the ribbon:

3 *COMMAND* 2048

Repeating this command will raise it again.

NOTE: ALL HYTYPES, and QUMES which do not have a red ribbon option, will drop the ribbon if the state is set to RED.

4 - FEED PAPER

The paper is moved the number of increments (48th's of an inch) specified by Data 1 through Data 512. If Data 1024 is 0, the paper is fed up (normal motion). If Data 1024 is 1, the paper is fed down (reverse motion). Data 1/2 and Data 2048 are ignored. Thus to feed the paper *N* inches (down if *N* is negative):

4 *COMMAND* (1024×*N*<0)+48×|*N*

5 - ENABLE PLATENS

Printers with split platens may have paper feed for each platen enabled independently. If a platen is disabled, it will ignore paper feed commands. If Data 2048 is 1, the left platen is enabled. If Data 1024 is 1, the right platen is enabled. Printers with only one platen are considered to have only a right platen.

6 - FORM FEED

Printers with the top-of-form option will feed the paper to the top of the next form. Others will ignore this command. All data bits are ignored.

7 - RESTORE PRINTER

If a check condition occurs, the printer must be restored to remove the condition. The carriage is restored to the left margin. No other functions are affected. All data bits are ignored.

EXAMPLE: To restore the printer execute:

```
⎕OU ADDR, 224
```

PRINTER STATUS

The status returned from the printer is an 8 bit binary word, defined as follows:

- Bit 7 - Paper feed ready
- Bit 6 - Carriage ready
- Bit 5 - Character print ready
- Bit 4 - Ribbon up
- Bit 3 - Ribbon red
- Bit 2 - Paper out
- Bit 1 - Check condition (see carriage motion)
- Bit 0 - Printer powered and ready

Normal (READY) status returned by the printer is 241 (BITS 7,6,5,4,0). Bits 0,1 and 2 should be checked prior to any operation. Bits 5,6 and 7 should be checked prior to each operation for the corresponding bit (ex: Bit 5 before printing a character). Bits 3 and 4 are used for setting a desired ribbon state. To set the ribbon unconditionally UP, for example:

```
3 COMMAND 2048×~STATUS[⎕IO+7-4]
```

Note that the bits are numbered in reverse order to that in which STATUS presents them. Note also the difference between this method and the one discussed under Ribbon States, which toggles the ribbon state.

4.2 DEVICE TABLES

The OUTPUT segment is set up by the system for an *APL* print wheel. If the user wishes to use another print wheel, he should replace the output segment with the translate codes appropriate to that wheel.

Usage of the CONTROL segment is as follows:

ROW 0

This row has four elements. The form is *XX*, *MISC*, *HSPACE*, *VSPACE*. The first element is described in the general documentation of the control tables. The second element has the value:

MISC←4 2 4 8 ⊥ *PLATENS*, *FLAG*, *RIBBON*, *STEP*

At the beginning of each output, the printer is set so that the platens are enabled according to *PLATENS*, and the printer flag and ribbon states are set to *FLAG* and *RIBBON*. *STEP* is the ribbon step which is issued with each print command.

The defaults are:

PLATENS = 1 (right)

FLAG = 1 (on)

RIBBON = 2 (up, black)

STEP = 7 (Max. Step)

The third and fourth elements of row 0 are the number of half increments per horizontal space (*HSPACE*) and vertical space (*VSPACE*), respectively. For HYTYPE I's the horizontal spacing should be even, and for all printers the vertical spacing should be even. The defaults are 12 (10 characters/inch) and 16 (6 lines/inch).

Row 1

This row was described in the general documentation. Its form is:

WIDTH, *CHAR*, *INDENT*

The defaults are 120, 130 (overstrike) and 6.

Row 2

Only the second element of this row is used. ALL output is indented the number of character spaces indicated by this element. The default is 0.

ROW 6

This row was also described in the general documentation. Its form is:

PAGE_LIMIT, *COUNT*, *CMD*, *DATA*

When the *PAGE_LIMIT* is reached, the 16-bit binary word represented by:

$$CMD+DATA\times COUNT$$

is issued to the printer. The defaults are 0 (paging off), 0, 128 (paper feed up), 16 (1 line.)

To print *M* lines on a form *N* lines high, the first two elements should be *M*,*N*-*M*.

4.3 HYTYPE APL/10 PRINT WHEEL CODES

0:W	32:•	64:¯	96:◊
1:W	33:¨	65:α	97:A
2:W	34:)	66:⊥	98:B
3:W	35:<	67:∩	99:C
4:W	36:≤	68:⌊	100:D
5:W	37:=	69:∊	101:E
6:W	38:>	70:_	102:F
7:W	39:]	71:∇	103:G
8:W	40:∨	72:∆	104:H
9:W	41:∧	73:⍳	105:I
10:W	42:≠	74:∘	106:J
11:W	43:÷	75:'	107:K
12:W	44:,	76:⎕	108:L
13:W	45:+	77:|	109:M
14:W	46:.	78:⊤	110:N
15:W	47:/	79:○	111:O
16:W	48:0	80:⋆	112:P
17:W	49:1	81:?	113:Q
18:W	50:2	82:⍴	114:R
19:W	51:3	83:⌈	115:S
20:W	52:4	84:∼	116:T
21:W	53:5	85:↓	117:U
22:W	54:6	86:∪	118:V
23:W	55:7	87:⍵	119:W
24:W	56:8	88:⊃	120:X
25:W	57:9	89:↑	121:Y
26:W	58:(	90:⊂	122:Z
27:W	59:[	91:←	123:{
28:W	60:;	92:⊢	124:⊣
29:W	61:×	93:→	125:}
30:W	62::	94:≥	126:$
31:W	63:\	95:-	127:¬

SECTION 5.0 CENTRONICS PRINTER

5.1 GENERAL DESCRIPTION

This interface may be used with any printer which uses a standard Centronics (R) parallel interface. The answer-back code for these devices is 70 (Output + Type 6). In order to use such a printer, the user must first obtain a copy of the communications tables for these devices. If the name of the object containing the tables is *YCP*, the user must then execute the statement:

⎕OU ⎕YA 'O' ⎕YW 'YCP'

after connecting the printer to the system, loading it with paper, turning it on, and pressing the SELECT button on the printer. If the printer is set to address 6, the above statement should return:

6 70 153

Any other result indicates a problem somewhere. The value 6 0 0 or 0 0 0 indicates that the printer is not connected to the system. The value 6 70 140 indicates that the SELECT button was not pressed.

The normal tables received from MCM for this type of printer will assume the normal *APL* character set at 132 characters per line.

Detailed operation of the printer is described in the sections which follow.

5.2 DIRECT ACCESS

As with all OMNIPORT devices, these printers may be driven with the direct access functions ⎕*OU* and ⎕*BO*. The status returned by the printer as the last element of the result of the function ⎕*OU* is an 8 bit binary word defined as follows (Bit 7 is the most significant bit):

1 - Bit 7 -	This bit specifies the type of printer 1 specifies a MCP-713 or 712 0 specifies a Printronix 300
0 - Bit 6 -	Reserved for future use, currently 0
0 - Bit 5 -	Reserved for future use, currently 0
1 - Bit 4 -	No printer hardware fault
0 - Bit 3 -	The printer has acknowledged the last byte sent to it.
1 - Bit 2 -	Printer busy
0 - Bit 1 -	Printer out of paper
1 - Bit 0 -	Printer ready and selected

Data should not be sent to the printer unless bits 5 through 0 are 0 1 0 0 1.

Only one command is defined for these printers, namely the RESET command. If the printer address is *A*, executing

⎕*OU A*, 1

will reset the printer. This stops any activity, clears the printer's buffer, and restores the print head to the left margin. It does not force the printer to the top of a page. Paging logic is unaffected by the reset command. The only way to reset the printer's page logic is to power it down and up again, or (on some models) reload its VFU.

These printers are essentially ASCII-driven devices. This means that all printer functions are driven by ASCII character codes. The exact function of some of the control codes is printer-dependent and the description of these functions will be found in the manuals for the printers in question. However, the following properties are common to all printers (code values are given in decimal):

CODE	FUNCTION
32	Space (blank)
33 - 126	Printable characters
161 - 254	Printable characters
10	Linefeed (causes line to be printed)
13	Carriage Return (Does not cause paper feed)
12	Top of Form (causes line to be printed)

Another property of these printers is that most of them are buffered. Printable characters are merely stored in a buffer memory until either the buffer fills up or a control character such as a Linefeed is received, causing the line to be printed. Note that a Linefeed also causes an implicit Carriage Return. That is, characters received following a Linefeed will be printed starting at the left margin of the next line.

As an example, try the following:

```
⎕BO 65
⎕BO 10
```

Note that nothing happens as a result of the first statement, but that the second causes the line "A" to be printed.

On some printers, receipt of a Carriage Return character does not cause the line to be printed. On these printers, the only overprinting that is possible is underlining. An attempt to overprint a character with anything other than an underline will cause the first character to be replaced by the second.

5.3 DEVICE TABLES

The output translate segment of the device control segment is capable of dealing with only the codes 0 to 127, but the printer itself may contain as many as 224 graphics. In order to deal with this problem, the CONTROL segment is set up as follows:

ROW 0:

```
XX,VFU,S01,S23
```

The first element XX is described in the general documentation Section 1.4. The second element VFU is 0 unless a VFU is installed on the printer (more about this later).

The last two elements are encoded as follows:

```
S01←(S0×32)+(S1×2)
S23←(S2×32)+(S3×2)
```

The graphic code set of the printer is broken up into six pieces of 32 characters each numbered 1,2,3,5,6,7. Whenever a code is fetched from the OUTPUT translate segment, the system breaks it into two fields:

```
CS←⌊CODE÷32
CR←CODE-CS
```

Since CODE can have the value 0 through 127, *CS* can have the value 0 through 3, and *CR* can have the value 0 through 31. The value of *CS* is then used to pick one of the values *S*0 through *S*3 defined above. Suppose this value is *SX*. This value is used in turn to pick one of the seven pieces of the printer's character set. The actual code transmitted is then:

```
TC←(SX+32)+CR
```

To summarize:

```
SS←S0,S1,S2,S3
CC←0 32⊤CODE
TC←SS[⎕IO+SS[1↑CC]]+1↓CC
```

Thus any four 32-character pieces of the printer's character set may be used with any one output operation. Printing a line which contains graphics from more than four of the 32-character pieces requires overprinting.

ROW 1:

This row is described in the Section 1.4, Device Control. Its form is:

```
WIDTH,CONT_CODE,CONT_INDENT
```

ROW 2:

The form of this row is:

```
XX,INDENT,CODE,CODE
```

The first element is not used. Every line of print is preceded by the number of spaces indicated in the second element, INDENT, and followed by the two codes given as the last two elements. These codes will normally be 128 10, specifying a linefeed only. In order to do overprinting, the codes 128 13 should be used, specifying a carriage return only.

ROW 6:

The form of this row is:

```
PAGESIZE,SKIPCOUNT,CODE,CODE
```

When *PAGESIZE* lines have been printed on a page, *SKIPCOUNT* repetitions of *CODE*,*CODE* are transmitted to the printer, unless *VFU* is 1 (see below), in which case the pair is transmitted only once. For a normal 60 line page on 11 inch paper at 6 LPI, this row will be 60 6 10 128.

ROWS 7, 8, 9, 10:

These rows are used only if an electronic VFU (Vertical Forms Unit) is installed on the printer. For such printers *VFU* (the second element of row 0 of the CONTROL segment) should be 1. In this mode, whenever the system finds itself about to do output at the top of a page (that is the second element of the result of ⎕*PC*⍳0 is zero), it loads the VFU as follows:

1) The code given in Row 7 is transmitted to start the VFU load function.

2) The two codes given in Row 10 are transmitted to indicate channel 1 (top of form) in the first position of the electronic VFU.

3) The two codes given in Row 9 are transmitted as many times as necessary to fill out the form size. This number is calculated from the first two elements of Row 6:

¯1+*PAGESIZE*+*SKIPCOUNT*

4) The code given in Row 8 is transmitted to terminate the VFU load function.

When using this mode, the last two elements of Row 6 will normally be 12 128, indicating a Form Feed only.

5.4 MCP-713 CHARACTER CODES

SECTION 0,4	1	2	3	5	6	7
0,128:NUL	32:	64:@	96:`	160:	192:¨	224:Ä
1,129: --	33:!	65:A	97:a	161:|	193:¯	225:é
2,130: --	34:"	66:B	98:b	162:♥	194:≤	226:ç
3,131: --	35:#	67:C	99:c	163:♠	195:≥	227:Ñ
4,132: --	36:$	68:D	100:d	164:♣	196:≠	228:Ü
5,133: --	37:%	69:E	101:e	165:⌿	197:∨	229:à
6,134: --	38:&	70:F	102:f	166:⍀	198:∧	230:è
7,135:BEL	39:´	71:G	103:g	167:Φ	199:÷	231:ì
8,136: --	40:(	72:H	104:h	168:⊖	200:×	232:ò
9,137: --	41:)	73:I	105:i	169:⍉	201:ω	233:ù
10,138:LF	42:*	74:J	106:j	170:⍋	202:∊	234:ä
11,139:VT	43:+	75:K	107:k	171:⍒	203:⍴	235:ë
12,140:FF	44:,	76:L	108:l	172:⍝	204:~	236:ï
13,141:CR	45:-	77:M	109:m	173:⍟	205:↑	237:ö
14,142: --	46:.	78:N	110:n	174:⍑	206:↓	238:Ö
15,143: --	47:/	79:O	111:o	175:⌹	207:⍳	239:ă
16,144: --	48:0	80:P	112:p	176:⍞	208:○	240:ĕ
17,145:SELECT	49:1	81:Q	113:q	177:⍕	209:←	241:Ĭ
18,146: --	50:2	82:R	114:r	178:⌷	210:→	242:ŏ
19,147:DESLCT	51:3	83:S	115:s	179:◊	211:α	243:Ŭ
20,148: --	52:4	84:T	116:t	180:⊢	212:⌊	244:ß
21,149: --	53:5	85:U	117:u	181:⊣	213:⌈	245:
22,150: --	54:6	86:V	118:v	182:⌶	214:∇	246:†
23,151: --	55:7	87:W	119:w	183:⊥	215:Δ	247:└
24,152: --	56:8	88:X	120:x	184:■	216:∘	248:┘
25,153: --	57:9	89:Y	121:y	185:⎕	217:'	249:┐
26,154: --	58::	90:Z	122:z	186:⍂	218:⎕	250:┌
27,155: --	59:;	91:[	123:{	187:⍃	219:⊂	251:├
28,156: --	60:<	92:\	124:|	188:⍄	220:⊃	252:┴
29,157:VFULOAD	61:=	93:]	125:}	189:∝	221:∩	253:┤
30,158:VFUEND	62:>	94:^	126:~	190:⌽	222:∪	254:┬
31,159:VFUCMD	63:?	95:_	127:DEL	191:¬	223:⊤	255:DEL.

SECTION 6.0 ON WRITING DEVICE SUPPORT

6.1 PROCEDURE

The following procedure should be used in setting up the device table for a completely new device. The examples are for a Teletype Model 33.

1. Determine the set of *APL* characters which corresponds directly to single characters on the device. Let us assume that we have this set in a vector called *A*1.

```
 A1←'0123456789'
  A1←A1,'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
  A1←A1,'$''()*+,_./'
  A1
01234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ $'()*+,_./
```

2. Determine the set of device codes which correspond to the above set of *APL* characters (ignore parity bits). Let us place this in a vector *D*1.

```
D1←48+(⍳10)-⎕IO
D1←D1,65+(⍳26)-⎕IO
D1←D1,32+0 4,7+(⍳9)-⎕IO
```

3. If the device is to be used for output, load these values into the SINK table output translate segment. (Note: Before accessing the device tables in any way, make sure that SOURCE and SINK are set to the device in question. For example, if the address of the device is 102, execute: ⎕*IN* 102∘⎕*OU* 102).

```
D1 ⎕YO[2+⎕IO]A1
```

4. If the device is to be used for prompted input (this requires the PROMPT bit in the answer-back code to be on), load these values into the Source table output translate segment.

```
D1 ⎕YI[2+⎕IO]A1
```

If this results in a *TYPE ERROR*, input may not be prompted. If PROMPT is required:

a) *EXPUNGE* the input table '*I*'⎕*YX* ⍳0

b) Turn on the prompt switch on the adapter box

c) Repeat the operation

5. If the device is to be used for input, *A*1 must be converted to numeric, and the result loaded into the Source table input translate segment.

```
(⎕Y A1)⎕YI[1+⎕IO]D1
```

NOTE: Only the Source table has an input translate segment.

6. Next determine the set of characters which are to be represented as overstrikes on the device. Let us call this set *A*2. Set these locations in both output translate segments to ¯2.(In this example, *A*2 is empty).

```
¯2 ⎕YO[2+⎕IO]A2
¯2 ⎕YI[2+⎕IO]A2
```

NOTE: The table of valid overstrike characters is in ROM and may not be altered.

7. Determine the set of characters which are to be represented as mnemonics on the device. Let us call this set *A*3. Set these locations in both output translate segments to ¯1.

```
ALF←⎕Y (⍳109)-⎕IO
A3←(~ALF∊A1)/ALF
¯1 ⎕YO[2+⎕IO]A3
¯1 ⎕YI[2+⎕IO]A3
```

8. Determine the set of device codes which correspond to control characters expected in input. Let us call this set *DC*. Load these locations in the input translate segment with the appropriate control codes (called *AC*).

```
DC←13     (carriage return)
AC←¯2     (newline)
AC ⎕YI[1+⎕IO]DC
```

9. Fill all remaining unspecified slots in the input translate segment with Idle or Bad Character codes, as desired.

```
X←(⍳32)-⎕IO
DI←((~X∊DC)/X),127
¯4 ⎕YI[1+⎕IO]DI
X←32+(⍳95)-⎕IO
DB←(~X∊A1)/X
108 ⎕YI[1+⎕IO]DB
```

This can also be done by block filling the segment with Idles and/or Bad Characters prior to step 5.

10. At this point, the input and output translate segments have been completely set up. If desired, the contents of these segments should now be saved:

```
YOO←⎕YO[2+⎕IO]ALF (see step 7.)
YIO←⎕YI[2+⎕IO]ALF
YII←⎕YI[1+⎕IO] (⍳128)-⎕IO
```

Note that since *YOO* and *YIO* are usually the same, some duplication can be avoided by simply setting up the SINK table output translate segment (which we have saved as *YOO*) and copying it into the source table output segment if prompts are to be output to the device.

```
YOO ⎕YI[2+⎕IO]ALF
```

11. It is now necessary to set up the control segments. Since the form of the control table is different for each type of device, only the most common type, EIA, will be discussed here.

First the row containing general information about the device (row 0) is set up.

(a) The first element of this row is made up of the sum of any of four possible values which represent flags controlling the device as follows:

1 - Mnemonics valid on input.
2 - Overstrikes valid on input.
4 - Trailing spaces on each logical line truncated on output.
32 - Prompt valid on input.

In our case (*TY*33) we shall set the first element to:

```
R00←32+4+1
```

Since the device has no backspace, overstrikes are impossible (This is also the usual setting for a CRT, for which backspace is destructive).

NOTE: The following discussion applies <u>only</u> to EIA devices.

(b) The second element is the sum of four possible flags and a count, defined as follows:

- 128 - Echo input back to the device (FULL DUPLEX)
- 64 - The device has a two-level code set.
- 32 - A Break from the device interrupts output.
- 16 - A Break from the device simulates an EOT on input.
- 0-15 - Ignore this many characters at the start of each input.

In our case, we shall set the first element to:

```
R01←128+32+16+0
```

(c) The third element is the sum of four possible flags and a count as follows:

- 64 - Ignore parity errors on input
- 16 - Serial word has two stop bits
- 8 - Serial word has no parity bit.
- 4 - Serial word has even parity
- 3-0 - Number of data bits in serial word, minus 5.

In our case, we shall set the third element to:

```
R02←64+16+4+(7-5)
```

(d) The fourth element is a code for the data rate, determined by the following formula:

```
R←⌊¯0.5+25000÷B
```

where *B* is the Baud rate for the device. In our case:

```
R03←⌊¯0.5+25000÷110
```

Having determined all four elements, they are then loaded into row 0 of the control segments:

```
R0←R00,R01,R02,R03
  R0 ⎕YO[⎕IO]0
  R0 ⎕YI[⎕IO]0
```

12. Row 1 of the control table is determined next. It consists of these elements - the display width to be used with the device, the code for the continuation character (device code, or 129 for mnemonic representation, 130 for overstrike representation, or 128 for no character), and the size of the indent to be placed at the beginning of continuation lines.

    In our case, we shall use a width of 72, mnemonic representation of the continuation character, and a 6-space indent:

```
R1←72 129 6
R1 ⎕YO[⎕IO]1
R1 ⎕YI[⎕IO]1
```

13. Row 2 of the control segment gives the newline output sequence for the device. The first element is a time-out for input. If the element is non-zero, and that many seconds elapse between two successive characters during an input operation, a newline is simulated, terminating the input. In our case, we shall set:

```
R20←0
```

    The second element represents an idle count for the operation. If this element is non-zero, the physical cursor (or carriage) position at the end of the line is divided by this number, and the result is the number of idles transmitted after the newline. In our case:

```
R21←0
```

    The third and fourth elements are a pair of codes forming the newline operation. (These must be either device codes, or 128 for no character). In our case, the newline operation consists of a carriage return (13) followed by a line feed (10).

    The resulting values are then loaded into the row:

```
R2←R20,R21,10 13
R2 YO[⎕IO]2
R2 YI[⎕IO]2
```

14. Rows 3 and 4 of the control segment give the backspace and idle output codes for the device. In our case, the device has no backspace, and the idle code is 0:

```
X34←2 1ρ128 0
X34 ⎕YO[⎕IO]3 4
X34 ⎕YI[⎕IO]3 4
```

    Row 5 is not used.

15. Row 6 gives the form feed output sequence for the device. The first element is the number of lines to be printed on each page. The second element is a count of the number of times the pair of codes represented by the third and fourth elements are to be transmitted to make up the form feed action.

    In our case, although a teletype usually has continuous paper, requiring the first element to be zero, we will assume we have loaded it with 66 line paper, of which 60 lines are to be used for printing:

```
R6←60 6 10 128
R6 ⎕YO[⎕IO]6
R6 ⎕YI[⎕IO]6
```

16. Rows 7 and 8 give the shift codes for a device with a two-level code set. Since the teletype is not such a device, we have:

```
X78←2 1ρ128
X78 ⎕YO[⎕IO]7 8
X78 ⎕YI[⎕IO]7 8
```

17. Rows 9 and 10 give the output sequences for EOT and BOT respectively. The code pair represented by the second and third element are output, followed by the number of idles given in the first element.

    In our case, End-of-Transmission, and Beginning-of-Transmission are not very meaningful, since the teletype is a full-duplex device. However, since EOT occurs at the beginning of a request for input, we can transmit a BEL (7) to the device to tell the user at the keyboard that input is being requested:

```
X9T←2 2ρ 7 128, 128 128
X9T ⎕YO[⎕IO]9 10
X9T ⎕YI[⎕IO]9 10
```

18. Now that the control segments have been set up, it will probably be useful to save them:

```
YOC←⎕YO[⎕IO](⍳11)-⎕IO
YIC←⎕YI[⎕IO](⍳11)-⎕IO
```

    Note that since in most cases the control segment can be the same for both source and sink tables, it is usually easier to set up one (say, the sink table), and use the saved result (which we have called *YOC*) to load the other:

```
YOC ⎕YI[⎕IO](⍳11)-⎕IO
```

19. Further, now that the entire contents of both tables have been loaded, it may be useful to save them in toto, in order to make reloading easier than setting up each segment individually:

```
'O' ⎕YR 'YYO'
'I' ⎕YR 'YYI'
```

At any future time, the tables may be reloaded as follows:

```
'O' ⎕YW 'YYO'
'I' ⎕YW 'YYI'
```

The expunge (⎕*YX*) is not necessary, because ⎕*YW* will replace an existing table. In this form the tables *YYO* and *YYI* are not valid data to the *APL* system (5↔⎕*NC*'*YYO*'). However they may be written to disk (or tape) using the *EASY* command

```
∇ ⎕XW 'YYO YYI'
```

and retrieved when needed via:

```
∇ ⎕XR 'YYO YYI'
```

6.2 2741 CONSIDERATIONS

The following special considerations should be noted in writing device tables for IBM 2741 terminals (or equivalent).

1. Since the 2741 can reside indefinitely in any one of three states, and since it is capable of printing output in only one of these states, the user must be careful never to transmit an EOT character (code 60) to the device, except via row 9 of the control segment. Similarly, row 10 must contain a BOT (code 52).

   If device state is altered (either by power down or switching temporarily into LOCAL mode), it must be reset by issuing a request for ⍞ input, and pressing the return key on the terminal. If the terminal is equipped with a reverse Break, the same affect may be achieved in direct access mode with the following sequence:

   ```
   ⎕OU A,4   (where A is the device address)
   ⎕DL 0.5
   ⎕BO 52
   ```

   This issues a reverse break and a BOT to the device. This will only work if I/O to the device has already been attempted.

2. Since the 2741 has a two-level code set, the Shift bit (flag value 64) in the second element of row 0 of the control segment, must be set, and the shift codes (31 and 28) must be loaded into rows 7 and 8 of the control segment.

   Further, to maintain synchronization with respect to case, the newline sequence in row 2 must contain a downshift (31).

3. The appropriate newline idle count is 8. The user may have to tune this for his specific terminal.

4. A 2741 runs at 134.5 Baud, with a 6-bit data word, 1 stop bit, and odd parity.

5. When an EOT (60) is transmitted to a 2741, it responds with a BOT (52). Thus the ignore count in the second element of control segment row 0 should be 1.

6. If the form feed is being used, it should be simulated using linefeed/idle (46 61)or newline/idle (45 61). The former is to be preferred, but some 2741 type terminals do not respond to a linefeed.

7. The following is a list of the code values for 2741 control characters and their corresponding input translate segment values:

Newline	- 54 (¯2)
Backspace	- 29 (¯3)
Idle	- 61 (¯4)
Linefeed	- 46
Downshift	- 31 (¯7)
Upshift	- 28 (¯8)
EOT	- 60 (¯9)
BOT	- 52 (¯10)

6.3 EXTERNAL SYSTEM COMMUNICATION

The following special considerations should be noted in communicating with external computer systems.

1. The most difficult problem to deal with in talking to an external computer system is the multi-line response. Since the Communications Subsystem will only accept one line of input at a time, steps must be taken to ensure that this condition is not violated, otherwise input will be lost.

This is best done by having a monitor run in both the local and remote-systems, which hands across and acknowledges one line at a time. Included in this description are the listings of a pair of *APL* functions, *M∆IN* and *M∆OUT*, which implements this procedure for character matrices. With these functions it is possible to build a package which transfers data of an arbitrary nature by first converting it to a character matrix, then converting back again after transfer.

Note the use of the function *XFER*, which guarantees that sychronization of the two systems is maintained.

2. Except in very unusual cases, the prompted input used for terminals is not meaningful for external computer systems. Thus the PROMPT bit in the answer-back of the hardware interface and/or the PROMPT flag in the first element of row 0 of the control segment should be off. Input should be requested using ⍞''.

3. If the two systems get out of sychronization, a request for input may wait indefinitely for a termination. In order to recover from this condition automatically, the time-out element of row 2 of the control segment should be set to a reasonable non-zero value (say, 30 seconds). This will cause the Communication Subsystem to terminate the input operation unilaterally if this time elapses without a response.

   At this time, all input which has been received up to this point is returned, and the program can then take action on the basis of the input which was received.

   Premature termination may also be triggered with an attention at the main keyboard (control, '←' for one second).

4. When first connecting to an external computer system, it may not be possible to have the monitors discussed in (1) active at that time. It is possible to deal in a very limited way with multi-line responses using the following technique.

   Pick a code which is known to be transmitted by external system immediately prior to a request for input (for 2741 compatible systems, this consists of an EOT). Specify that location in the input translate segment as a newline (¯2).

   Specify the location in the input translate segment corresponding to the device code for newline to some other character (such as Bad Character).

   The lines contained in an input response will then be delimited by the character substituted for the newline character, and can be picked out and dealt with separately.

The most serious limit to this technique is that the entire input response must not overflow the Communication Subsystem's input buffer of 133 characters.

5. Since the external system may respond to an EOT with a non-zero number of non-significant acknowledgement codes, the ignore count in the second element of row 0 of the control segment should be set to bypass that many characters.

6. Since some systems will not tolerate any delay between a newline and an EOT, it may be necessary to set the newline output code to nothing (128), and the EOT output code to newline/EOT.

APPENDIX A

ERROR MESSAGES AND POSSIBLE CAUSES

COMM TABLE ERROR

The device tables indicate a logically impossible operation:

(a) Overstrike or mnemonic representation has been requested for an output character which has none (e.g.'*A*').

(b) A negative value other than ¯1 or ¯2 was fetched from the output translate segment for an output character.

(c) A negative value was fetched from the output translate segment for one of the two characters being used in an overstrike representation of an output character.

(d) A value >127 was specified as the output code for a character in a control sequence (N.B. This does not apply to the continuation sequence).

(e) A width less than 3 plus the size of the continuation character plus the continuation indent was discovered at continuation time.

COMM DEVICE ERROR

A hardware error has occurred on the current comm device, or a physically impossible device operation was requested.

HYTYPE/QUME DRIVER

(a) Device not powered on.
(b) Device out of paper.
(c) Margins violated.
(d) Paper or carriage motion of ≥1024 increments requested.

EIA DRIVER

(a) Data Set (or Terminal) not ready.
(b) Transmit buffer not empty within 1 second (usually caused by too low or zero data rate).
(c) Modem Carrier Lost.
(d) Break received (while enabled) during output.

COMM TYPE ERROR

The current comm device has the wrong answer-back code for the current requested operation.

(a) Input (or output) requested for a non-input (or non-output) device.

(b) ⍞ input or ⎕ output requested for an unsupported device.

(c) A device table input (or output) translate segment requested for a non-input (or non-output) device.

(d) The current device answer-back is different from the type that existed at the time the current device table was created (that is, the table was built for another type of device).

COMM SYSDEV ERROR

I/O was requested for a device which is used by AVS/EASY.

COMM BUFFER ERROR

The comm buffer overflows on the current output operation. This occurs if the device width is set larger than 133, or if a long prompt (somewhat less than 133, depending on continuation sequence settings) is issued.

APPENDIX B

APL OVERSTRIKE CHARACTERS

⍱	~	∨
⍲	~	∧
⍟	*	○
!	'	.
⌿	/	-
⍀	\	-
⌽	○	|
⊖	○	-
⍉	○	\
⍋	∆	|
⍒	∇	|
⍝	∩	∘
⍎	⊥	∘
⍪	,	‾
⌹	⎕	÷
⍞	⎕	'
$	|	*S*
⍕	∘	⊤
⌷	[	]
⊂∘	⊂	∘

APPENDIX C

MNEMONIC REPRESENTATIONS - SORTED BY CHARACTER

_	$UL	UNDERLINE
∆	$DT	DELTA
⎕	$QD	QUAD
¯	$NG	NEG
<	$LT	LESS THAN
≤	$LE	LESS OR EQUAL
=	$EQ	EQUAL
≥	$GE	GREATER OR EQUAL
>	$GT	GREATER THAN
≠	$NE	NOT EQUAL
∨	$OR	OR
∧	$AN	AND
⍱	$NR	NOR
⍲	$ND	NAND
×	$ML	MULTIPLY
÷	$DV	DIVIDE
⍟	$LG	LOG
⌊	$MN	MINIMUM
⌈	$MX	MAXIMUM
|	$MD	MODULUS
!	$EX	EXCLAMATION
○	$CI	CIRCLE
~	$TL	TILDE
?	$QU	QUERY
⌿	$SM	SLASH*
\	$BS	BACKSLASH
⍀	$BT	BACKSLASH*
↑	$TA	TAKE
↓	$DR	DROP
⊥	$BV	BASE VALUE
⊤	$RP	REPRESENTATION
∊	$EP	EPSILON
⍳	$IO	IOTA
⍴	$RO	RHO
⌽	$RT	ROTATE
⊖	$RU	ROTATE*
⍉	$TP	TRANSPOSE
←	$IS	IS
;	$SC	SEMICOLON
→	$GO	GOTO
⍋	$UG	UPGRADE
⍒	$DG	DOWNGRADE
[	$OB	OPEN BRACKET
]	$CB	CLOSE BRACKET
∘	$NL	NULL
⍝	$LP	LAMP
'	$QT	QUOTE
:	$CL	COLON
∇	$DL	DEL
⍺	$AL	ALPHA
⍵	$OM	OMEGA
∩	$IX	INTERSECTION
∪	$UN	UNION
⊃	$IN	INCLUSION
⊂	$ID	IMBED
¨	$DQ	DOUBLE QUOTE
⍎	$EV	EVALUATE
⍪	$CN	COMMA*
⌹	$XD	MATRIX DIVIDE
⍞	$QP	QUAD-PRIME
$	$DO	DOLLAR
⍕	$FM	FORMAT
▯	$BC	BAD CHARACTER
[illegible]	$CO	CONTINUATION

** - ALONG 1ST CO-ORDINATE*

MNEMONIC REPRESENTATIONS - SORTED BY MNEMOMIC

$AL	⍺	ALPHA	$LG	⍟	LOG
$AN	∧	AND	$LP	⍝	LAMP
$BC	⌷	BAD CHARACTER	$LT	<	LESS THAN
$BS	\	BACKSLASH	$MD	∣	MODULUS
$BT	⍀	BACKSLASH*	$ML	×	MULTIPLY
$BV	⊥	BASE VALUE	$MN	⌊	MINIMUM
$CB	]	CLOSE BRACKET	$MX	⌈	MAXIMUM
$CI	○	CIRCLE	$ND	⍲	NAND
$CL	:	COLON	$NE	≠	NOT EQUAL
$CN	⍪	COMMA*	$NG	¯	NEG
$CO	⍺	CONTINUATION	$NL	∘	NULL
$DG	⍒	DOWNGRADE	$NR	⍱	NOR
$DL	∇	DEL	$OB	[	OPEN BRACKET
$DO	$	DOLLAR	$OM	⍵	OMEGA
$DQ	¨	DOUBLE QUOTE	$OR	∨	OR
$DR	↓	DROP	$QD	⎕	QUAD
$DT	∆	DELTA	$QP	⍞	QUAD-PRIME
$DV	÷	DIVIDE	$QT	'	QUOTE
$EP	∊	EPSILON	$QU	?	QUERY
$EQ	=	EQUAL	$RO	⍴	RHO
$EV	⍎	EVALUATE	$RP	⊤	REPRESENTATION
$EX	!	EXCLAMATION	$RT	⌽	ROTATE
$FM	⍕	FORMAT	$RU	⊖	ROTATE*
$GE	≥	GREATER OR EQUAL	$SC	;	SEMICOLON
$GO	→	GOTO	$SM	⌿	SLASH*
$GT	>	GREATER THAN	$TA	↑	TAKE
$ID	⊂	IMBED	$TL	~	TILDE
$IN	⊃	INCLUSION	$TP	⍉	TRANSPOSE
$IO	⍳	IOTA	$UG	⍋	UPGRADE
$IS	←	IS	$UL	_	UNDERLINE
$IX	∩	INTERSECTION	$UN	∪	UNION
$LE	≤	LESS OR EQUAL	$XD	⌹	MATRIX DIVIDE

* - ALONG 1ST CO-ORDINATE

APPENDIX D

APL SYSTEM CHARACTER CODE VALUES

0:0	30:*T*	60:|	90:⍝
1:1	31:*U*	61:!	91:'
2:2	32:*V*	62:○	92::
3:3	33:*W*	63:~	93:*V*
4:4	34:*X*	64:?	94:⍺
5:5	35:*Y*	65:⌿	95:⍵
6:6	36:*Z*	66:/	96:∩
7:7	37:∆	67:\	97:∪
8:8	38:⎕	68:⍀	98:⊃
9:9	39	69:↑	99:⊂
10:_	40:.	70:↓	100:¯
11:*A*	41:¯	71:⊥	101:⍎
12:*B*	42:<	72:⊤	102:⍕
13:*C*	43:≤	73:∊	103:⌹
14:*D*	44:=	74:⍳	104:⍞
15:*E*	45:≥	75:,	106:$
16:*F*	46:>	76:⍴	107:⍫
17:*G*	47:≠	77:⌽	108:⌷
18:*H*	48:∨	78:⊖	¯1:*CO*
19:*I*	49:∧	79:⍉	¯2:*NL*
20:*J*	50:⍱	80:←	¯3:*BS*
21:*K*	51:⍲	81:;	¯4:*IDL*
22:*L*	52:+	82:→	¯5:*CRR*
23:*M*	53:-	83:⍋	¯6:*FF*
24:*N*	54:×	84:⍒	¯7:*SO*
25:*O*	55:÷	85:[	¯8:*SI*
26:*P*	56:⋆	86:]	¯9:*EOT*
27:*Q*	57:⍟	87:(	¯10:*BOT*
28:*R*	58:⌊	88:)	
29:*S*	59:⌈	89:∘	

NOTE: NEGATIVE CODES ARE CONTROL CHARACTERS

APPENDIX E

IBM 2741 CORRESPONDENCE TRANSMISSION CODES

ARBOUT VALUE

ROW+16×*COL*

	0	1	2	3	4	5	6	7
0	0 sp	16 *T*	32 ←	48 *J*	64 sp	80 ~	96 →	112 ∘
1	1 1	17 *X*	33 *M*	49 *G*	65 ¨	81 ⊃	97 |	113 ∇
2	2 2	18 *N*	34 ∘	50 ×	66 ¯	82 ⊤	98 :	114 ÷
3	3 3	19 *U*	35 *V*	51 *F*	67 <	83 ↓	99 ∪	115 _
4	4 5	20 *E*	36]	52 *P*	68 =	84 ∊	100)	116 *
5	5 7	21 *D*	37 *R*	53 [	69 >	85 ⌊	101 ⍴	117 (
6	6 6	22 *K*	38 *I*	54 *Q*	70 ≥	86 ∘	102 .	118 ?
7	7 8	23 *C*	39 *A*	55 ,	71 ≠	87 ∩	103 α	119 ;
8	8 4	24 *L*	40 *O*	56 /	72 ≤	88 ⎕	104 0	120 \
9	9 0	25 *H*	41 *S*	57 *Y*	73 ∧	89 Δ	105 ⌈	121 ↑
10	10 *Z*	26	42	58	74 ⊂	90	106	122
11	11 9	27 *B*	43 *W*	59 +	75 ∨	91 ⊥	107 ω	123 -
12	12	28	44	60	76	92	108	124
13	13	29 lf	45 cr	61 tab	77	93 lf	109 cr	125 tab
14	14 uc	30	46 bs	62 lc	78 uc	94	110 bs	126 lc
15	15 eot	31	47 idl	63	79 eot	95	111 idl	127

APPENDIX F

IBM 2741 BCD TRANSMISSION CODES

	0	1	2	3	4	5	6	7
0	0 *SP*	16 ←	32 +	48 ×	64 *SP*	80 →	96 -	112 ⍭
1	1 1	17 /	33 *J*	49 *A*	65 ¨	81 \	97 ∘	113 ⍺
2	2 2	18 *S*	34 *K*	50 *B*	66 ¯	82 ⌈	98 '	114 ⊥
3	3 3	19 *T*	35 *L*	51 *C*	67 <	83 ~	99 ⎕	115 ∩
4	4 4	20 *U*	36 *M*	52 *D*	68 ≤	84 ↓	100 |	116 ⌊
5	5 5	21 *V*	37 *N*	53 *E*	69 =	85 ∪	101 ⊤	117 ∊
6	6 6	22 *W*	38 *O*	54 *F*	70 ≥	86 ⍵	102 0	118 _
7	7 7	23 *X*	39 *P*	55 *G*	71 >	87 ⊃	103 *	119 ∇
8	8 8	24 *Y*	40 *Q*	56 *H*	72 ≠	88 ↑	104 ?	120 ∆
9	9 9	25 *Z*	42 *R*	57 *I*	73 ∨	89 ⊂	105 ⍴	121
10	10 0	26	42	58	74 ∧	90	106	122
11	11]	27 ,	43 [	59 .	75)	91 ;	107 (	123 :
12	12	28	44	60	76	92	108	124
13	13	29 lf	45 cr	61 tab	77	93 lf	109 cr	125 tab
14	14 uc	30	46 bs	62 lc	78 uc	94	110 bs	126 lc
15	15 eot	31	47 idl	63	79 eot	95	111 idl	127

ARBOUT VALUE

ROW+16×*COL*

APPENDIX G

APL/ASCII TYPEWRITER-PAIRING TRANSMISSION CODES

	0	1	2	3	4	5	6	7
0	0 *NUL*	16 *DLE*	32 *SP*	48 0	64 ¯	80 *	96	112 *P*
1	1 *SOH*	17 *DC*1	33 ¨	49 1	65 α	81 ?	97 *A*	113 *Q*
2	2 *STX*	18 *DC*2	34)	50 2	66 ⊥	82 ρ	98 *B*	114 *R*
3	3 *ETX*	19 *DC*3	35 <	51 3	67 ∩	83 ⌈	99 *C*	115 *S*
4	4 *EOT*	20 *DC*4	36 ≤	52 4	68 ⌊	84 ~	100 *D*	116 *T*
5	5 *ENQ*	21 *NAK*	37 =	53 5	69 ∊	85 ↓	101 *E*	117 *U*
6	6 *ACK*	22 *SYN*	38 >	54 6	70 _	86 ∪	102 *F*	118 *V*
7	7 *BEL*	23 *ETB*	39]	55 7	71 ∇	87 ω	103 *G*	119 *W*
8	8 *BS*	24 *CAN*	40 ∨	56 8	72 ∆	88 ⊃	104 *H*	120 *X*
9	9 *HT*	25 *EM*	41 ∧	57 9	73	89 ↑	105 *I*	121 *Y*
10	10 *LF*	26 *SUB*	42 ≠	58 (	74 ∘	90 ⊂	106 *J*	122 *Z*
11	11 *VT*	27 *ESC*	43 ÷	59 [	75	91 ←	107 *K*	123 [
12	12 *FF*	28 *FS*	44 ,	60 ;	76	92	108 *L*	124
13	13 *CR*	29 *GS*	45 +	61 ×	77 |	93 →	109 *M*	125)
14	14 *SO*	30 *RS*	46 ∘	62 :	78 ⊤	94 ≥	110 *N*	126 *S*
15	15 *SI*	31 *US*	47 /	63 \	79 ○	95 -	111 *O*	127 *DEL*

ARBOUT VALUE

ROW+16×*COL*

APPENDIX H

APL/ASCII BIT-PAIRING TRANSMISSION CODES

	0	1	2	3	4	5	6	7
0	0 *NUL*	16 *DLE*	32 *SP*	48 0	64 ←	80 ⋆	96 →	112 *P*
1	1 *SOH*	17 *DC1*	33 ¨	49 1	65 α	81 ?	97 *A*	113 *Q*
2	2 *STX*	18 *DC2*	34 ¯	50 2	66 ⊥	82 ρ	98 *B*	114 *R*
3	3 *ETX*	19 *DC3*	35 <	51 3	67 ∩	83 ⌈	99 *C*	115 *S*
4	4 *EOT*	20 *DC4*	36 ≤	52 4	68 ⌊	84 ~	100 *D*	116 *T*
5	5 *ENQ*	21 *NAK*	37 =	53 5	69 ∊	85 ↓	101 *E*	117 *U*
6	6 *ACK*	22 *SYN*	38 ≥	54 6	70 _	86 ∪	102 *F*	118 *V*
7	7 *BEL*	23 *ETB*	39 >	55 7	71 ∇	87 ω	103 *G*	119 *W*
8	8 *BS*	24 *CAN*	40 ≠	56 8	72 ∆	88 ⊃	104 *H*	120 *X*
9	9 *HT*	25 *EM*	41 ∨	57 9	73	89 ↑	105 *I*	121 *Y*
10	10 *LF*	26 *SUB*	42)	58]	74 ∘	90 ⊂	106 *J*	122 *Z*
11	11 *VT*	27 *ESC*	43 (	59 [	75	91 -	107 *K*	123 -
12	12 *FF*	28 *FS*	44 ,	60 ;	76 ⎕	92	108 *L*	124 *S*
13	13 *CR*	29 *GS*	45 +	61 -	77 |	93 [	109 *M*	125]
14	14 *SO*	30 *RS*	46 .	62 :	78 ⊤	94 ×	110 *N*	126 ÷
15	15 *SI*	31 *US*	47 /	63 \	79 ○	95 ∧	111 *O*	127 *DEL*

ARBOUT VALUE

ROW+16×*COL*

APPENDIX I

SCI-1205/1210 RS232C INTERFACE SIGNALS

(25 Pin Male Connector)

PIN NO.	SIGNAL	FUNCTION
1	Protective Ground (AA)	Chassis Ground
*2	Transmitted data (BA)	Digital Input to Computer
*3	Receive data (BB)	Digital Output from Computer
*4	Request to send (CA)	Control output from computer On when computer is ready to transmit data
*5	Clear to Send (CB)	Control input to computer On when modem is ready to transmit data
*6	Data set ready (CC)	Control input to computer On when data set is ready
7	Signal ground (AB)	Common signal and power ground return
*8	Data carrier detect (CF)	Control input to computer On when modem has carrier
9	Spare	None
10	Spare	None
**11	PROMPT	When on, system can issue output to device designated as input
**12	Aux Data Terminal Ready	Aux control input to computer On simulates DTR for a terminal or DSR, DCD and CTS for a modem
13	Spare	None
##14	Received Data (TTY)	Digital input from TTY
15	Spare	None
##16	Transmitted Data (TTY)	Digital output to TTY

17	Spare	None
##18	Received Data Return (TTY)	Common return for receive data
**19	TTY Enable	Control input to computer On enables TTY
*20	Data Terminal Ready (CD)	Control output from computer On indicates computer is ready (forces CTS, DSR, DCD)
*21	Monitor Data	Output data from computer Monitors transmitted and received data
22	Spare	None
23	Spare	None
24	Spare	None
##25	Transmitted Data Return (TTY)	Common return for TTY transmitted data

SIGNAL LEVELS

* These are EIA RS232C signals with the following characteristics:

Inputs:	Mark -3 to -25 Volts	
	Space 3 to 25 Volts	
Outputs:	Mark -9 with 2K Load	
	Space 9 with 2K Load	

** These lines are enabled when grounded (connected to pin 7)

These are 20 ma current loop signals.

APPENDIX J

SCI-1200 RS232C MODEM INTERFACE SIGNALS

PIN NO.	SIGNAL	FUNCTION
1	Protective Ground (AA)	Chassis ground
*2	Transmitted data (BA)	Digital input to computer
*3	Receive Data (BB)	Digital output from computer
*4	Request to send (CA)	Control output from computer On when computer is ready to transmit data
*5	Clear to send (CB)	Control input to computer On when modem is ready to transmit data
*6	Data set ready (CC)	Control input to computer On when data set is ready
7	Signal ground (AB)	Common signal and power ground return
*8	Data carrier detect (CF)	Control input to computer On when modem has carrier
**11	PROMPT	When on, system can issue output to device designated as input
*20	Data Terminal Ready (CD)	Control output from computer On indicates computer is ready (forces CTS, DSR, DCD)

SIGNAL LEVELS

* These are EIA RS232C signals with the following characteristics:

Inputs: Mark -3 to -25 Volts
Space 3 to 25 Volts
Outputs: Mark -9 with 2K Load
Space 9 with 2K Load

** These lines are enabled when grounded (connected to pin 7)

APPENDIX K

SCI-1200 RS232C PRINTER/TERMINAL INTERFACE SIGNALS

PIN NO.	SIGNAL	FUNCTION
1	Protective Ground (AA)	Chassis ground
*2	Transmitted data (BA)	Digital input to computer
*3	Receive data (BB)	Digital output from computer
*5	Clear to send (CB)	Control input to computer ON when modem is ready to transmit data
*6	Data set ready (CC)	Control input to computer ON when data set is ready
7	Signal ground	Common signal and power ground return
*8	Data carrier detect (CF)	Control input to computer ON when modem has carrier
**11	PROMPT	When on, system can issue output to device designated as input
**12	Aux Data Terminal Ready	Aux control input to computer On indicates DTR
**20	Data Terminal Ready (CD)	Control output from computer ON indicates computer is ready (forces CTS, DSR, DCD)

SIGNAL LEVELS

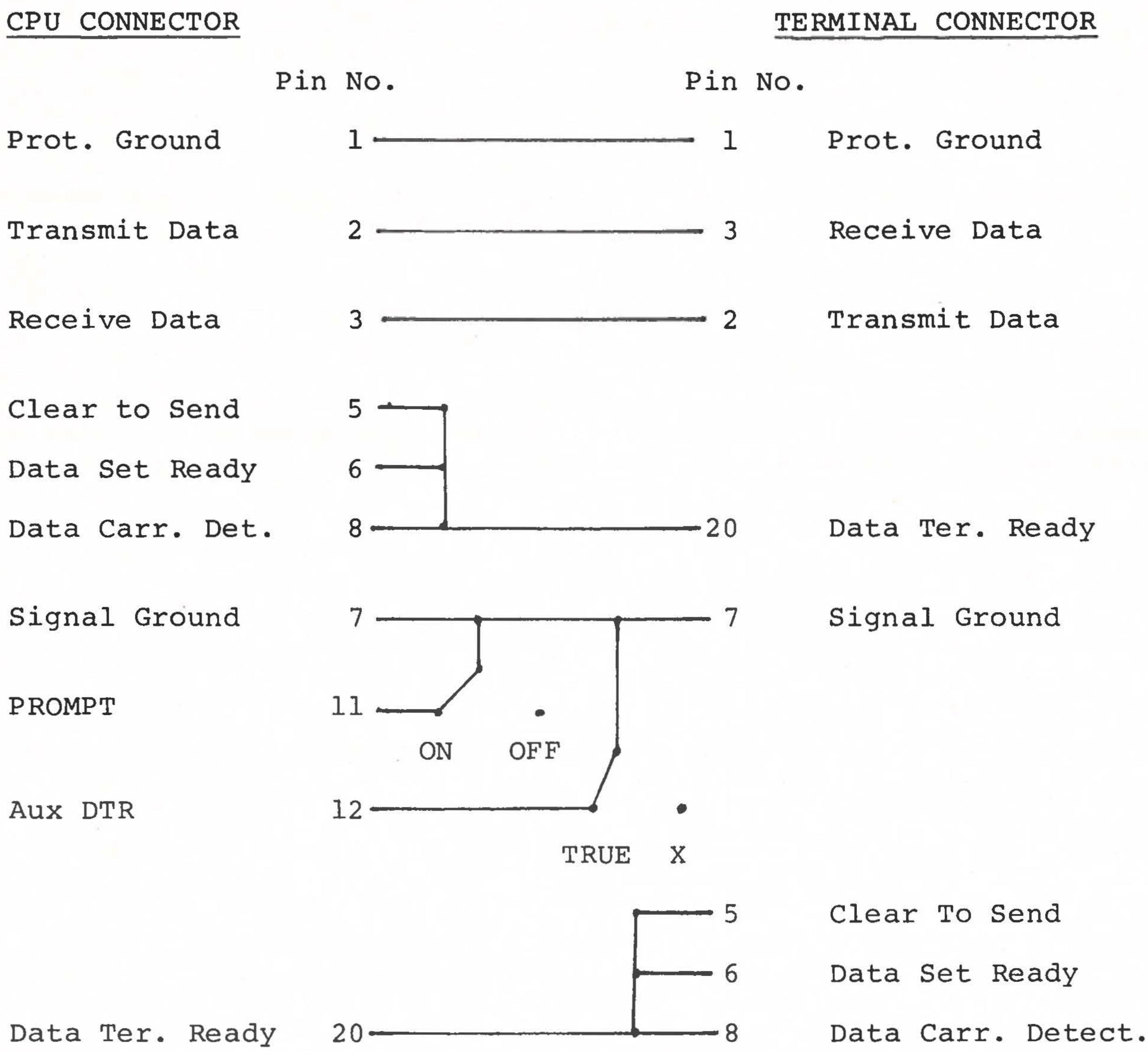
* These are EIA RS232C signals with the following characteristics:

Inputs:	Mark	-3 to -25 Volts
	Space	3 to 25 Volts
Outputs:	Mark	-9 with 2K Load
	Space	9 with 2K Load

** These lines are enabled when grounded (connected to pin 7)

APPENDIX L

TERMINAL ADAPTER BOX
(SCI-1205/1210 ONLY)



APPENDIX M

TELETYPE CURRENT LOOP ADAPTER

COMPUTER	Pin No.	TELETYPE
Prot. Ground	1	Prot. Ground
Signal Ground	7	
PROMPT	11	
Aux DTR	12	
TTY Enable	19	
RXD (TTY)	14	+ Input from TTY
RXD Return	18	- Input return
TXD (TTY)	16	+ Output to TTY
TXD Return	25	- Output return