APPLE TELL TECHNICAL DOCUMENTATION

Author: Francois Grieu

History:

23 Dec. 83 - Pre-history.

4 Jan. 84 - MODESLCT specification changed, to reflect that it uses the SETRTS routine.

5 Jan. 84 - SETRTS specification changed, to enable listening to the back carrier; also, the routine no longer clears the FIFO nor receive error conditions, to avoid the loose of characters in half duplex operation.

6 Jan. 84 - Memory usage of MODEMIO routines added.

13 Jan. 84 - A minor error corrected in EMITSTAT and RINGTEST specifications - sorry.

23 Jan. 84 - The first cards have been sent to devellopers, with ROMs that does NOT follow these specifications AT ALL, but are rather of the much older type we used during the hardware conception. This has been done because I have not been able to release ROM rev. 1.2 in time...

29 Jan. 84 - ROM 1.2 comes to life.
- Introduction of the firmware issue number at location $CsFE. Table 4 modified accordingly.
- State that N and Z flags are set according to A on exit of a MODEMIO routine that returns a result.
- ASCDIAL timing stated in it's specifications.

2 Feb. 84 - With ROM 1.3, it's realy too difficult to detect a carrier while providing a timeout. To make things simpler, I introduce SMARTCD. Here comes ROM 2.0
- Information about answer modes clarified.

13 Feb. 84 - Changes have been made in the ROM, wich is now ROM 3.1, and have been widely sent to software devellopers. This version includes a dumb terminal in the ROM, wich is copied in RAM prior to use. Also, the default data format in this ROM 3.1 is 7 data bits + even parity + 1 stop bit, for some obscure reasons. Despite of the ROM 3.1 sticker, the internal revision number (at locations $CsFE, $CsFF) is 2.0, due to an error at assembly time.

17 Feb. 84 - SMARTCD specifications changed (introduction of the detection time parameter).
- ASCDIAL timing corrected (that was a huge bug).
- CLEANUP routine added.
- Serial number added.

19 Feb. 84 - BIGBELL routine added, to facilitate ring detection.
- RINGTEST specifications and code slightly changed.

20 Feb. 84 - A new version of the dumb terminal is now in the ROM. This one works into the ROM itself.
- Revision number is now 4.0.

3 Mar. 84 - A bug in the 8530 initialisation has been fixed. The so-called unused bits in WR03 must be nulls to insure that the receiver realy samples the data bits in their middle, avoiding a problem when receiving NULLs and SOHs (much usefull for binary transfers).
- The AMD recognises that 'some' of his 'early' Am7910 rev D chips are subject to a latchup of the RD pin. The manufacturer promises to prevent, not test out, the problem in a future rev D-1 of the chip, and provides a magic formula to clear up the problem: select CCITT V23 back loopback with the reset line low, wait, pull up reset, wait, assert DTR, wait (all waits more that 1.5 ms). MODESLCT does it and yeah, it works !!!
- Revision number is now 4.1 (dumb terminal text in English) and 4.2 (French).

This documentation is written to give the software developer the information needed to use the Apple-Tell card in its own application. It includes information about the hardware (from a programmer's point of view), and on the built-in firmware.

OVERVIEW OF THE APPLE-TELL CARD

It's an add-on for the Apple][(or][+ or][e). It provides on a single board:

- a communication section to enable the Apple to transmit and receive data to or from a nother computer, through a normal phone line. There are:

  - a direct connect phone line interface with pulse dialing and ring detection capacities

  - a modem that supports a wide range of modulation systems:
    - Bell 103 (up to 300 bps full duplex)
    - CCITT V.21 (up to 300 bps full duplex)
    - CCITT V.23 mode 1 (600 bps half duplex)
    - CCITT V.23 mode 2 (1200 bps with a 75 bps back channel)
    - Bell 202 (1200 bps with a 5 bps back channel)

    Both originate and answer modes are supported, as well as carrier detection, answer protocol and loopback modes.

  - a smart UART featuring:
    - 2 independent channels, each with a programmable baud rate generator and handshake lines
    - programmable character format, parity generation/detection
    - extended interrupt capacities, as well as a pooling mode
    - hardware reception FIFO buffer (3 characters + character being received) on each channel, making reception timing much less critical

  All these communication parameters are selectable by software: there is no switch on the board.

- a Videotex section that drives an external VDU - the Apple black & white monitor, or an external black & white or color monitor (or T.V. set with direct video input) - . It can display characters and mid-resolution graphics with a wide range of sizes, colors, underlining... In Europe, this video display is standard for general public remote terminals, for applications such as phone ordering, telephone diary, stock exchange rates, transport schedules...

  IMPORTANT: There is a low-cost version of the Apple-Tell card that does not include this section. Because it is smaller, this version can fit a regular Apple III.

- a firmware section with, in a 2K byte memory, the software that supports the modem capacities. The original 2716 EPROM can, for development purposes, be replaced by a 6116 RAM (or equivalent).

In the Videotex version, the Apple-Tell card comes with a software package to emulate the popular Minitel (a Videotex remote terminal), and adds some very important functions to it such as auto-dial, macro capacities with a unique auto-learn mode, screen capture, text capture, interface with other software...

HARDWARE DESCRIPTION

WARNING: The scope of this paragraph is to describes how the various LSIs and functional blocks in the Apple-Tell card interact with each other,and does not explain their inner functioning. The reader can refer to the chip's technical specifications and to the detailed plans (see appendix). Anyway, complete knowledge of the hardware is not necessary to use it with the built-in firmware.

Figure 1 shows a diagram of the hardware, simplified in the sections that are of no interest to the programmer.

The Am28530 is the heart of the system, providing serial communication functions, control over the various devices of the board, and clock generation for the baud rate generators and the modem. The Am7910 is a state-of-the-art FSK modem circuit, with on-chip filters, externally programmable to match a lot of standards. These chips communicate through two channels (A / Main, B / Back), each with the standard EIA signals TD, RD, RTS', CTS', CD'. The back channel is used only in CCITT V.23 mode 2 (for character transmission at a slow 75 bps rate, e.g. in a Videotex context the characters entered from the keyboard) and in Bell 202 (for transmission of a single bit such as a busy or request signal represented by the presence or absence of a low frequency carrier, at a slow 5 bps rate).

The (optional) Videotex Display Generator is built around two dedicated VLSIs (EF9341 / EF9340) and 2K of static RAM. When using a B&W monitor, the video source (Apple video or Videotex) can be selected by software.

The interface with the expansion slot bus maps the various addresses / memory locations as follow:

The card is assumed to be in slot s (s = 1 to 7).
All addresses are given in hexadecimal.
The hardware is made so that any valid 6502's addressing mode can be used to access any register.

$C080 + $s0	TRA	9341 Register
$C081 + $s0	TRB	9341 Register
$C082 + $s0	CRA	9341 Register
$C083 + $s0	CRB	9341 Register
$C088 + $s0	RRB	8530 Read Register, channel B
$C089 + $s0	RRA	8530 Read Register, channel A
$C08A + $s0	RBB	8530 Receive Buffer, channel B
$C08B + $s0	RBA	8530 Receive Buffer, channel A
$C08C + $s0	WRB	8530 Write Register, channel B
$C08D + $s0	WRA	8530 Write Register, channel A
$C08E + $s0	TBB	8530 Transmit Buffer, channel B
$C08F + $s0	TBA	8530 Transmit Buffer, channel A

$Cs00...$CsFF	Peripheral-card ROM space contains addresses $700...$7FF of the 2716 EPROM accessing these addresses sets the ROM-enable flip-flop
$C800...$CEFF	Expansion ROM space contains addresses $000...$6FF of the 2716 EPROM only if the ROM-enable flip-flop is enabled
$CF00...$CFFF	Accessing these addresses resets the ROM-enable flip-flop

Figure 2

When the 8530 generates an interrupt (that will occur only if it is programmed to do so), it will pull the Apple's IRQ' line low. The daisy-chained interrupt facility described by Apple (Apple IIe reference manual, page 170) is fully supported. The ability of the 8530 to generate an hardware interrupt vector can't be used in a 6502 system, therefore the INTACK' line of the 8530 is held high to disable this feature.

When the Apple is resetted (on power-on, or by pressing Ctrl-Reset), the 8530 is put in a reset state; that hangs up the phone line and selects the Apple's Video. Of course, the entire modem circuitry must be re-initialised before use. The Videotex section is also resetted.

A 7 bit binary counter is used to select a communication mode for the 7910, reset it, and assert DTR'. To prepare the 7910 to operate in a given MODE (found in figure 3), one must:

- Make sure the 8530 generates a stable 2.4576 Mhz clock on the CLK' line of the 7910.
- Apply a negative pulse on the W/REQB' line of the 8530. That resets the counter.
- Apply MODE*4+3 negative pulses on the W/REQA' line of the 8530. The first MODE*4 pulses position the MC4...MC0 lines of the 7910 to the binary value of MODE, the next 2 pulses drive the RESET' line of the 7910 from low to high, thus putting it in operating mode, waiting for a Data Terminal Ready condition. The last pulse makes this high-to-low transition on the DTR' line of the 7910.

MODE		
hex	binary	
		normal modes
00	00000	Bell 103 originate 300bps full duplex
01	00001	Bell 103 answer 300pbs full duplex
02	00010	Bell 202 1200bps half duplex (*)
03	00011	Bell 202 with equalizer 1200 bps half duplex (*)
04	00100	CCITT V.21 originate 300bps full duplex
05	00101	CCITT V.21 answer 300bps full duplex
06	00110	CCITT V.23 mode 2 1200 bps half duplex (**)
07	00111	CCITT V.23 mode 2 with equalizer 1200 bps half duplex (**)
08	01000	CCITT V.23 mode 1 600 bps half duplex
		loopback (local test) modes
10	10000	Bell 103 originate loopback
11	10001	Bell 103 answer loopback
12	10010	Bell 202 main loopback
13	10011	Bell 202 with equalizer loopback
14	10100	CCITT V.21 originate loopback
15	10101	CCITT V.21 answer loopback
16	10110	CCITT V.23 mode 2 main loopback
17	10111	CCITT V.23 mode 2 with equalizer loopback
18	11000	CCITT V.23 mode 1 main loopback
19	11001	CCITT V.23 back loopback (**)

(*) in this mode, there is a 5bps back channel used to transmit a status bit, which is physically represented by the presence or absence of a back carrier.

(**) in this mode, there is a 75bps back channel used to transmit characters, by FSK modulation of a back carrier. In a Videotex application, this back channel usually transmits the characters entered from a keyboard.

Figure 3

FIRMWARE DESCRIPTION

The 2K of firmware are designed:
- To provide the user with facilities such as a terminal program, PR# and IN# support...
- To free the software developer from the cumbersome task of programming the modem I/O directly at the 8530 register level. Instead, most modem functions can be performed by calling a routine in the built-in EPROM. Operation is, as much as possible, the same regardless of the communication standard.

This firmware supports the standard Apple peripheral card firmware specifications, as regard to the Basic protocol and Pascal 1.1 protocol. It must alas be noted that these protocols don't include the features necessary for a modem application, and that non-standard features had to be added, for example to support operating mode selection and carrier detection.

In this revision $04 of the firmware, operation is supported only in a pooling mode (the 8530 never generates interrupt; the application program must constantly look if a character is being received, if not, some characters may be lost).

Recognizing the Apple-Tell card (see figure 4)

As described in the 'Apple][e Design Guidelines' manual, page 6, a program can check if there is an Apple-Tell card in slot 's' by reading address $Cs05, $Cs07, $Cs0B, $Cs0C.
The firmware identification bytes will permit a future program to check if the firmware supports the needed features (versions should be upward compatible). We increment the firmware revision number when we add capabilities to the firmware, with changes in the present specifications.
The firmware issue number distinguishes firmwares with the same revision number, thus the same specifications. We increment it each time we fix the last bug...
The serial number is there so that software can recognise a particular board.
The hardware features bytes identify the current an future hardware options present on the board, so that a program can recognize the hardware configuration.

```
Address        Value
 $Cs05     (*)  $38  ; Standard for all firmware cards
 $Cs07     (*)  $18  ; Standard for all firmware cards
 $Cs0B     (*)  $01  ; Standard for all firmware cards
 $Cs0C     (*)  $49  ; Apple-Tell card generic modem signature
 $CsF6          $00  ; Hardware features - low byte (reserved for future use)
 $CsF7          $00  ; Hardware features (reserved for future use)
 $CsF8          $00  ; Hardware features (reserved for future use)
 $CsF9          $E8  ; Hardware features - high byte
                x.....  ; card includes a 9340/9341 VDU
                .x.....  ; card includes a Am7910 World Chip
                ..x.....  ; card includes a Z8530 SCC
                ...x....  ; card includes a Z8531 SCC
                ....x...  ; card can pulse-dial
                .....x..  ; card can tone-dial
                ......x.  ; card includes a 'peri-informatique' interface
                .......x  ; card includes a 'smart card' interface
 $CsFA          ???  ; Serial number - low byte
 $CsFB          ???  ; Serial number
 $CsFC          ???  ; Serial number
 $CsFD          ???  ; Serial number - high byte
 $CsFE          $00  ; Firmware issue number
 $CsFF          $04  ; Firmware revision number
```

(*) The value of these bytes -and only these- will never change.

Figure 4

The main MODEM I/O routine

At location $Cs11 stands a very important entry point of the firmware, labelled MODEMIO. Through this entry point, all the modem control functions offered by the firmware are accessible.

On entry, the 6502 registers must contain:

Y register code number of the operation to be performed
A register parameter (if one or more needed)
X register parameter (if two needed)

On exit, there will be:

A register result (if applicable).
When the routine returns a result in A, the N and Z status bits are set according to the contents of A.
The decimal flag (D) is returned clear (normal mode).
The interrupt status flag (I) remains unchanged throughout the call.
X and Y register and other status bits are usually disturbed.

The MODEMIO routines use the following memory locations:

$07F8 MSLOT contains $Cs at any time MODEMIO uses the expansion ROM space, and on exit of any MODEMIO routine. This is to be compatible with the standard way of restoring the expansion ROM status after an IRQ or NMI handled by an other peripheral card using the expansion ROM space.
$0478+s (*) current MODE.
$04F8+s (*) current FORMAT.
$0578+s (*) EMITCH emit channel: $s1 for main channel, $s0 for back channel.
$05F8+s (*) RECVCH receive channel, same as above.
$0678+s (*) MISCBITS ....x RTSA (set <=> sending carrier on main channel)
                      ......x. RTSB (set <=> sending carrier on back channel)
                      ..xxxx.. reserved for future use.
                      .x...... DTRA (set <=> line picked up).
                      x....... DTRB (set <=> videotex video selected)
$06F8+s Used for temporary storage.
$0778+s Reserved for future use.
$07F8+s Reserved for future use.

(*) Though it is not recommended, the user program can read this location, but must never modify it.

No other RAM space is used by MODEMIO (exept some stack space, of course).

MODEMIO provides the following functions:

- INITUART

The first MODEMIO routine called after an hardware reset (including power-up) must be INITUART. This routine:
- Executes a 'Force Hardware Reset' on the 8530 circuit (that hangs up the phone line).
- Programs it to generate the clock for the 7910.
- Sets the standard character format (7 data bits, no parity, one stop bit).
- Falls into MODESLOT.

Input: Y = $00
        A = MODE to be selected - found in figure 3
Output: None

- MODESLCT
  - Sets the baud rate generators (to the maximum supported rate) according to the MODE selected.
  - Programs the 7910 to this mode, resets it and asserts DTR.
  - Falls into SETRTS with the parameter A = $00. That puts the modem in the state where it sends no carrier, and make the DCD bit returned by RECVSTAT reflect the status of the main carrier.

This routine can be used with the phone line picked up, but then transient noises may be sent on the phone line.

Input: Y = $01
A = MODE to be selected - found in figure 3
Output: None

- DTAFORMT

This routine selects a data format.

Input: Y = $02
A = data format, for both transmitter and receiver
.....x0 no parity
.....01 odd parity
.....11 even parity
....00.. prohibited
....01.. 1 stop bit
....10.. 1.5 stop bit
....11.. 2 stop bits
...0.... normal operation
...1.... send BREAK
.00..... 5 bits/character
.10..... 6 bits/character
.01..... 7 bits/character
.11..... 8 bits/character
x..... not defined
Output: none

- LINEREL

This routine controls the phone line relay.

Input: Y = $03
A = $00 hangs up (the reset state)
other picks up
Output: None

- VIDEOREL

This routine controls the video selection relay.

Input: Y = $04
A = $00 selects Apple video (the reset state)
other selects Videotex video
Output: none

- SETRTS
  - Waits till all characters being sent have completely cleared the transmitter, if applicable.
  - Choose the emit/receive channels and positions the RTS' and BRTS' lines, according to MODE and the parameter.
  - Waits till the modem acknowledges the change using the CTS' and BCTS' lines.

  Input: Y = $05
  A = $00 Makes RTS and BTRS inactive, thus sends no carrier. Selects main channel as receive channel in all modes; the RECVSTAT routine will thus return the main carrier status in the DCD bit.
  $01 asserts RTS, thus sends carrier over the main channel. Selects main channel as emit channel in all modes. Selects main channel as receive channel exept if MODE = $02, $03, $06 or $07, where back channel is used as receive channel. Invalid if MODE=$19
  $02 Valid only if MODE = $02, $03, $06, $07 or $19. Asserts BRTS, thus sends carrier over the back channel. Selects back channel as emit channel (although in modes $02 or $03, no character can be sent with the normal SENDCHAR routine). Selects main channel as receive channel, exept for mode $19 where back channel is the receive channel.
  $03 Valid only if MODE = $02, $03, $06, $07 or $19. Makes RTS and BRTS inactive, thus sends no carrier. Selects back channel as receive channel; the RECVSTAT routine will thus return the back carrier status in the DCD. This function has been added to provide a mean of monitoring the back carrier status without sending a carrier in the Bell 202 modes.

When MODE = $02 or $03 (Bell 202), the 5bps back channel bit is transmitted by calling SETRTS with A = $02 to send the back carrier, A = $00 to stop the back carrier.

- EMITSTAT

This routine returns the transmitter status.

  Input: Y = $06
  Output: A = $00 Transmitter busy
  A <> $00 Transmitter ready to accept data

- SENDCHAR
  - Waits till the transmit channel is ready to accept a character for output
  - Outputs it. If the RTS / BRTS status is consistent with the mode selected, the byte will be outputed from the modem.

  Input: Y = $07
  A = character to be outputed
  Output: none

- RECVSTAT

This routine returns the receiver status.

  Input: Y = $08
  Output: A = receiver's status bits

```
.......x 1 (=) a character is available in the receiver
.....00. always 0
....x... 1 (=) there is a receive carrier
...x.... 1 (=) parity error
..x..... 1 (=) receiver overrun error (FIFO overflow)
.x...... 1 (=) framing error (spike on the phone line)
x....... 1 (=) a Break sequence has been encountered
```

Note: to detect that the carrier belongs to another modem, not to an obscure ring or somebody's shouting at the other end, the DCD bit must be tested several times (say, for about 2 seconds). A long delay (see Am7910 product specification, table 3(b)) is provided by the modem between changes of this bit, so 256 samples of the DCD bit at 8ms intervals all with DCD=1 are a reliable clue that there is a carrier.

- READCHAR
  - Waits till a character is available in the receive channel.
  - Returns it in the accumulator.
  - Clears any receive error state.
  Input: Y = $09
  Output: A = character received (no masking is done)

- ASCDIAL
  - Picks up the phone line (no wait time is provided, see note 2).
  - Pulse-dials the number whose ASCII code is given on input. Characters other than 1234567890 have no effect. ASCII codes 1, 2, 3, 4, 5, 6, 7, 8, 9 and 0 induce 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10 pulses, each made of a 66ms period without current in the phone-line followed by a 33ms period with current.
  - Waits (whether the number was valid or not) for the time necessary between numbers in the pulse-dial system (700ms) with the line picked up.
  Input: Y = $0A
  A = ASCII code of the number to dial.
  The high order bit is irrelevant.
  Output: None
Note 1: On exit, the phone line is always picked up.
Note 2: The phone line must be picked up and a reasonable time (1..3 sec. in France) allowed before dialing. This can be done by calling this routine a few (2..5) times with an 'invalid' argument.

- RINGTEST
This routine tests (for about 290 ms) if the phone is ringing.
  Input: Y = $0B
  Output: A = $00 not ringing.
  A <> $00 it's ringing.

- ANSWER
This routine instructs the modem to enter its auto-answer mode by applying a negative pulse on RING'. See the Am7910 product specification for details.
  Input: Y = $0C
  Output: none
Note: This routine is of interest only in auto-answer applications. It should be called after the line as been picked-up, and prior to calling SETRTS. In the exemple of CCITT V.23 mode 2 (M = $06 or $07), this routine instructs the modem to disable the receiver/transmitter operation (CD and BCD forced off), set silence for 1.9 sec., emit an answer tone (2100hz) for 3 sec., set silence for about 13 ms., then re-enable the receiver/transmitter. This is the protocol proposed by the CCITT for establishing a communication in a videotex environment.

- SMARTCD
This routine will determine wether or not there is a stable carrier. It terminates in 3 cases:
(1) the ESC key is pressed (more precisely, if the KBD location reads $9B).
(2) it has run for more than the specified time period T1, and no carrier is beeing detected.
(3) there has been a stable carrier for a T2 period.
  Input: Y = $0D
  A = timeout parameter. The timeout period lasts about
  T1 = (A^ 2 + 3*A + 2) * 5 ms
  X = detection hysteresis parameter. The detection time is
  T2 = X * 10 ms (exept if X=0, in which case T2 = 2560 ms)
  Output: A = $00 routine terminated in cases (1) or (2)
  <> $00 routine terminated in case (3)
Note: remember that the World Chip itself delays the CD and BCD bits (refer to Am7910 product specification, table 3b), making SMARTCD usefull only when a 'long' carrier loose is expected (generaly at connect time, and to attempt to recover from an accidental spike over the line).

Programming guidelines

A typical session with a Bell 103 timesharing system, with auto dial, would be, from the application program's point of view:

- Find the Apple-Tell board in slot 's' and build a routine MODEMIO1 at a known location containing a JMP to MODEMIO (if you use an indirect JMP, make sure the vector does not cross a page boundary).
- Use INITUART to select Bell 103 originate (call MODEMIO1 with Y=$00, A=MODE=$00)
- Instruct the user to hang up his phone.
- Ask the user for the phone number.
- Pick up the phone line and wait for the dial tone by calling ASCDIAL 4 times with a $A0 in the accumulator.
- For each character in the phone number, put it in the accumulator and call ASCDIAL.
- Wait for a stable carrier: call SMARTCD with A = 76 (decimal) for a 30 sec. timeout and X = 170 (decimal) for a 1.7 sec. carrier detection delay. Don't consider these parameter as final, but rather experiment or obtain trustable information.
- Assert RTS by calling SETRTS with A=$01. Our modem now emits its carrier. Communication can begin.
- Enter a loop where simultaneously:

1) - RECVSTAT is called and:
   - In case DCD comes to be 0, the communication is stopped (optionally, check that it was not a short line dropout by calling SMARTCD).
   - If a character is available,
     - Optionally, check for an error condition and treat it (for example, flag that the character is to be output in inverse video).
     - Call READCHAR to get the character and possibly reset the error condition.
     - Make it available for the rest of the program.
   - Goto 1) and, most difficult, do this at the average speed of Baud_rate/10 = 30 loops per second, with a maximum wait of 3/(Baud_rate/10) = 0.1 seconds between two execution of the loop.

   Note: if RECVSTAT is not called and/or the 'receive character available' bit not tested prior to calling READCHAR, a risk exist to enter an infinite loop if the other modem never sends a character.

2) - When a character is to be sent:
   - Optionally, call EMITSTAT to check that the transmitter is ready. If it's not, some CPU time is available.
   - Send the character through SENDCHAR.

3) - Run its own, highly sophisticated application.

- To exit and hang up the line, either:
  - Induce the user to press Ctrl-Reset (a really huge sound, or displaying 'INITIALIZING MASTER DISK' and spinning the drive will do the job).
  - call LINEREL with A=$00.

Apart from the value of MODE when calling INITUART, the process would be the same for CCITT V.21 originate (exept the value of MODE when calling INITUART).

For operation in CCITT V.23 mode 2 with reception at 1200bps, SETCTS should be called with A=$02.

For answer modes (both Bell 103 and CCITT V.21) and CCITT V.23 mode 2 with emission at 1200bps, the program should:

- init the modem in the appropriate mode using INIMOD
- wait for a ring using RINGTEST
- pick up the line using LINEREL
- optionnaly, call the ANSWER routine to start the auto-answer sequence (this is standard in CCITT V.23 mode 2)
- call SETCTS with A = $01, thus send it's carrier
- wait for a carrier using SMARTCD.
- clear the receive FIFO by reading any available character up to 4 times.
- Communicate exactly as above.

For Bell 202 half duplex operation, please refer to the Am 7910 product specification, Figure 10. It's very difficult for us to experiment for we have no reference Bell 202 system, nor any documentation...

DUMB TERMINAL PROGRAM

This function is subject to changes in future revisions.

ROM 4.x incorporates a dumb terminal program, that is to say that emulates a terminal connected to a modem. It can operate in 40 or 80 columns mode.
To activate it, do an IN#s for 40 colums, PR#s for 80 colums (s is the slot of the Apple Tell board, the 80 columns card must be in slot 3). The screen then shows a parameter summary, and you are prompted for parameters. Type in the characters relatives to the options you choose (you may ommit a parameter, it will default to the first option shown for it) and end with a <RETURN>.
If you choosed an originate mode, you are then prompted for the number to dial. To just pick up the line (after a manual dialing) just press <RETURN>. You can incorporate pauses during the dialing by putting minus signs in the number.
Press <RETURN> to start dialing. The dialing can be interrupted with <ESC>.
If you choose an answer mode, the modem will wait for a ring. It will then pick-up (after 2 rings) and send the answer sequence. You can force pickup and skip the anwer sequence by pressing <ESC>.
The modem will then wait for a stable carrier (while sending it's own one in answer modes). When both modems send their carrier, the communication can begin.
Any character typed from the keyboard will be transmitted. Any character received will be passed as-is to COUT, exept LF wich is ignored and BELL that sounds the speaker directly. When using an Apple IIe with 80 colums at 1200 bps (videotex mode), some characters may be lost durring scrooling if the source does not provide a delay after CR. This does not occur with the Videx/Videoterm (Registered Trade Marks) card for it is much faster.
Press <RESET> to end communication (the modem hangs up by itself if the carrier get lost)

- CLEANUP

This little routine clears the receive FIFO and receive errors. Much usefull after connection or spike over the line.

Input: Y = $0E
Output: none

- BIGBELL

This routine is there to facilitate ring detection, make it more reliable, and insure that line will be picked up at a time when there is no high voltages over the phone line -beetween two rings-. This avoids unnecessary stresses to the phone line interface (the ring voltages won't damage it, but make it inoperative for a bout half a second).

First, the routine hangs up the line. Then it enters a loop where it monitors the Ring Detect bit, and the Keyboard. The routine will terminate in two cases:

(1) the ESC key is pressed (more precisely, if the KBD location reads $9B).

(2) the ring has rung two times, and we are in the middle of a silence period. The main program should then pick-up the line immediately.

Input: Y = $0F
A = $00 for compatibility with future versions, wich may require parameters for customizing the detection scheme (presently tested only in France).

Output: A = $00 if the routine terminated in case (1)
A <> $00 if the routine terminated in case (2)