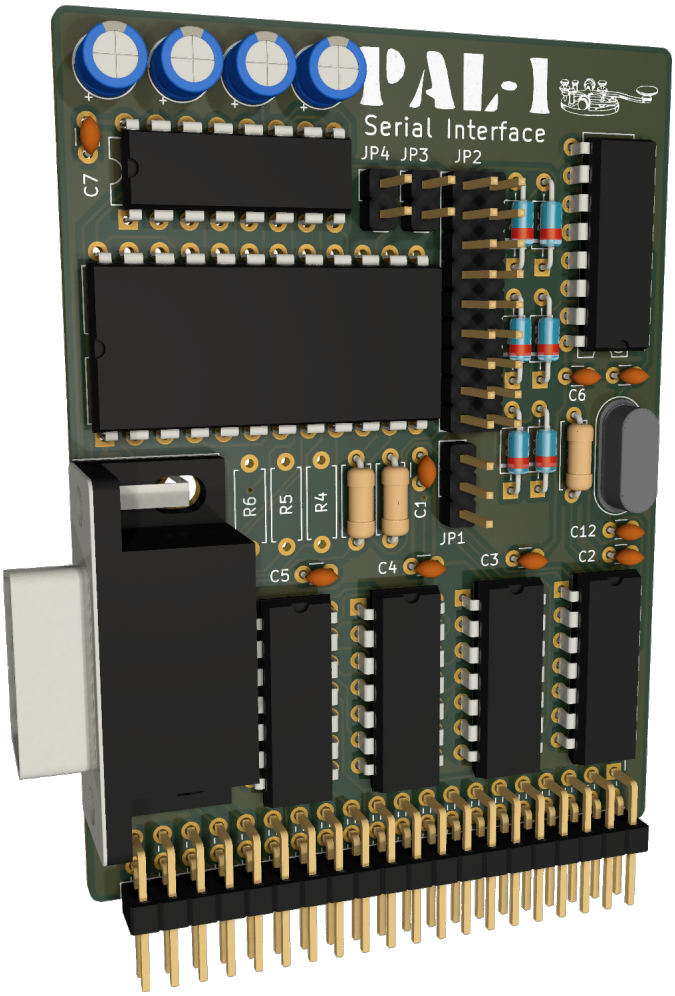


PAL-1 Serial Interface Adapter

Plastic Objects Limited

Published by **Plastic Objects Limited**
Woodbridge, UK

October 2022

Disclaimer

Contents

List of Figures

Introduction

The PAL-1 Serial Interface Adapter is an expansion module for the PAL-1 system[1] based on the Motorola MC6850 Asynchronous Communications Interface Adapter (ACIA)[2]. The design borrows from a homebrew project in CPC Schneider International magazine[3].

The MC6850 ACIA manages data formatting and asynchronous data communications control. Data from the PAL-1 is serially transmitted and received by the ACIA's asynchronous data interface, with appropriate formatting and error checking. The functional configuration of the ACIA may be programmed during system initialisation. It includes variable word lengths, clock division ratios, and interrupt conditions giving full control over serial communications[4].

Unlike the on-board PAL-1 serial interface, the PAL-1 Serial Interface Adapter provides hardware flow control at rates from 150 to 38400 baud. Serial connections may be made at either TTL or RS-232 levels, allowing users to connect to other devices with either a TTL-to-USB adaptor or using a 9 pin D-sub serial cable.

[1] Liu Ganning, 'PAL-1 Microcomputer User Manual' (November 2020), ⟨`http://pal.aibs.ws/assets/PAL_en.pdf`⟩ accessed 2022-07-21.

[2] Motorola Semiconductors, 'MC6850, Asynchronous Communications Interface Adapter (ACIA)' (1994).

[3] Joachim Schweda, 'Schnittstelle RS-232 im Selbstbau', *CPC Schneider International*, 7:3 (1986), pp. 88–92.

[4] Motorola Semiconductor Products, 'Motorola Microcomputer Components' (1978), p. 11.

1. *Getting Started*

Board assembly

Before assembling your PAL-1 Serial Interface Adapter board check the package contents against the Bill of Materials on page 25, and contact your distributor as soon as possible if any items are missing.

No specialist tools are required for assembling the board, though care must be taken when handling ESD sensitive components, especially the ICs. Before inserting the ICs into their sockets, check the board for dry joints and solder bridges. Also be sure to pay special attention to the orientation of the ICs, ensuring pin 1 of each IC is correctly aligned.

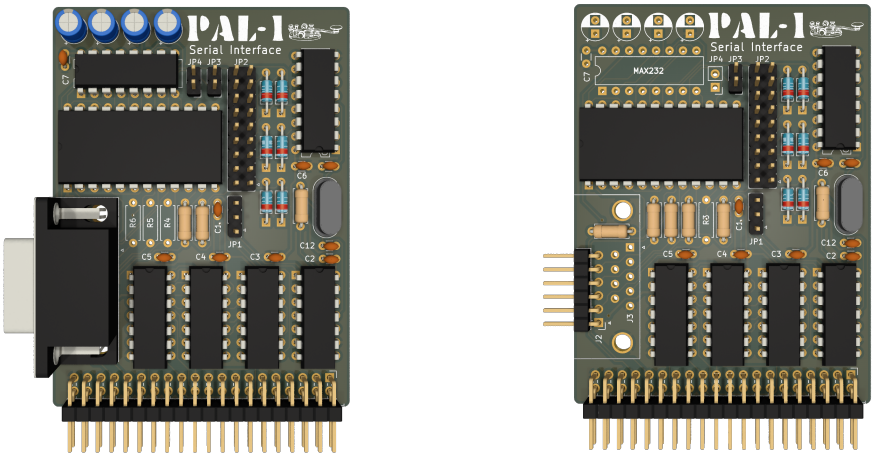
To prevent damage to your system, *always* power off your PAL-1 before installing or removing the PAL-1 Serial Interface Adapter board. Ensure that the pins are correctly aligned when inserting the board into the PAL-1 motherboard or when directly connecting it to the PAL-1 expansion port using a 40-pin IDC cable.

Builds options

The PAL-1 Serial Interface Adapter can be assembled with or without a MAX232 dual transmitter / dual receiver, depending on whether RS-232 or TTL level signals are preferred (see also Figure 1.1). If only TTL level signals are required, the components listed in Table 1.1 can be omitted. In this case pin-header J3 and resistors R7 to R10 should be installed to provide access to the TTL level signals (see also '*TTL Pin Header*' on page 20).

Parts	Description
C9 - C12	1.0uF capacitors
J2	DB-9 serial connector
JP5	Jumper, 2-pole, open
U6	MAX232 dual transmitter / dual receiver

Table 1.1: Components to be omitted for TTL level signals



(a) With RS-232 (b) TTL only

Figure 1.1: PAL-1 Serial Interface Adapter build options.

Configuration

Base I/O address

The MC6850 ACIA has an 8-bit data bus, which is memory mapped on the PAL-1 system. The base address at which it is mapped can be configured using jumper JP1, as shown in Table 1.2.

The default base address (and the address used in all examples in this manual) is $16E8. In situations where multiple serial interface adapters are installed in the same PAL-1 system, each board must have a unique base address.

JP1 pins	Address
1-2	$16E8 ($5864_{10}$)
2-3	$16EA ($5866_{10}$)

Table 1.2: I/O address configuration

Clock frequency

The MC6850 ACIA uses an external clock from which data transmission rates are derived. The frequency of this clock is configured using jumpers `JP2` as shown in table 1.3. For example, to select a clock frequency of 19.2kHz, shunt pins 2, 4, 5, and 8 as shown in figure 1.2. **Note that only the combinations shown in this table are valid — other settings may result in unpredictable behaviour.**

	JP2 pins									Baud rates	
Clock	1	2	3	4	5	6	7	8	9	Min	Max
9600 Hz	-	-	⊓	-	⊓	⊓	-	-	⊓	150	9600
19.2 kHz	-	⊓	-	⊓	⊓	-	-	⊓	-	300	19,200
38.4 kHz	⊓	-	⊓	⊓	-	-	⊓	-	-	600	38,400

Table 1.3: Clock frequency configuration

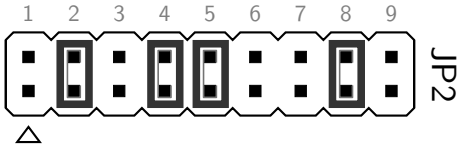


Figure 1.2: Clock set to 19.2 kHz.

Interrupts

Interrupts result from conditions in both the transmitter and receiver sections of the MC6850 ACIA. To enable interrupt handling on the PAL-1, pins JP3 must be shunted. For details about ACIA interrupt processing, refer to the section on interrupts on page 7.

Data Terminal Ready

The MC6850 ACIA does not provide a Data Terminal Ready (DTR) signal. While normally disconnected, the DTR signal can be set to active using JP4 if required by the data communications equipment (DCE). Note that the DTR signal is only available on the RS-232 connector (see also page 19).

2. *Using the Serial Interface Adapter*

The MC6850 ACIA appears on the PAL-1 as two addressable memory locations (see table 2.1). Internally, the ACIA has four registers. Of these, two are read-only, the Status and Receive Data registers. The remaining two registers are write-only, the Control and Transmit Data registers.

Address[1]	Access	ACIA Register
\$16E8 (5864_{10})	Read	Status Register
	Write	Control Register
\$16E9 (5865_{10})	Read	Receive Data
	Write	Transmit Data

[1] The PAL-1 Serial Interface Adapter base address is configured using jumper `JP1`. See page 2 for details.

Table 2.1: MC6850 registers

Much of the information in this chapter is taken from the MC6850 datasheet[1]. Further details on the ACIA may be found in the MC6850 application note (AN-754)[2].

Master Reset

The *master reset bits* (`CR0` and `CR1`) must be set immediately after power up to ensure the reset condition and to prepare for programming the ACIA functional configuration. After a master reset, the Control Register can be set to configure options, including clock divider ratios, word lengths, the number of stop and parity bits.

[1] Motorola Semiconductors.

[2] Karl Fronheiser, 'Device Operation and System Implementation of the Asynchronous Communications Interface Adapter (MC6850), AN-754', in: *MCU/MPU Applications Manual* (Motorola Inc, 1982), pp. 21–32.

ACIA Control Register

The MC6850 ACIA Control Register is used to configure serial communication parameters, including baud rates, word lengths, and transmission control parameters. The format of the 8-bit write only register is shown in figure 2.1.

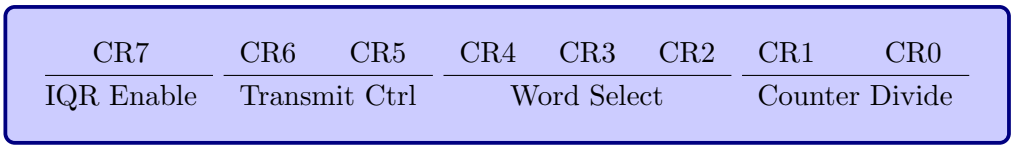


Figure 2.1: ACIA Control Register

Counter Divide and Reset

The *counter divide bits* (`CR0` and `CR1`) determine the divide ratios used in both the transmitter and receiver sections of the ACIA. Additionally, these bits are used to force a master reset of the ACIA, which clears the Status Register and initialises both the receiver and transmitter. Note that after power-on or a restart these bits must be set high to reset the ACIA. After resetting the clock divide ratio may be selected. The counter select bits provide for the following clock divide ratios and corresponding baud rates:

			Baud rate[1]		
CR1	CR0	Function	9600Hz	19.2kHz	38.4kHz
0	0	÷1	9600	19200	38400
0	1	÷16	600	1200	2400
1	0	÷64	150	300	600
1	1	master reset			

[1] Baud rates are shown for all clock frequencies.

Table 2.2: Counter Divide / Reset bits

Word Select

The *word select bits* (`CR4`,`CR3` and `CR2`) are used to select word length, parity, and the number of stop bits as shown in table 2.3. Note that word length, parity select, and stop bit changes are not buffered and therefore become effective immediately.

CR4	CR3	CR2	Word Length	Parity	Stop bits
0	0	0	7	even	2
0	0	1	7	odd	2
0	1	0	7	even	1
0	1	1	7	odd	1
1	0	0	8	none	2
1	0	1	8	none	1
1	1	0	8	even	1
1	1	1	8	odd	1

Table 2.3: Word Select bits

Interrupts

The *transmitter control bits* (`CR5` and `CR6`) and the *receive interrupt enable bit* (`CR7`) define the circumstances under which interrupts are raised. Note that in addition to setting the relevant control register bits, pin `JP3` on the PAL-1 Serial Interface Adapter board must be shunted (see also page 3).

Transmitter Control

The *transmitter control bits* (`CR5` and `CR6`) are used to configure the interrupt from the Transmit Data Register Empty condition, the Request to Send ($\overline{RTS}$) output, and the transmission of a Break level (space), as shown in table 2.4.

CR5	CR6	$\overline{RTS}$	Transmitter IRQ
0	0	low	disabled
0	1	low	enabled
1	0	high	disabled
1	1	low	disabled and break

Table 2.4: Transmitter Control bits

Receiver Interrupt Enable

Setting the *receiver interrupt enable bit* (`CR7`) will enable interrupts for the following conditions: Receive Data Register fill, Overrun, or a low-to-high transition on the Data Carrier Detect ($\overline{DCD}$) signal line.

ACIA Status Register

The status of the MC7850 ACIA may be read from the ACIA Status Register. The information stored in this register indicates the status of the Transmit Data Register, the Receive Data Register and error logic, and the peripheral status inputs of the ACIA.

SR7	SR6	SR5	SR4	SR3	SR2	SR1	SR0
$\overline{IRQ}$	PE	OVRN	FE	$\overline{CTS}$	$\overline{DCD}$	TDRE	RDRF

Figure 2.2: ACIA Status Register

.

Receive Data Register Full (RDRF)

The *receive data register full bit* (`SR0`) indicates that received data has been transferred to the receive data register. The *receive data register full bit* is cleared when the Receive Data Register is read or by a master reset. When the *receive data register full bit* is not set the data in the Receive Data Register is not current.

Transmit Data Register Empty (TDRE)

The *transmit data register empty bit* (`SR1`) is set when the contents of the Transmit Data Register have been transferred, indicating that new data may be entered. When the *transmit data register empty* is not set, the transmission of the character currently in the Transmit Data Register has not yet begun.

Data Carrier Detect ($\overline{DCD}$)

The *data carrier detect bit* (`SR2`) would normally indicate the modem carrier status. However, as the $\overline{DCD}$ pin of the MC6850 is permanently connected to ground on the PAL-1 Serial Interface Adapter, the *data carrier detect bit* will never be set.

Clear-to-Send ($\overline{CTS}$)

The *clear to send bit* (`SR3`) indicates the Clear-to-Send input from the modem. A low $\overline{CTS}$ indicates that there is a Clear-to-Send from the modem. In the high state, the Transmit Data Register Empty bit is inhibited and

the Clear-to-Send status bit will be high. Master reset does not affect the Clear-to-Send status bit.

Framing Error (FE)

The *framing error bit* (`SR4`) indicates that the received character is improperly framed by a start and stop bit. Framing errors are detected by the absence of the first stop bit. This error indicates a synchronisation error, a faulty transmission, or a break condition. The framing error bit is set or reset during the receive data transfer time. Therefore, this error indicator is present throughout the time that the associated character is available.

Receive Overrun (OVRN)

A receive overrun error indicates that one or more characters in the data stream were lost. This happens when a character or number of characters were received, but not read from the Receive Data Register prior to subsequent characters being received. The *receive overrun bit* (`SR5`) is reset after reading data from the Receive Data Register, or by a master reset.

Parity Error (PE)

The *parity error bit* (`SR6`) is set when the number of highs (ones) in the character does not agree with the preset odd or even parity. If no parity is selected then both the transmitter parity generator output and receiver parity check results are inhibited.

Interrupt Request ($\overline{IQR}$)

The *interrupt request bit* (`SR7`) indicates that an interrupt has occurred. This status bit may be set by any enabled interrupt condition. It is cleared when the Receive Data Register is read, or when a character is written to the Transmit Data Register.

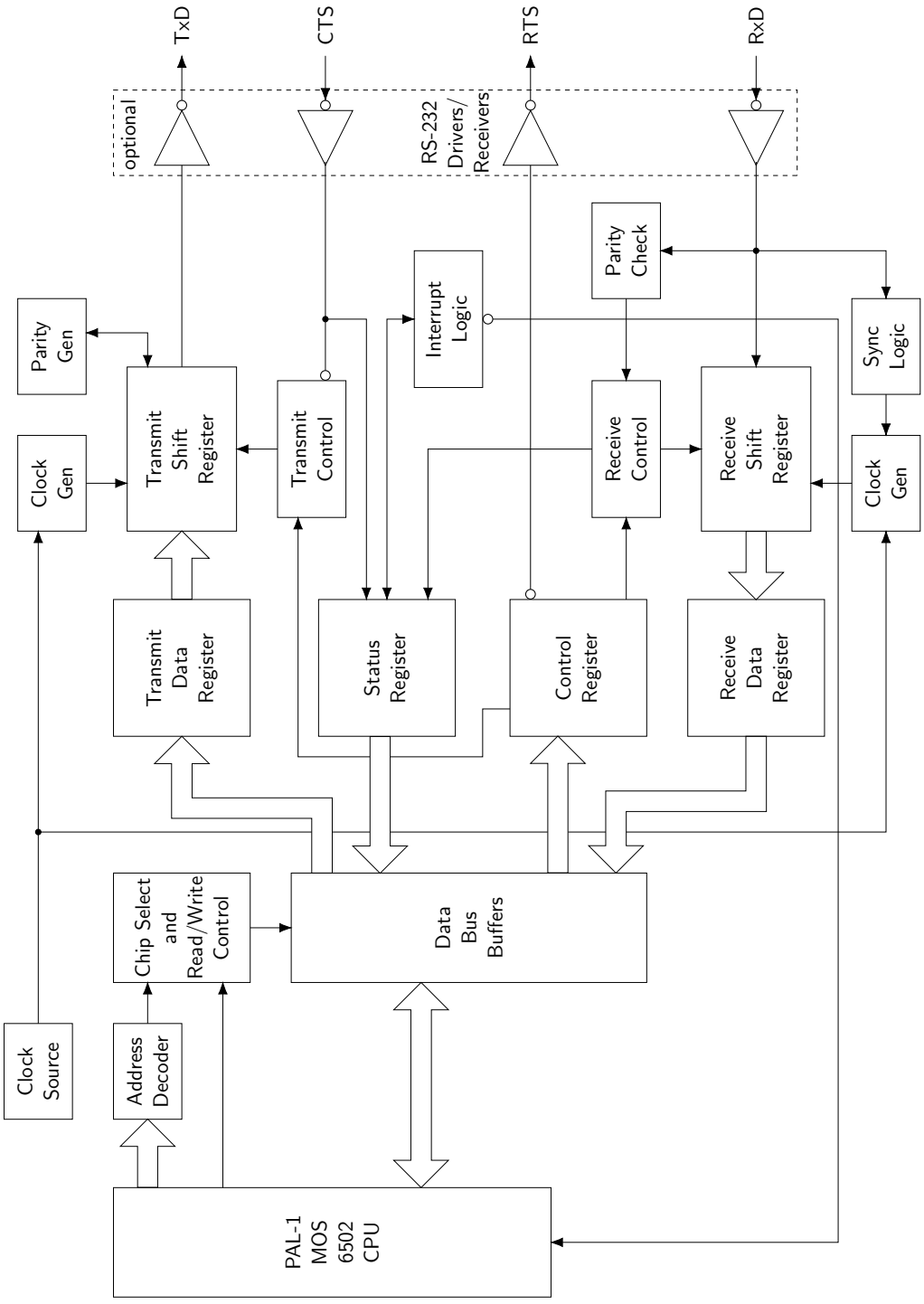


Figure 2.3: PAL-1 Serial Interface Adapter block diagram.

Hardware flow control

Devices process data at different rates. When transferring data from a faster device to a slower one, data may be lost unless the slower device is able to control the flow of data. Such flow control may be implemented using different methods. When using *software flow control*, special characters are sent via the data lines to pause and resume the flow of data. Alternatively, *hardware flow control* signals the pausing and resumption of data flows using additional wire connections between the communicating devices. Hardware flow control may be implemented in two different ways, as discussed below.

Hardware flow control

To implement hardware flow control, RTS (*Request to Send*) and CTS (*Clear to Send*) need to be cross-coupled (as shown in figure 2.4). Each device uses RTS to indicate that it is ready to accept data. Before sending any data a device needs to check CTS to ensure that the remote device is in a state to receive data.

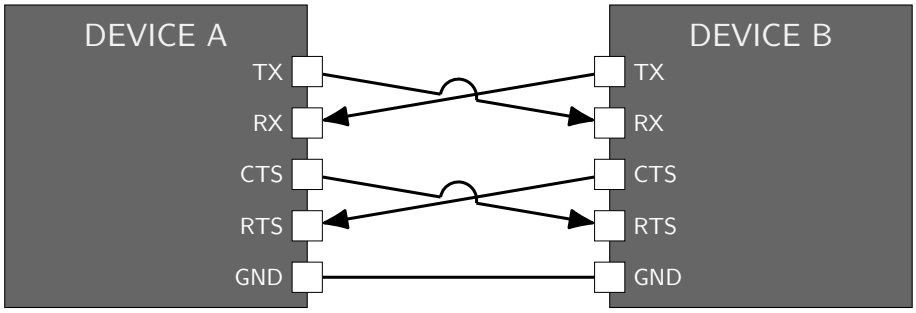


Figure 2.4: Hardware flow control.

A device will keep its RTS line asserted while it is ready to accept data. To ensure no data is lost, a device needs to negate RTS well before its receive buffer is full. No data should be sent to a device until RTS is asserted.

This implementation allows for bi-directional flow control; either device may request that data transmission is halted or resumed.

Legacy hardware flow control

In legacy systems hardware flow control is uni-directional, with the *Data Terminal Equipment* (DTE) managing the rate of flow to and from the *Data Communication Equipment* (DCE). In such systems the RTS (*Request to Send*) and CTS (*Clear to Send*) connections are wired straight through, rather than being cross-coupled (see figure 2.5).

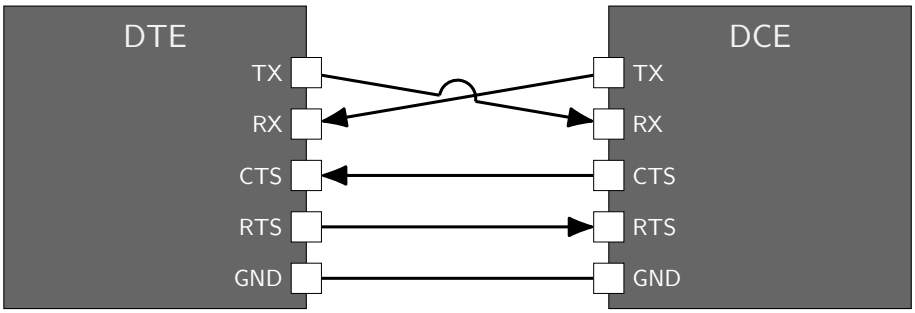


Figure 2.5: Legacy hardware flow control.

When it wants to transmit data, the DTE asserts RTS. If the DCE is in a state to accept data, it responds by asserting CTS. Transmission continues until the DCE negates CTS. When the DTE has completed transmitting data it negates RTS.

3. *Sample Code*

Programming the PAL-1 Serial Interface Adapter simply requires the ability to write to and read from the mapped I/O address. Many programming languages provide statements to achieve this, including the `POKE` and `PEEK` statements in BASIC, and `C!` and `C@` in Forth.

Note that all examples included below use the default base address (ie. \$16E8, 5864_{10}). Further examples may be found on Github at `https://github.com/dimitrit/pal1serial`.

BASIC

The following example was taken from the '*Do-It-Yourself RS-232 Interface*' article in the 1986 Special Edition, Issue 3 of CPC Schneider International magazine[1]. Because the MC6850 ACIA is memory mapped on the PAL-1 all `OUT` statements have been replaced with `POKE` statements, and `IN` with `PEEK`. Note that this example requires loopback connections from `TXD` to `RXD`, and from `RTS` to `CTS`.

```
10 REM ******************
20 REM *   RS-232-TEST   *
30 REM ******************
40 BA=5864:REM PAL-1 SERIAL I/F ADAPTER BASE ADDRESS
50 PRINT "␣␣␣␣␣␣␣␣␣␣␣␣RS-232"
60 PRINT "------------------------------"
70 PRINT:PRINT:PRINT
80 PRINT "BAUDRATE:␣A)␣␣300␣Bd␣[␣600␣Bd]"
90 PRINT "␣␣␣␣␣␣␣␣␣␣␣B)␣1200␣Bd␣[2400␣Bd]"
100 PRINT:PRINT:INPUT "SELECT␣BAUDRATE␣";B$
110 IF B$="A" THEN BD=2:GOTO 130
120 IF B$="B" THEN BD=1 ELSE GOTO 100
130 PRINT:PRINT:PRINT"ENSURE␣THE␣FOLLOWING␣PINS␣ARE␣CONNECTED:"
140 PRINT"␣␣␣␣TXD␣<--->␣RXD"
150 PRINT"␣␣␣␣RTS␣<--->␣CTS"
160 PRINT:PRINT
170 INPUT"␣<ENTER>␣";E$
180 PRINT:PRINT
190 POKE BA,3:REM MASTER RESET
```

[1] Schweda, pp. 88–92.

```
200 POKE BA,16+BD:REM SET WORD LENGTH & BAUD RATE
210 PRINT "␣␣␣␣***␣RS-232␣TEST␣***"
220 IF PEEK(BA) AND 8 THEN PRINT"CTS␣AND␣RTS␣NOT␣SHORTED":END
230 FOR A=32 TO 127
240 IF PEEK(BA) AND 2 THEN POKE BA+1,A ELSE 240
250 IF PEEK(BA) AND 1 THEN GOTO 260 ELSE 250
260 B=PEEK(BA+1)
270 PRINT "␣";CHR(A);"-";CHR(B);"␣";
280 IF A<>B THEN PRINT"FAILURE:␣";PEEK(BA):END
290 NEXT
300 END
```

Listing 3.1: RS-232 test program

6502 assembly language

The assembly language example shown below implements a simple terminal application. Characters read from the TTY device connected to the PAL-1 are sent to the Serial Adapter, and vice versa. As this example uses interrupts to process received characters, JP3 on the Serial Adapter Interface must be closed (see also page 3).

```
ACIA    = $16E8         ; MC6850 BASE I/O ADDRESS
ACIAC   = ACIA          ; ACIA CONTROL REGISTER
ACIAS   = ACIA          ; ACIA STATUS REGISTER
ACIAD   = ACIA+1        ; ACIA DATA REGISTER

SAD     = $1740         ; A DATA REGISTER
PADD    = $1741         ; A DATA DIRECTION REGISTER
SBD     = $1742         ; B DATA REGISTER
PBDD    = $1743         ; B DATA DIRECTION REGISTER

GETCH   = $1E5A         ; PUT CHAR FROM TTY IN A
OUTCH   = $1EA0         ; PRINT ASCII CHAR IN A ON TTY
SCANS   = $1F1F         ; DISPLAY $F9, $FA, AND $FB

; ACIA CONTROL REGISTER DEFINITIONS
;
; COUNTER DIVIDE AND RESET
RESET   = %00000011     ; RESET
DIV1    = 0             ; DIVIDE BY 1
;
; WORD SELECT
D8NP1S  = %00010100     ; 8 BITS, NO PARITY, 1 STOP BIT
;
; TRANSMITTER CONTROL
RTSLOW  = 0             ; /RTS LOW, XMIT INTERRUPT DISABLED
RTSHIGH = %01000000     ; /RTS HIGH, XMIT INTERRUPT DISABLED
;
```

```
; RECEIVE INTERRUPT ENABLE
RCVIRQ  = %010000000    ; ENABLE RECEIVE INTERRUPT

; ACIA STATUS REGISTER DEFINITIONS
;
RDRF    = %00000001     ; RECEIVE DATA REGISTER FULL
;
; ZERO PAGE VARIABLES
ACTRL   = $60           ; COPY OF CONTROL REG
WRITEPTR = $61          ; WRITE POINTER
READPTR = $62           ; READ POINTER

        *=$200

        CLD             ; BINARY MODE
        LDA #0          ; CLEAR POINTERS
        STA WRITEPTR
        STA READPTR

        LDA #$7F        ; SET DIRECTIONAL REGS
        STA PADD
        LDA #$3F
        STA PBDD

        LDA #RESET      ; RESET MC6850 ACIA
        STA ACIAC

;       LDA #(RCVIRQ|RTSHIGH|D8NP1S|DIV1)
        LDA #(RCVIRQ|D8NP1S|DIV1)
        STA ACTRL       ; REMEMBER INITIAL CONFIG
        STA ACIAC       ; AND CONFIGURE ACIA
        JMP ENABLE
CHECKPTRS
        SEI             ; DISABLE INTERRUPTS
        LDX READPTR     ; GET READ POINTER
        CPX WRITEPTR    ; AND COMPARE TO WRITE PTR
        CLI             ; ENABLE INTERRUPTS
        BEQ ENABLE      ;
PRINT
        LDA RECVBUF,X   ; GET NEXT CHAR IN RECVBUF
        JSR OUTCH       ; WRITE CHAR TO TTY
        INC READPTR
        JMP DONE
ENABLE
        LDA ACTRL       ; GET CURRENT STATUS
        AND #($FF~RTSHIGH)      ; CLEAR /RTS
        STA ACTRL       ; SAVE LOCALLY AND
        STA ACIAC       ; UPDATE ACIA CONTROL REG
DONE
        LDA ACTRL
        ROL             ; ROTATE STATUS TWICE TO
        ROL             ; GET RTS BIT INTO CARRY
        LDA #$D0        ; LOWER CASE 'R'
        BCC RTSSTATUS   ; /RTS LOW
```

```
        ORA #1          ; /RTS HIGH, SET TOP BAR
RTSSTATUS
        STA DISPBUF+6   ; SAVE RTS STATUS


                ; LDA ACIAS     ; GET ACIA STATUS
                ; ROR           ; ROTATE STATUS
                ; ROR
                ; ROR
                ; ROR
                ; BCC           ;

                ; PLA           ; RETRIEVE STATUS
                ; ROR
                ; PHA
                ; BCC           ;

                ; PLA

        LDX #5
        LDY #$13
CHARLOOP
        LDA #0
        STA SAD         ; NO FLICKER
        STY SBD
        LDA DISPBUF,X
        STA SAD
        TXA
        LDX #4
CHARDELAY
        DEX
        BNE CHARDELAY  ; LIGHT UP CHARACTER
        TAX
        DEY
        DEY
        DEX
        BNE CHARLOOP

        JMP CHECKPTRS

; SENDCHAR      PHA             ; SAVE CHARACTER
; WAITSND               LDA ACIA        ; GET ACIA STATUS
;               AND #%010
; IS TRANSMIT RECVBUF EMPTY?
;               BEQ WAITSND     ; NO, WAIT LONGER
;               PLA
;               STA ACIA+1      ; WRITE CHARACTER TO RECVBUF
;               RTS
; AND DONE

INITSVC
        PHA             ; PRESERVE A
        TXA             ; PRESERVE X
        PHA
```

```
        LDA WRITEPTR    ; GET WRITE POINTER
        CMP READPTR     ; COMPARE TO READ POINTER
        BCS COPYCHAR    ; WRITE POINTER AHEAD OF READ
        CLC             ; DON'T LET WRITE GET WITHIN
        ADC #$18        ;   24 BYTES OF READ POINTER
        CMP READPTR     ;
        BCC COPYCHAR    ; ALL GOOD
        LDA ACTRL       ; GET CURRENT STATUS
        ORA #RTSHIGH    ; SET /RTS
        STA ACIAC       ; UPDATE ACIA CONTROL REG
        STA ACTRL       ; AND SAVE LOCALLY
COPYCHAR
        ;LDA ACIAS      ; GET ACIA STATUS
        ;AND #RDRF      ; CHAR IN RECEIVE RECVBUF?
        BEQ SVCDONE     ; NO, ALL DONE
        LDX WRITEPTR    ; GET WRITE POINTER
        LDA ACIAD       ; GET CHARACTER FROM RECV BUF
        STA RECVBUF,X   ; ADD STORE IN LOCAL RECVBUF
        INC WRITEPTR    ; INCREASE POINTER
SVCDONE
        PLA             ; RESTORE Y
        TAX
        PLA             ; RESTORE A
        RTI             ; AND RETURN

        ; P = PARITY ERROR
        ; O = RECEIVER OVERRUN
        ; F = FRAMING ERROR
        ; C = CLEAR TO SEND
        ; R = REQUEST TO SEND
DISPBUF
        .BYTE  0,$F3,$DC,$F1,$D8,$D0

        .ALIGN 8        ; ALIGN TO NEXT PAGE BOUNDARY
RECVBUF
        .BYTE 0         ; RECEIVE BUFFER

        *=$17FE         ; ORG AT IRQ VECTOR
IRQT
        .WORD INITSVC
        .END
```

Listing 3.2: MiniTerm program listing

A. *Connector Pin Outs*

RS-232 D-Sub Connector

The pin-out of the RS-232 D-Sub 9-pin male connector (J2) follows the standard convention for Data Terminal Equipment (DTE), as shown in Figure A.1.

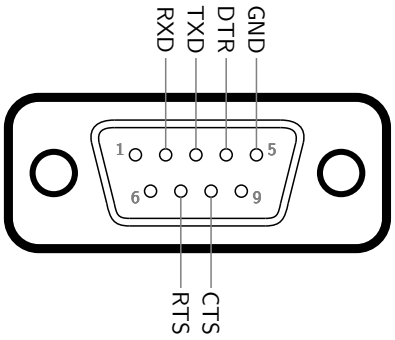


Figure A.1: D-Sub 9-Pin Male Connector

Pin	Signal	Description
1	NC	Not Connected
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready[1]
5	GND	Ground
6	NC	Not Connected
7	RTS	Request to Send
8	CTS#	Clear to Send
9	NC	Not Connected

[1] DTR is set using `JP5` (see page 3).

Table A.1: D-Sub 9-Pin Male Connector

TTL Pin Header

The PAL-1 Serial Adapter TTL pin header (J3) allows 6-way FTDI based USB to TTL converters to be directly connected[1]. **Be sure to check the USB to TTL converter voltage levels and signals before connecting as incorrect voltage and/or incompatible signals may result in damage to the PAL-1 Serial Adapter.**

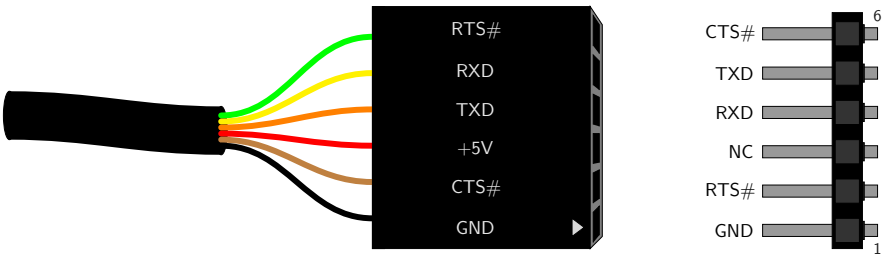


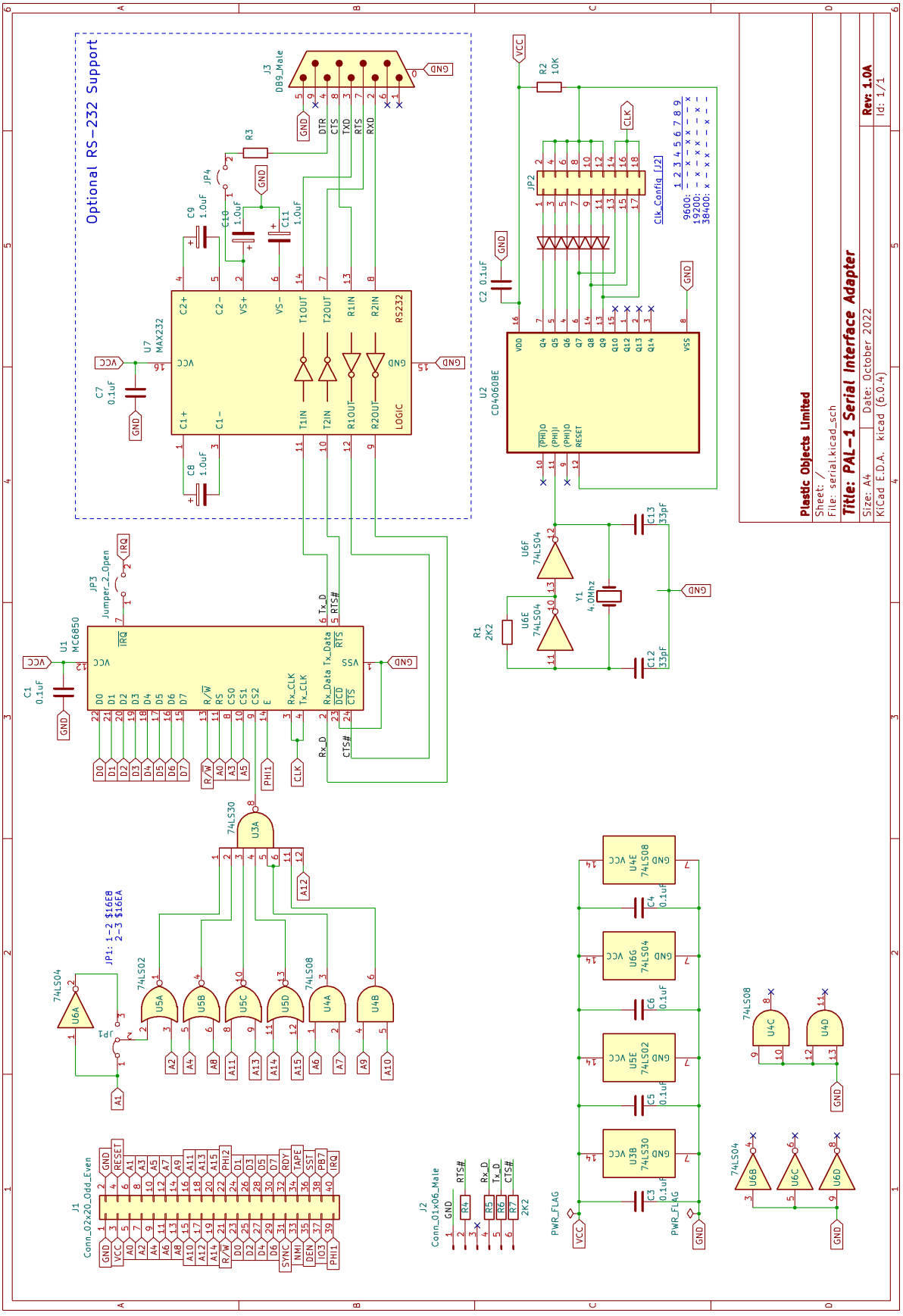
Figure A.2: TTL Pin Header

Pin	Signal	Description	TTL-232R-5V	Signal
1	GND	Ground	Black	GND
2	RTS#	Request to Send	Brown	CTS#
3	NC	Not Connected	Red	5V
4	RXD	Receive Data	Orange	TXD
5	TXD	Transmit Data	Yellow	RXD
6	CTS#	Clear to Send	Green	RTS#

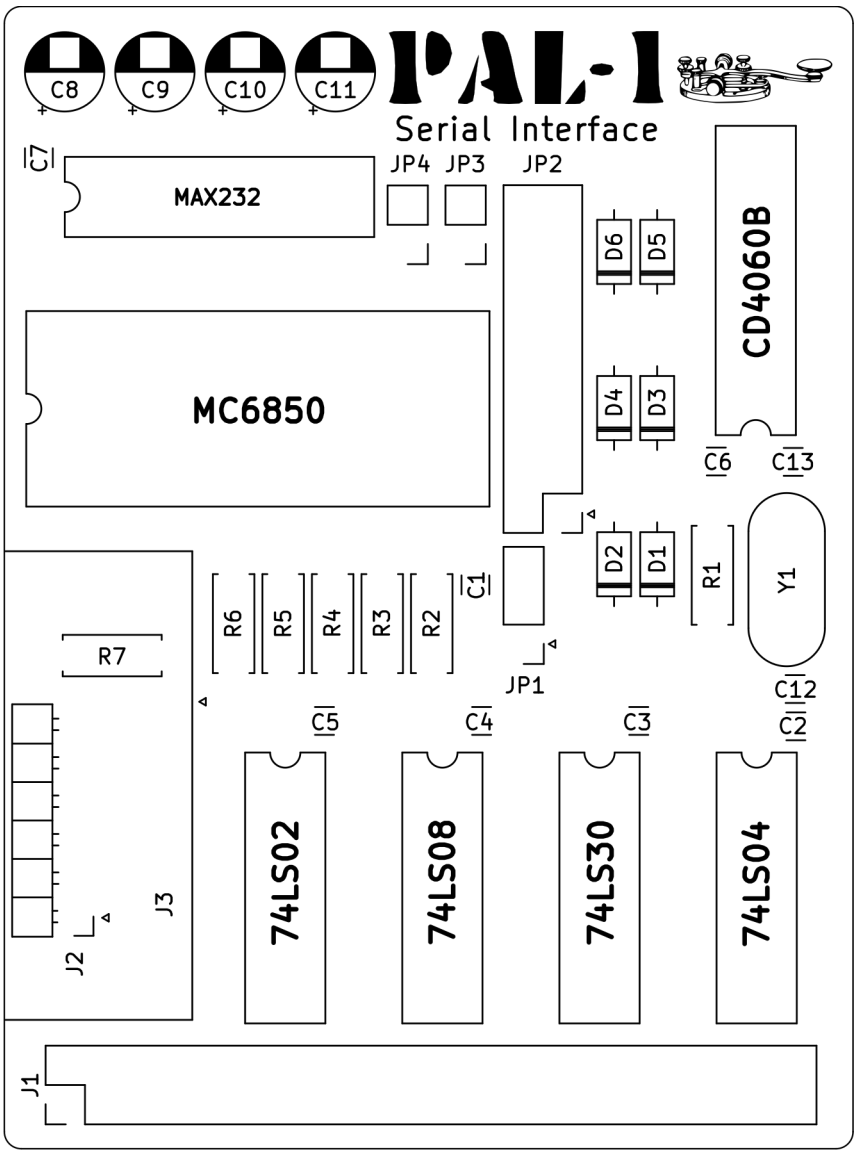
Table A.2: TTL Pin Header & FTDI USB Adapter Connections

[1] FTDI Chip, 'TTL-232R TTL to USB Serial Converter Range of Cables Datasheet' (2019), ⟨`https://ftdichip.com/wp-content/uploads/2021/02/DS_TTL-232R_CABLES.pdf`⟩ accessed 2022-10-27, pp. 11–12.

B. *Schematic Diagram*


Optional RS-232 Support
Plastic Objects Limited
Sheet: /
File: serial.kicad_sch
Title: PAL-1 Serial Interface Adapter
Size: A4
Date: October 2022
KiCad E.D.A. kicad (6.0.4)
Rev: 1.0A
Id: 1/1

C. *Board layout*



D. *Bill of Materials*

Part	Qty	Value	Description
C1 - C7	7	0.1uF	Unpolarized capacitor
C8 - C11	4	1.0uF	Polarized capacitor
C12, C13	2	33pF	Unpolarized capacitor
D1 - D6	6	1N4148	Standard switching diode, DO-35
J1	1		Connector, double row, 2x20
J2	1		Connector, single row, 1x6
J3	1		Connector, 9-pin male D-SUB
JP1	1		Jumper, 3-pole, pins 1+2 closed/bridged
JP2	1		Connector, double row, 2x9
JP3, JP4	2		Jumper, 2-pole, open
R1, R4-R7	5	2K2	Resistor
R2	1	10K	Resistor
R3	1	2K7	Resistor
U1	1	MC6850	Asynchronous Communications Interface Adapter 1MHz
U2	1	CD4060BE	14-stage binary counter/divider
U3	1	74LS30	8-input NAND gate
U4	1	74LS08	Quad 2-input AND gate
U5	1	74LS02	Quad 2-input NOR gate
U6	1	74LS04	Hex Inverter
U7	1	MAX232	Dual RS232 driver/receiver
Y1	1	4.0Mhz	Two pin crystal

Table D.1: PAL-1 Serial Interface Adapter v1.0A components

References

Fronheiser, Karl, 'Device Operation and System Implementation of the Asynchronous Communications Interface Adapter (MC6850), AN-754', in: *MCU/MPU Applications Manual* (Motorola Inc, 1982), 21–32.

FTDI Chip, 'TTL-232R TTL to USB Serial Converter Range of Cables Datasheet' (2019), ⟨`https://ftdichip.com/wp-content/uploads/2021/02/DS_TTL-232R_CABLES.pdf`⟩ accessed 2022-10-27.

Ganning, Liu, 'PAL-1 Microcomputer User Manual' (November 2020), ⟨`http://pal.aibs.ws/assets/PAL_en.pdf`⟩ accessed 2022-07-21.

Motorola Semiconductor Products, 'Motorola Microcomputer Components' (1978).

Motorola Semiconductors, 'MC6850, Asynchronous Communications Interface Adapter (ACIA)' (1994).

Schweda, Joachim, 'Schnittstelle RS-232 im Selbstbau', *CPC Schneider International*, 7:3 (1986), 88–92.