ADDI-DATA GmbH
Dieselstraße 3
D-77833 OTTERSWEIER

Technical support:
+49 (0)7223 / 9493 – 0

Technical description

ADDICOM PA 750

Serielle Schnittstelle

7th edition 12/1993

TABLE OF CONTENTS

1. FOR A QUICK AND SAFE INSTALLATION...

For a quick and safe installation of this board please read absolutely following chapters :

3. Board adjustment

CAUTION !

Befor installing the board **PA 750** check that the selected base address and the related address range are not already used by another board or by the computer itself.

In this case choose a free address area for board **PA 750**, otherwise it could cause damages to board and computer.(See chapter 3).

Be sure to turn power off before adjusting or removing the board.

2. Functions

Board **PA 750** is an interface with four asynchronous ports, which can be freely configurated with the software package delivered with the board.

You can define the following parameters individually for each port :

- Baud rate
- Parity
- Even, Odd
- Stop bits

The setting applies generally to RS 232, RS 422, RS 485 and current loop. The board does not affect the function of COM 1 and COM 2.

The ports are called COM 3 to COM 6.

3. Board adjustment

The board's different functions can be adjusted over wire wrap connections. Each COM interface can be set independently from the others to the adequate function. The interrupt line can be also selected via wire wrap connections.

3.1 Setting the base address

The board occupies 32 bytes within the 64KB I/O address space of the PC. The base address is set via two DIP switches. The following switches are related to the following addresses :

S1/8	A15
S1/7	A14
S1/6	A13
S1/5	A12
S1/4	A11
S1/3	A10
S1/2	A9
S1/1	A8
S2/4	A7
S2/3	A6
S2/2	A5
S2/-	

In terms of function, the switch designations S1/8 - S2/2 correspond to the address bits A15-A5 of the PC bus.

At delevery, the base address of the board is set on 300H.

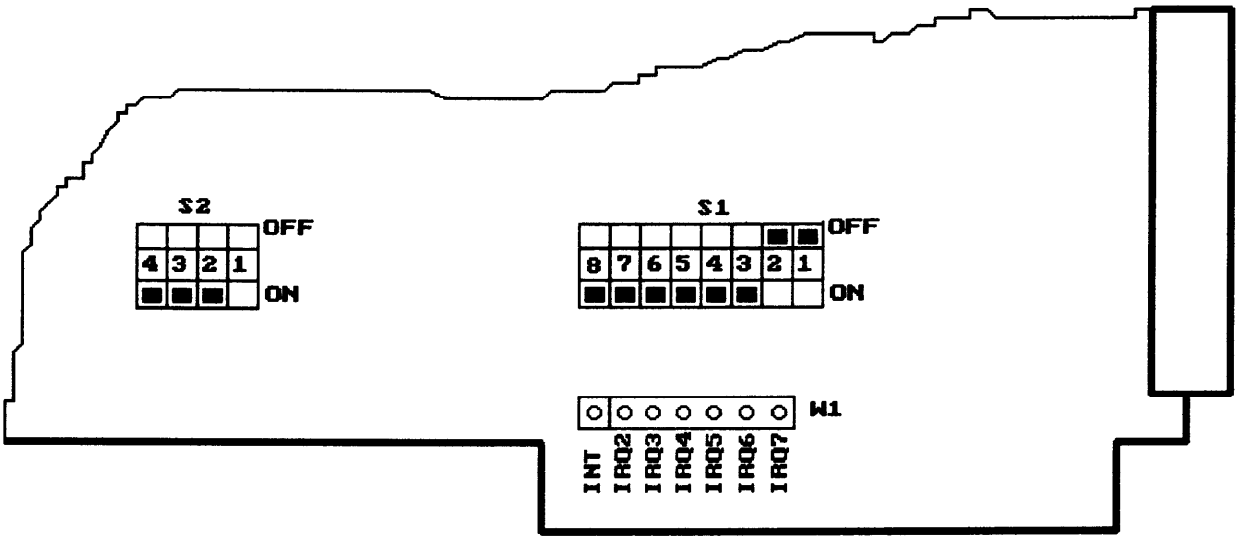
You will find in the next table the setting of the base address.

SETTING THE BASE ADDRESS

S2				S1								DIP switch
1	2	3	4	1	2	3	4	5	6	7	8	
	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	**Address range**
X	ON	ON	OFF	ON	OFF	ON	ON	ON	ON	ON	ON	0280H-029FH
X	ON	OFF	OFF	ON	OFF	ON	ON	ON	ON	ON	ON	02C0H-02DFH
X	ON	ON	ON	OFF	OFF	ON	ON	OFF	ON	ON	ON	1300H-131FH

On the address bus, a switch position "ON" corresponds to a logical "0" and a switch position "OFF" corresponds to a logical "1".

DIP switch on the board : (address range 0300H-031FH)



CAUTION :

When setting the base address, the address bits A3 and A4 must have the logic value "0".

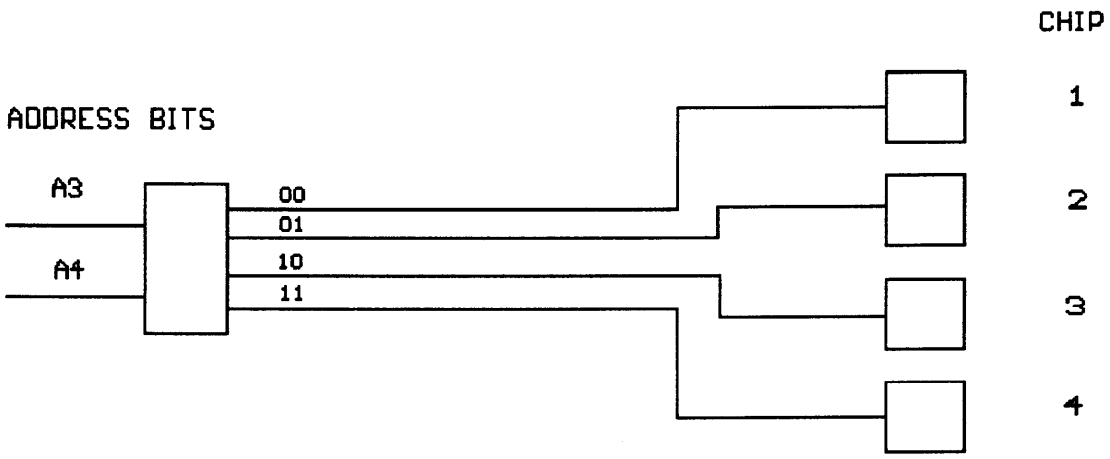
BASE ADDRESS SET ON 0300H

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	1	1	0	0	0	**0**	**0**	0	0	0
0				3				0				0			

BASE ADDRESS SET ON 0310H (FALSE SETTING)

A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
0	0	0	0	0	0	1	1	0	0	0	**1**	**0**	0	0	0
0				3				1				0			

SELECTION OF ONE CHIP VIA ADDRESS BITS A3 AND A4



TRUTH TABLE

Inputs A4 and A3	Selected output
00	chip 1
01	chip 2
10	chip 3
11	chip 4

For adjusting correctly the base address, the address bits A3 and A4 have to select the first chip.

If the base address was set on 0310H, the address bits A3 and A4 would not select the first but the third chip. The next address settings would be shifted.

The only purpose of the DIP switches is to set the base address of the board. The following table shows the logical functions of the different addresses.

CLASSIFICATION OF THE I/O ADDRESS RANGES

Address	I/O function		Description		Port
yyyy+0	IORD		Received data register		COM3
yyyy+0		IOWR	Transmitted data register	Baud rate	
yyyy+1	IORD	IOWR	Interrupt enable register	Baud rate	
yyyy+2	IORD		Interrupt identification register		
yyyy+3	IORD	IOWR	Line control register		
yyyy+4	IORD	IOWR	Modem control register		
yyyy+5	IORD		Line status register		
yyyy+6	IORD		Modem status register		
yyyy+8	IORD		Received data register		COM4
yyyy+8		IOWR	Transmitted data register	Baud rate	
yyyy+9	IORD	IOWR	Interrupt enable register	Baud rate	
yyyy+10	IORD		Interrupt identification register		
yyyy+11	IORD	IOWR	Line control register		
yyyy+12	IORD	IOWR	Modem control register		
yyyy+13	IORD		Line status register		
yyyy+14	IORD		Modem status register		

yyyy = Base address

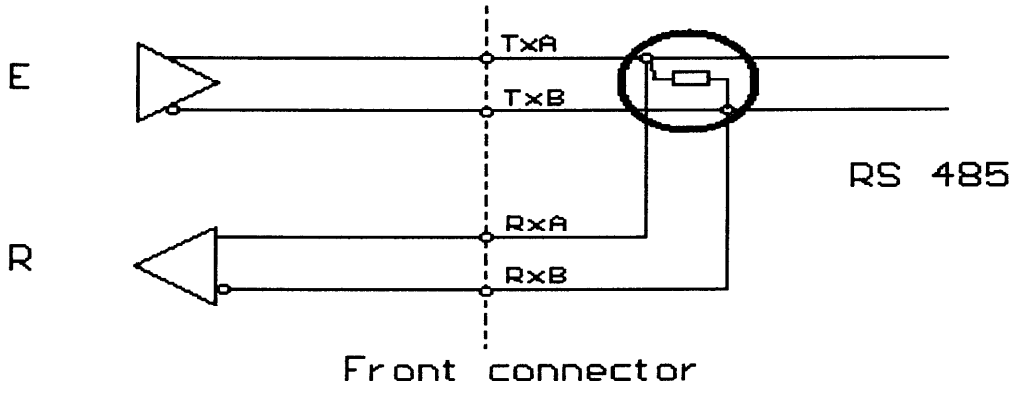
Address	I/O function		Description		Port
yyyy+16	IORD		Received data register		COM5
yyyy+16		IOWR	Transmitted data register	Baud rate	
yyyy+17	IORD	IOWR	Interrupt enable register	Baud rate	
yyyy+18	IORD		Interrupt identification register		
yyyy+19	IORD	IOWR	Line control register		
yyyy+20	IORD	IOWR	Modem control register		
yyyy+21	IORD		Line status register		
yyyy+22	IORD		Modem status register		
yyyy+24	IORD		Received data register		COM6
yyyy+24		IOWR	Transmitted data register	Baud rate	
yyyy+25	IORD	IOWR	Interrupt enable register	Baud rate	
yyyy+26	IORD		Interrupt identification register		
yyyy+27	IORD	IOWR	Line control register		
yyyy+28	IORD	IOWR	Modem control register		
yyyy+29	IORD		Line status register		
yyyy+30	IORD		Modem status register		

yyyy = Base address

3.2 Comments to RS 232 and RS 485

RS 232 is used with or without modem control signals (only TxD and RxD).

RS 485: you can add a 56 ohm resistor between A and B (receiver). According to the COM interface used, Jumper B2 to B9 must be open. The diagram below shows the necessary cabling.



Each COM interface can be set individually. See the block diagram in the appendix.

3.3 Setting interrupts

The board generates four internal interrupts, which are linked to one interrupt signal via a special decoder.

This signal is available on ww pin INT. The interrupt of a COM interface can be blocked by software.

When the interrupt release is blocked in the modem control register (OUT2), this blocking of the hardware interrupt can be carried out for every COM interface.

The common interrupt line can be switched to IRQ2,-3,-4,-5,-6 and IRQ7 interrupt line via ww connections.

4. Mechanical set-up

A 1,6 mm thick printed circuit is the mechanical and electrical connection (233 x 99 mm). The connection with the microcomputer bus occurs over a 62 pole gold plated direct plug. The four COM interfaces are connected over a 37 pole MIN-D pin connector. The board is plugged directly into the PC and is screwed onto the back pannel of the appliance with an hinge.

The board's functions are adjusted over ww connections. The base address is set with two DIP switches.

5. Driver

5.1 Introduction

DRV750 is a device driver for IBM PC/XT/AT and compatibles. With this driver it is possible to use under MS-DOS up to 4 serial ports of board **PA 750** designated XCOM1 to XCOM4. DRV750 is equipped with receiver buffers (512 characters each). Transfer rates up to 56 000 Bauds are therefore possible. This driver uses interrupts for transmitting or receiving data.

Data can be sent and received with any programming language (C, Pascal, Basic etc).

A communications protocol (RTS/CTS) is used and managed automatically by the driver and can be easily inhibitted.

At any time you can change the transfer rate and the communication parameters individually

for each serial port of board **PA 750**.

For example the function COPY of MS-DOS allows to send a file on one of the serial ports of the board.

5.2 Floppy disk

The floppy disk contains the following files:

CONFIG.SYS	Example for installing the driver in file CONFIG.SYS
DEMOx.C	Programming example
DEMOx.Pas	Programming example
XMODE.C	Source file
ACTYPE.H	Header file
XMODE.EXE	Initilization program equivalent to the command MODE of MS-DOS. It permits initializing each port of board **PA 750**.
DRV750.SYS	Driver program to install in CONFIG.SYS.
DRV750.EXE	Install program

To install the driver: A: INSTALL C:

5.3 Board PA 750

The serial ports COM1 and COM2 are the standards for PC/XT, COM1 to COM4 are the standards for PC/AT. If you want to use more than 4 ports, you need a board **PA 750** and a driver for having a more easy-to-handle interface between the user program and the board.

The driver is loaded in the memory when the PC is booted. It occupies about 11K bytes and remains resident. For loading the driver automatically while the PC is booted, you have to add a command line in the file CONFIG.SYS (see chapter 10.4).

5.4 File CONFIG.SYS

5.4.1 Configuring the ports in common interrupt

Symbol signification

	A =		Base Address
	C =		Common interrupt
Ex.:	A 300	=	Base Address at 300 in Hex
	C	=	Common interrupt at IRQ15

```
DEVICE = C:\PA750\DRV750.SYS    - A 300        - C3
             |     |              |              |
         driver    driver        base          common
       directory     name       address       interrupt
```

This configuration permits the access to 4 serial ports (XCOM1 to XCOM4) in common interrupt (IRQ3).

Can be used in common interrupt: IRQ2, IRQ3, IRQ4, IRQ5, IRQ6 and IRQ7.

5.4.2 Default setting for the serial ports

The following default setting is valid for the 4 ports.

Baud rate:	9600 Bds
Parity:	None
Data bit:	8
XON:	11Hex
XOFF:	13Hex
Protocol:	RTS/CTS

These parameters can be changed by software (see chapter 5.7.3).

5.5 Using DOS functions

Since the driver DRV750.SYS uses peripheral names, it can be directly called in the command line with MS-DOS functions.

Ex.:

COPY ABC.TXT XCOM1	Copies file ABC.TXT on XCOM1
COPY XCOM1 ABC.TXT	Reads from port XCOM1 and copies data in ABC.TXT. For this example, it is necessary that the code 1AH (end of file under DOS) is sent at the end ofreceiving. Otherwise MS-DOS would expect infinitely on this end of file code and would not give the hand.

5.6 Programming with DRV750.SYS

The access to a serial port is possible with any programming language and is carried out in the same way as for acceding to a disk file.

5.6.1 Pascal

```
PROGRAM   TEST

USES
    CRT, DOS;

VAR
   Input, Output : TEXT;          (* TEXT file  *)
   Send          : STRING [80];   (* Buffer containing the message to be sent     *)
   Receive       : STRING [80];   (* Buffer containing the message to be received *)

BEGIN
   ASSIGN (Input, 'XCOM1') ; RESET (Input);        (* Open the communication *)
   ASSIGN (Output, 'XCOM1') ; REWRITE (Output);    (* channel XCOM1          *)
   SEND    := "This is the message to send";
   RECEIVE := "                              ";

   WRITE (Output, Send);       (* Send data  *)
   READ  (Input, Receive);     (* Receive data *)

   CLOSE (Output);
   CLOSE (Input);
```

```
END.
```

You will find programming examples on the diskette. The communication type with the driver is the same as for using the file.

5.7 IOCTL functions

The functions IOCTL (I/O control) of MS-DOS allow to configure easily the driver. The following parameters can be changed: baud rate, parity, number of data and stop bits, and the communication parameter protocols RTS/CTS or XON/XOFF.

You can acceed to the IOCTL functions, either directly over a keyword of the programming language, if it is possible, or indirectly over a MS-DOS interrupt 21Hex by using the function 44Hex.

5.7.1 Example of a direct call in C

```
/*

+-------------------------------------------------------------------+
| (C) Addi-Data GmbH          Daimlerstr. 2          77815 Buhl     |
+-------------------------------------------------------------------+
| Tel : 07223/9493-20                                               |
| Fax : 07223//9493-92                                              |
+-------------------------------------------------------------------+
| Project     : -             | Compiler   : BORLAND C++            |
| Module name : DEMO3.C       | Version    : 2.00                   |
+-------------------------------------------------------------------+
| Descript. : Demonstration for using the driver with board PA750.  |
+-------------------------------------------------------------------+

*/

#include <dos.h>
#include <fcntl.h>
#include <stdio.h>
#include <process.h>
#include <string.h>
#include <io.h>
#include <conio.h>

#define SEND    27
#define IOCTL   50
#define RECEIVE 512


/*
```

```
Function name: main
*/

void main (void)
   {
   char Ioctl   [IOCTL];        // Configuration buffer.
   int  Serial2;
   int  Serial1;                // Handler number returned by 'open'


   Serial1 = open ("XCOM1", O_RDWR | O_BINARY); // Opening XCOM1 (PA750).
   Serial2 = open ("XCOM2", O_RDWR | O_BINARY); // Opening XCOM2 (PA750).
   if (!Serial1 || !Serial2)
      {
      printf ("XCOMx can not be opened !!!");
      exit (1);
      }

   ioctl (Serial1, 2, Ioctl, IOCTL);    // Reading configuration.
   printf ("\nConfiguration XCOM2: %s", Ioctl);
   }
   ioctl (Serial2, 2, Ioctl, IOCTL);    // Reading configuration.
   printf ("\nConfiguration XCOM1: %s", Ioctl);

   strcpy (Ioctl, "b1200;pN;s1;d8;o17;f19;c0\n");

   ioctl (Serial2, 3, Ioctl, IOCTL);    // Writing configuration.
   ioctl (Serial1, 3, Ioctl, IOCTL);    // Writing configuration.

   ioctl (Serial1, 2, Ioctl, IOCTL);    // Reading configuration.
   printf ("\nConfiguration XCOM2: %s", Ioctl);

   ioctl (Serial2, 2, Ioctl, IOCTL);    // Reading configuration.
   printf ("\nConfiguration XCOM1: %s", Ioctl);

   close (Serial1);
   close (Serial2);
   }
```

5.7.2 Example of an indirect call in Pascal

```
(*
   +-------------------------------------------------------------------+
   | (C) ADDI-DATA GmbH          Daimlerstr. 2          77815  Buhl     |
   +-------------------------------------------------------------------+
   | Tel : 07223/9493-20                                               |
   | Fax : 07223//9493-92                                              |
   +-------------------------------------------------------------------+
   | Project     : -             |  Compiler   : TP                    |
   | Module name : DEMO3.PAS     |  Version    : 5.00                  |
   +-------------------------------------------------------------------+
   | Descript. : Demonstration for using the driver with board PA750.  |
   +-------------------------------------------------------------------+
*)
PROGRAM DEMO3;

USES
    CRT, DOS;

TYPE
    iotyp = string[80];

VAR Input, Output : TEXT;
    IoInBuf       : iotyp;
    regs          : REGISTERS;
(*
   +-------------------------------------------------------------------+
   | Function name   : IoctlIn                                         |
   +-------------------------------------------------------------------+
   | Task            :                                                 |
   |         Reads the configuration of the communication port with a  |
   |         DOS function.                                             |
   +-------------------------------------------------------------------+
*)
PROCEDURE IoctlIn (var Handler : Text; var Buf : iotyp);
BEGIN
   regs.BX := MEM[SEG(Handler):OFS(Handler)]+256*MEM[SEG(Handler):OFS(Handler)+1];
   regs.AX := $4402;
   regs.CX := LENGTH (Buf);
   regs.DX := OFS (Buf)+1;
   regs.DS := SEG (Buf);
   msdos (regs);
END;
```

```
(*
 ┌──────────────────────────────────────────────────────────────┐
 │ Function name   : IoctlOut                                   │
 ├──────────────────────────────────────────────────────────────┤
 │ Task        :                                                │
 │   Initializing the configuration of the communications port. │
 └──────────────────────────────────────────────────────────────┘
*)
PROCEDURE IoctlOut (var Handler : Text; var Buf : iotyp);
BEGIN
   regs.BX := MEM[SEG(Handler):OFS(Handler)]+256*MEM[SEG(Handler):OFS(Handler)+1];
   regs.AX := $4403;
   regs.CX := LENGTH (Buf);
   regs.DX := OFS (Buf)+1;
   regs.DS := SEG (Buf);
   msdos (regs);
END;


VAR
   Send    : STRING[80];
   Receive : STRING[255];
   i       : BOOLEAN;

BEGIN
   ASSIGN (Input, 'XCOM1'); RESET (Input);
   ASSIGN (Output, 'XCOM1'); REWRITE (Output);

   Send    := 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
   IoInBuf := '                            ';

   IoctlIn (Input, IoInBuf);
   WRITELN ('a) Read configuration of XCOM1 : ', IoInBuf);

   IoInBuf := 'b1200;pN;s1;d8;o17;f19;c1' + chr(13) + chr(10);
   IoctlOut (Output, IoInBuf);

   IoctlIn (Input, IoInBuf);
   WRITELN ('b) Read configuration of XCOM1 : ', IoInBuf);

   CLOSE (Input); CLOSE (Output);

END.
```

5.7.3 Setting a new port configuration

The configuration line must end with CR,LF (0DHex, 0AHex)

a) Parametering the transfer rate

Keyword: bxxxx or Bxxxx

xxxx is the wished transfer rate

Possible rates: 50, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, 19 200, 38 400, 56 000 Bauds

If the value xxxx is not in the list below, a rate of 9600 Bauds is set automatically.

Example in C

```
strcpy (BufIO; "b9600\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
                |
        Setting configuration
```

b) Parametering the parity bit

Keyword: px or Px

x refers to the parity wished

N=None
O=Odd
E=Even

\n\r = n for line feed - r for carriage return

Example in C

```
strcpy (BufIO; "b9600; pN\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
                |
        Setting configuration
```

c) Parametering the number of data bits

Keyword: dx or Dx

x represents the number of bits which have to be contained in one data word (5, 6, 7, 8).

Example in C

```
strcpy (BufIO, "b9600; pN; d8\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
        Setting configuration
```

d) Parametering the number of stop bits

Keyword: Sx or sx

x : represents the number of stop bits to add to the word that you wish to send over the serial port.

1,2 are possible numbers

Example in C

```
strcpy (BufIO; "b9600; pN; s1; d8\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
        Setting configuration
```

e) Parametering the XON code (if protocol is enabled - see § g)

Keyword: Oxx or oxx

When the receiver buffer is half-empty (256 characters), the driver sends the XON code to the periphery which can carry on emission. The xx value is expressed in decimals.

Default: XON = 11 (decimal)

Example in C

```
strcpy (BufIO; "b9600; pN; s1; d8; o17\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
        Setting configuration
```

\n\r = n for line feed - r for carriage return

f) Parametering the XOFF code (if protocol is enabled - see § g)

Keyword: Fxx or fxx

When the three-quarters of the receiver buffer are full (384 characters) the driver sends the XOFF code to the periphery which stops emission. The xx value is expressed in decimals.

Default: XOFF = 13 (decimal)

Example in C

```
strcpy (BufIO; "b9600; pN; s1; d8; o17; f19\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
```

Setting configuration

g) Parametering the protocol

Keyword: Cx or cx

x = 0 Disables the XON/XOFF protocol and enables the RTS/CTS protocol.

x = 1 Enables the XON/XOFF protocol and disables the RTS/CTS protocol

Example in C

```
strcpy (BufIO; "b9600; pN; s1; d8; o17; f19; c1\n\r");
ioctl (Handler, 3, BufIO, strlen(BufIO));
```

Setting configuration

h) Parametering the timeout

Keyword: Tx or tx

x = 0 Disables the timeout
0 < x < 65536 Enables the timeout (x = 18 = 1s.)

Example in C

```
strcpy (BufIO, "b9600;c0;t18");
ioctl (Handler, 3, BufIO, strlen (BufIO));
```

Setting configuration

5.7.4 Reading the configuration of a serial port

Example in C

ioctl (Handler, 2, BufIO, 80); /* Reading configuration */

printf ("\n configuration XCOMx:%s", BufIO);

a) Format of buffer BufIO

Baud rate; Data length; Parity; Stop; AdrUsart; IrqUsart; XonXoff; Prot; Error; Xoncode; Xoffcode; Buffer; Timeout; \n\r; 0DH0AH (cr, lf)

Baud rate Transfer rate in bits per second initialized

Data length Number of bits contained in one data word (5, 6, 7, 8)

Parity Parity type

N = None
O = Odd
E = Even

\n\r = n for line feed - r for carriage return

Stop Number of stop bits (1, 2)

AdrUsart Port address in hexadecimal

IrqUsart Interrupt number in hexadecimal

XonXoff State of transmit line

0 = emission is possible

1 = Xoff has been detected --> emission is not possible

Prot = 0 Protocol RTS/CTS is active. The driver controls the filling rate of the receiver buffer. When the three-quarters of the buffer are full (384 characters), the driver sets RTS to "0" and the periphery can no longer send data. When the receiver buffer is half-empty (256 characters), the driver sets RTS to "1" and the periphery can send data again.

Prot = 1 Protocol XON/XOFF is active. The driver controls the filling rate of the receiver buffer. If the buffer is three-quarters full (384 characters) the driver sends XOFF on the serial port and the periphery can no

longer send data. If the buffer is half empty (256 characters) the driver sends XON on the serial port and the periphery can again send data.

Errors 1 = Overrun

2 = Parity

4 = Framing

8 = Break

These error codes refer to errors happening in the USART and not in the receiver buffer.

Errors are only valid for the last character received.

XonCode between 0-255

XoffCode Code value currently parametered.

Buffer = 0 Receiver buffer is empty
Buffer = 1 Receiver buffer is neither empty nor full
Buffer = 2 Receiver buffer is full

Timeout = 0 Timeout is disabled
Timeout > 0 Timeout is enabled. Value / 18.2 = timeout in s.

5.8 Timeout

The driver is provided with a timeout system which is set on 1 sec whatever the transfer rate. This value can be changed. When writing (ex WRITE in C) it allows not be blocked if the driver has received an XOFF or if CTS = "0". It is also true with the reading function. Indeed the driver does not remain blocked if the number of characters received is smaller than the number of characters requested during the call of the Read function. After 1 sec, the driver generates a DOS error.

Since the driver uses "int 1CH" of DOS, we advise you to make an interrupt chain with the former address of int 1CH.

Example in C

```
/*
```

Function name	: Timer_int (interrupt)
Task	: This interrupt is set on INT 1CH. It is called by IRQ0 every 18.2/s (55ms)
Input parameters	: No
Input parameters	: No

```
*/

void interrupt Timer_int  (void)
   {
   /* ----------------------------------------- */
   /* The former 1C interrupt routine is called up */
   /* ----------------------------------------- */

    Old_interrupt_1C();

   /* ----------------------------------------- */
   /* Your user program                         */
   /* ----------------------------------------- */
   enable ();
   }
```

5.9 Using XMODE

XMODE is intended for initializing the transfer rate of each serial port. When driver DRV755 is loaded XMODE uses the same parameters as function MODE under MS-DOS.

```
XMODE  XCOM1: 9600,  N,  8,  1,  18
              |      |   |   |   |
              a      b   c   d   e   f
```

a) Name of the port to be initialized

b) Baud rate

c) Parity

d) Word data length

e) Number of stop bits

f) Timeout value

Attention!

If driver DRV750.SYS has not been loaded in CONFIG.SYS, program XMODE can crash down the PC. All the Read and Write operations of your user program are crashed down on the serial ports.

5.10 Operating modes

The driver supports all the operating modes except RS 485.

- Protocol XON/OFF
Can be used in RS 232, RS 422 and current loop.

- Protocol RTS/CTS
Can be used only in RS 232.

- No protocol

If you do not want to use any protocol, communication must occur **without** modem control signals (see jumper adjustment in chapter 3.2 and 7.4), and parameter the RTS/CTS protocol (see chapter 5.7.3).

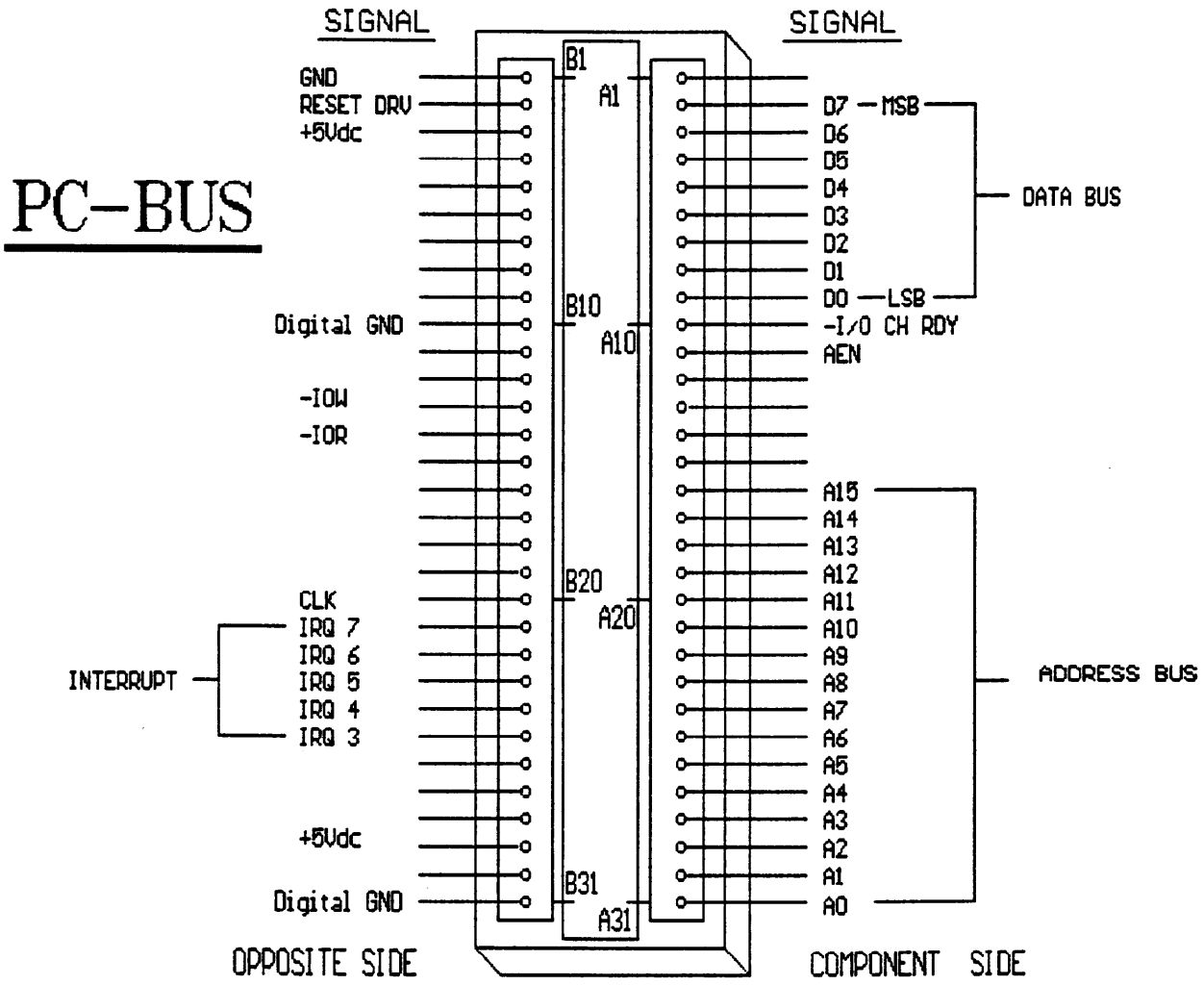
This mode can be set in RS 232, RS 422 and current loop. In this mode the receiver buffer is not managed. If the three-quarters of the buffer are full, it will no longer fill up the charachters that it receives and it will lose them.In RS 422, the driver enables emission with RTS, DIR and GND (see chapter 3.3.4) with the XON/XOFF protocol and without protocol.

5.10.1 Installing several drivers

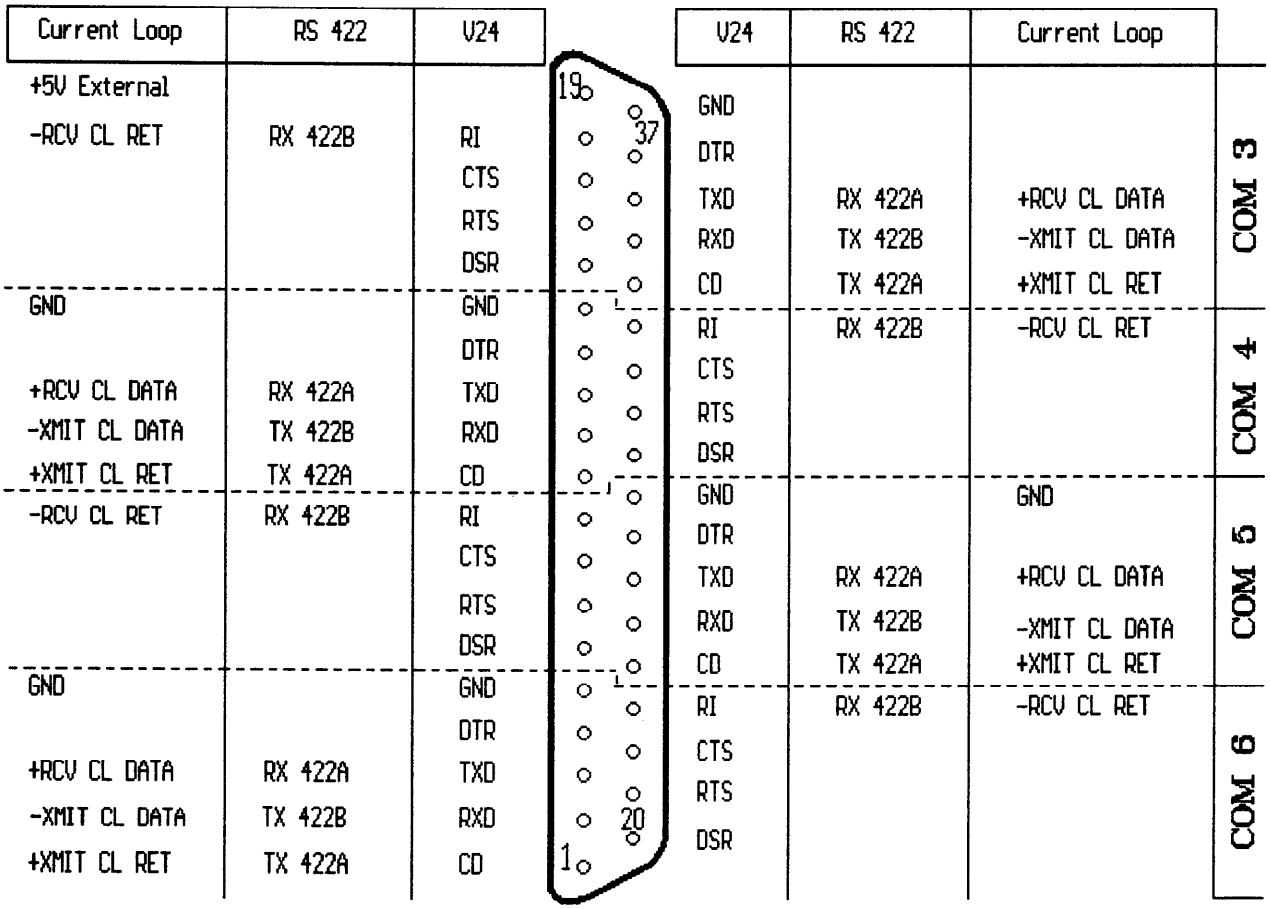
You can use several boards PA 750 in your PC. For each board, you have to install the driver. Driver 1 will support XCOM1 to XCOM4, driver 2 will support XCOM5 to XCOM8 etc.

6. Appendix

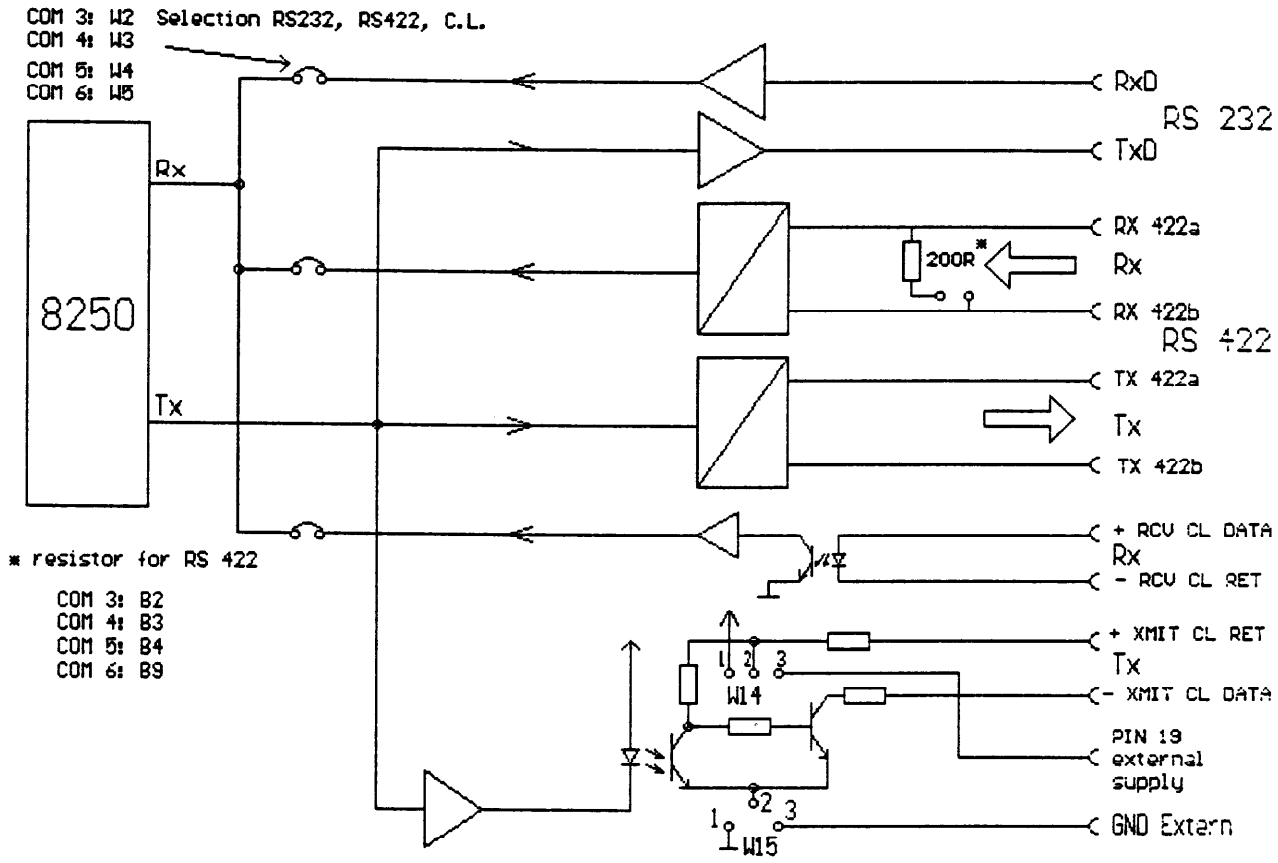
6.1 Signals of the 62 pole direct plug



6.2 Signals of the 37 pole MIN-D pin connector

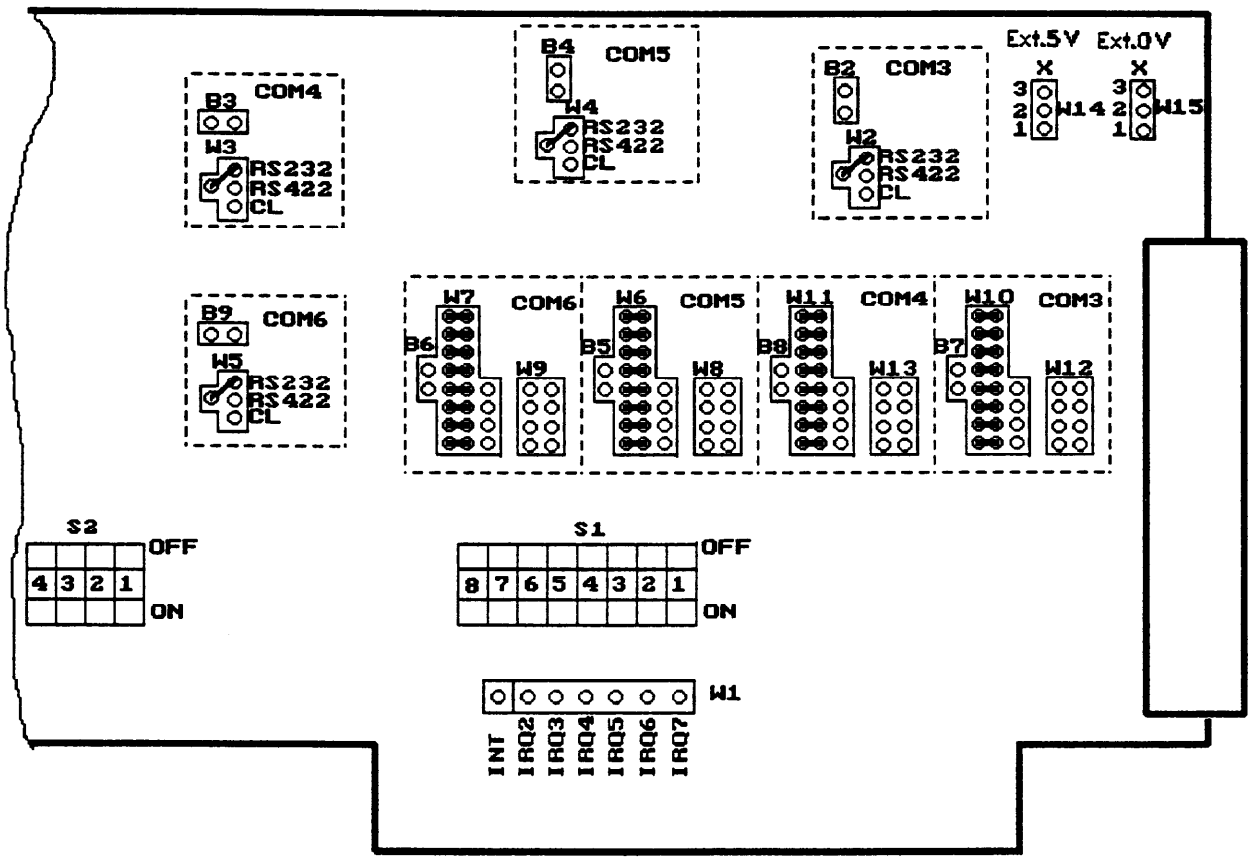


6.3 Block diagram

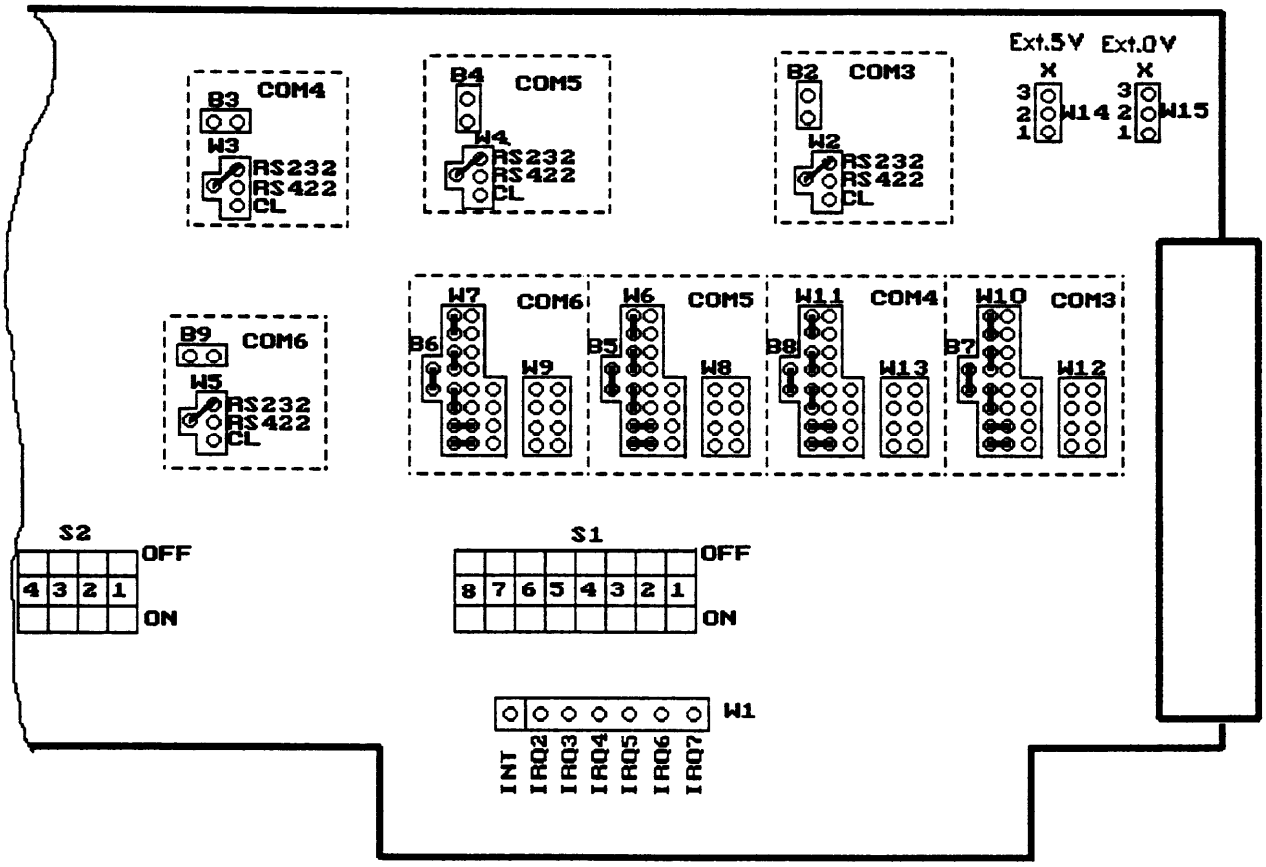


* In the case of several receivers, this resistor has to be switched on to the last receiver (end of line).

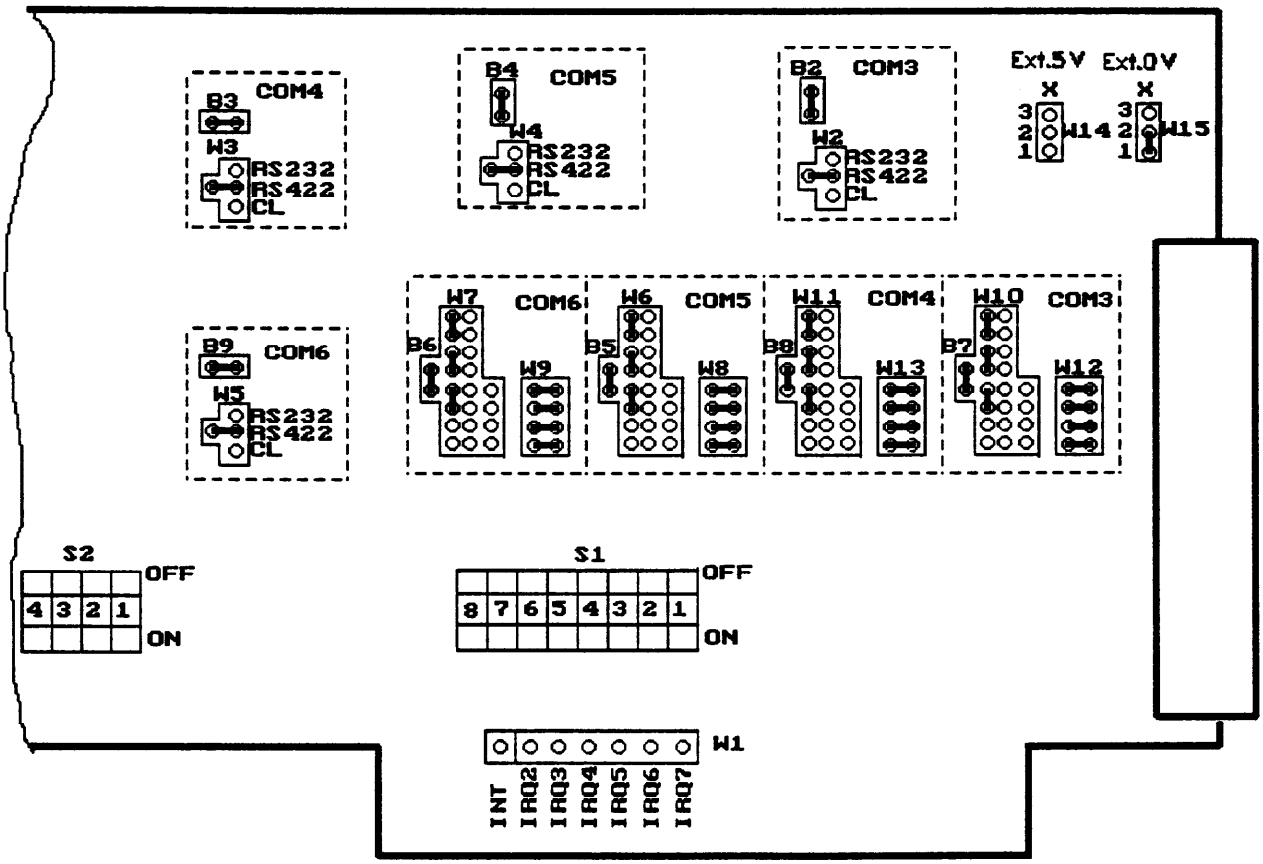
6.4 Wire wrap field: RS 232 (with MODEM control functions)



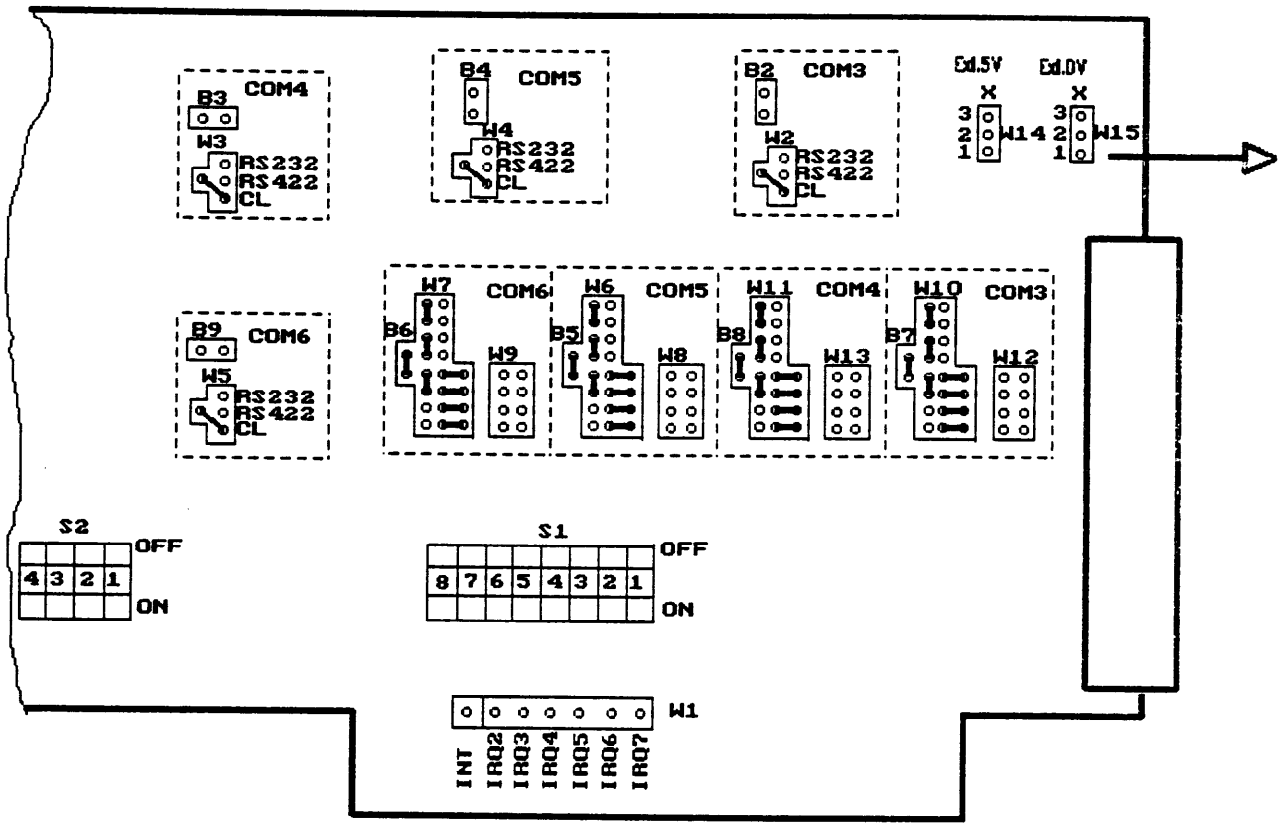
6.5 Wire wrap field: RS 232 (without MODEM control functions)

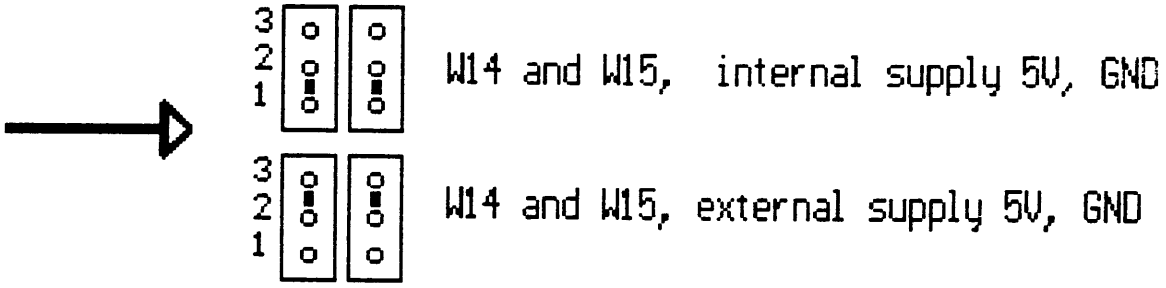


6.6 Wire wrap field: RS 422



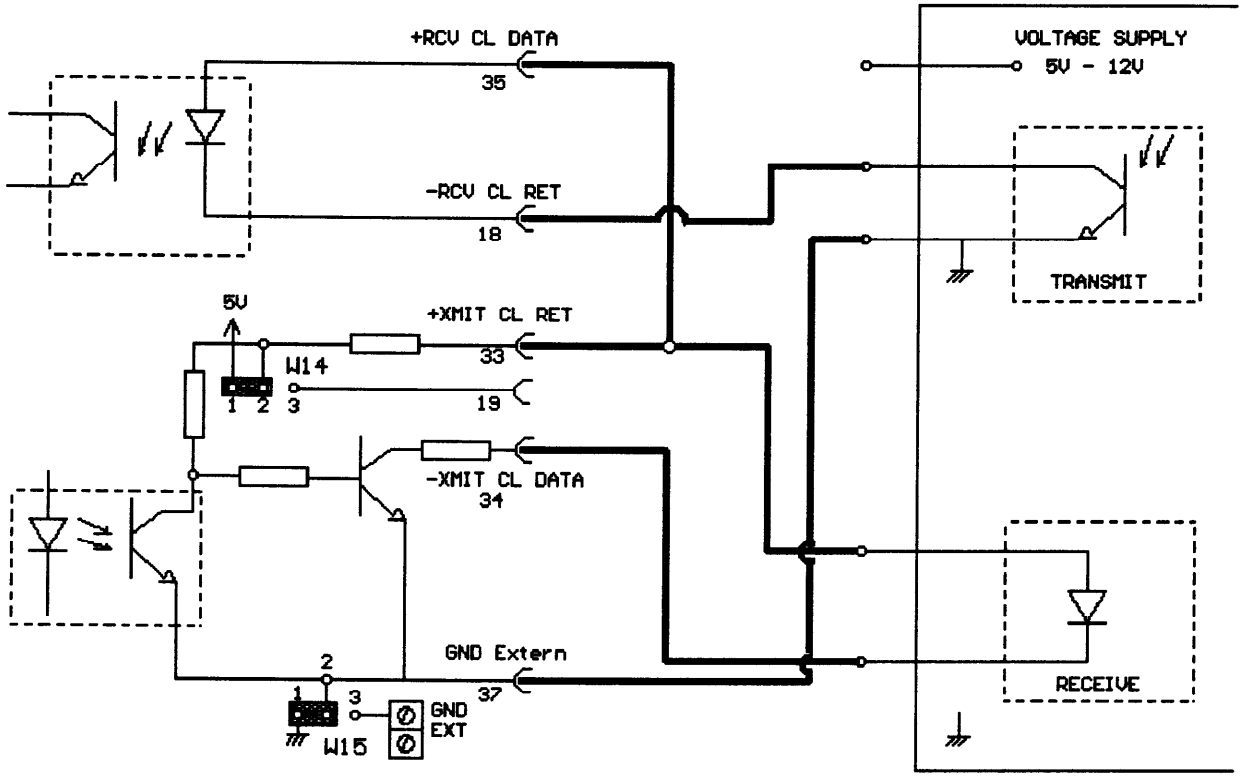
6.7 Wire wrap field : current loop



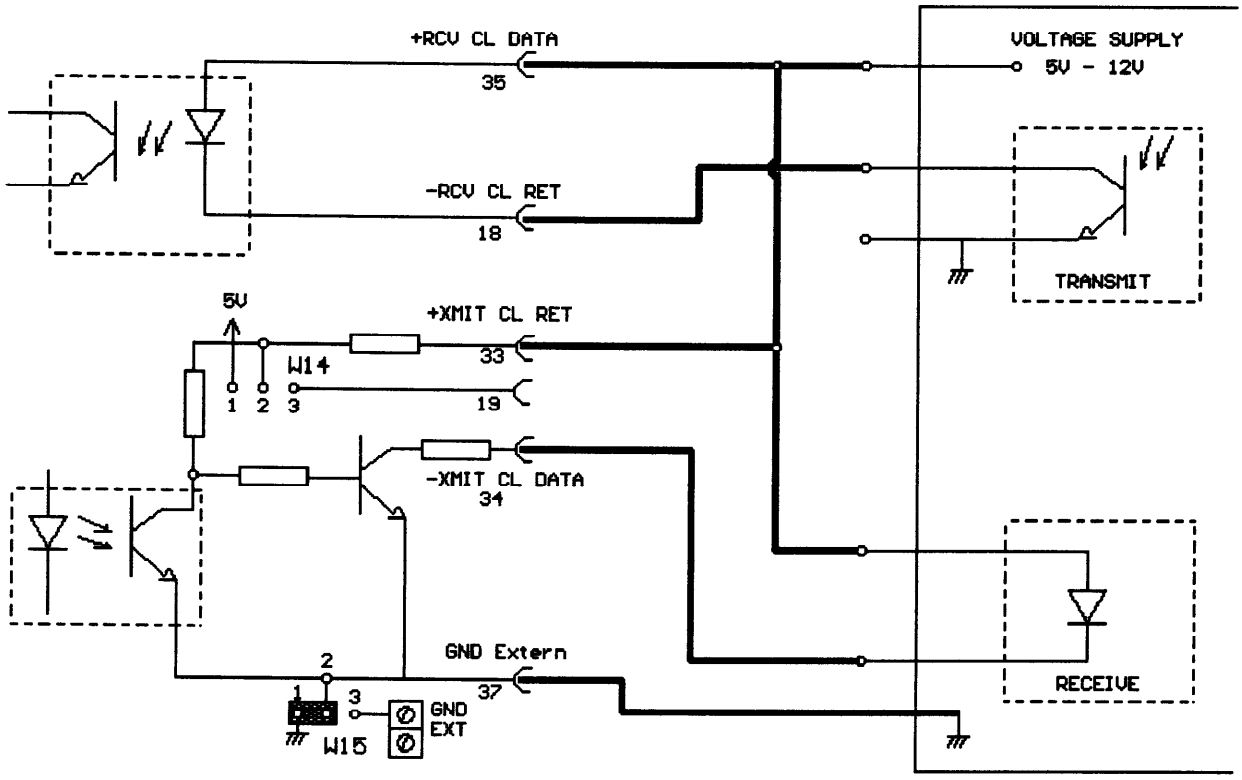


6.8 Current Loop - Example with COM3

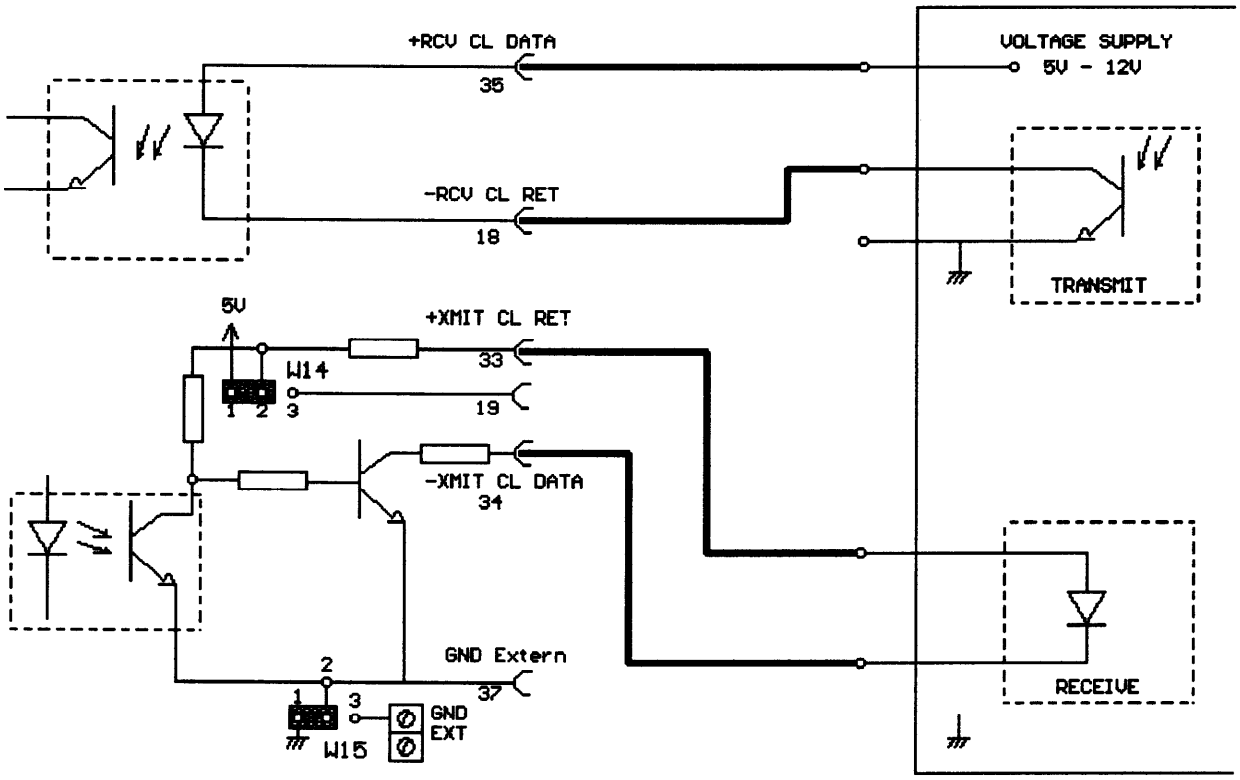
6.8.1 Active transmitter and receiver



6.8.2 Passive transmitter and receiver



6.8.3 Active transmitter and passive receiver



Caution!
Passive transmitter and active receiver do not function.