PCM-3610
PC/104 Isolated Dual-port RS-232 and RS-485/422 Module

Introduction

The PCM-3610 is a PC/104-compatible RS-422/485/232 serial interface module. It works with PC/104 CPU modules or CPU cards which accept PC/104 expansion modules. It provides two independent serial interfaces, accessed through two male DB-9 connectors. You can configure the first port for RS-422, RS-485 or RS-232 operation. The second port offers only RS-422 or RS-485 capability.

The module's industry-standard 16C550 asynchronous communication chip is fully programmable. The module requires no special commands or control codes if you use the standard COM1 and COM2 port addresses.

The module's RS-485 function uses an automatic direction control circuit, so you don't to change any jumpers to switch the module between driver and receiver.

Optical isolation protects your system from ground loops and increases reliability in industrial environments. An additional surge protection circuit protects other devices on the RS-485 network.

Features

- Two isolated serial interfaces
  Channel 1: RS-422, 485 and 232
  Channel 2: RS-422 and RS-485
- Long distance communicationÑup to 4000 feet (1.2 Km) with RS-422/485
- High speed data transmissionÑup to 115,200 Bps.
- Switch selectable addresses (COM1, COM2 or any other address from hex 200 to 3F8)
- 16 bytes FIFOs
- Jumper selectable interrupt level
- Four LEDs indicate status of TX, RX lines (red LED represents TX, green LED represents RX)

RS-42V485

- Supports TX, RX, RTS and CTS signals
- 2-wire or 4-wire operation
- Auto direction control for RS-485
- Overcurrent and surge protection for TX and RX lines

RS-232 (Ch.1 only)

- Supports TX, RX, RTS, CTS, DTR, DSR, DCD and RI signals

Specifications

- **Dimensions:** 3.775" x 3.550" (9.6 cm x 9.0 cm)
- **Bus:** PC/104
- **Baud rate:** 50 to 115,200 bps
- **Character length:** 5, 6, 7 or 8 bits
- **Parity:** Even, odd or none
- **Stop bit:** 1, 1.5 (5-bit data only) or 2
- **UO connectors:** Dual male DB-9
- **Interrupt level:** IRO 3, 4, 5, 6, 7 or 9
- **Clock input:** 1.8432 MHz
- **Isolation**
  Power: 500 V_{DC}
  Signal: 2500 V_{RMS}
- **Optical isolators:** PC 900 on each signal line
- **Driver/receiver**
  Differential input threshold: 0.2 V max.
  Hysteresis: 50 mA typical
  Input impedance: > 12 Kohm without temminators
- **Power consumption (+5 V):** 400 mA typical, 950 mA maximum

Part no. 2000361000 1st Edition Printed in Taiwan May 1994

Initial inspection

We carefully inspected the PCM-3610 both mechanically and electrically before we shipped it. It should be free of marks and scratches and in perfect electrical order on receipt.

Handle the board only by its edges. The static charge on your body may damage its integrated circuits. Keep the card in its anti-static package whenever it is not installed. You can use this package to return the card if it should need repair.

Switches and jumpers

The following chart shows the switches and jumpers corresponding to each serial interface channel:

Ch. 1	Ch. 2	Function
SW1	SW2	I/O base address
JP2	JP3	Interrupt level
JP4	JP5	RS-485 or RS-422
JP10	N/A	RS-232 or RS-485/422

Switch locations appear in the figure below.

RS-23V42V485 selection IJP4, 5,10)

Channel 1

RS-232 JP10 up to 232

RS-422 JP10 down to 485/422
JP4 right to RS-422

RS-485 JP10 down to 485/422
JP4 leR to RS-485

Note: *You must connect channel 1's extema/ signal cable to JP7 for RS-Z32 or JPS for RS-422/4S5.*

Channel 2

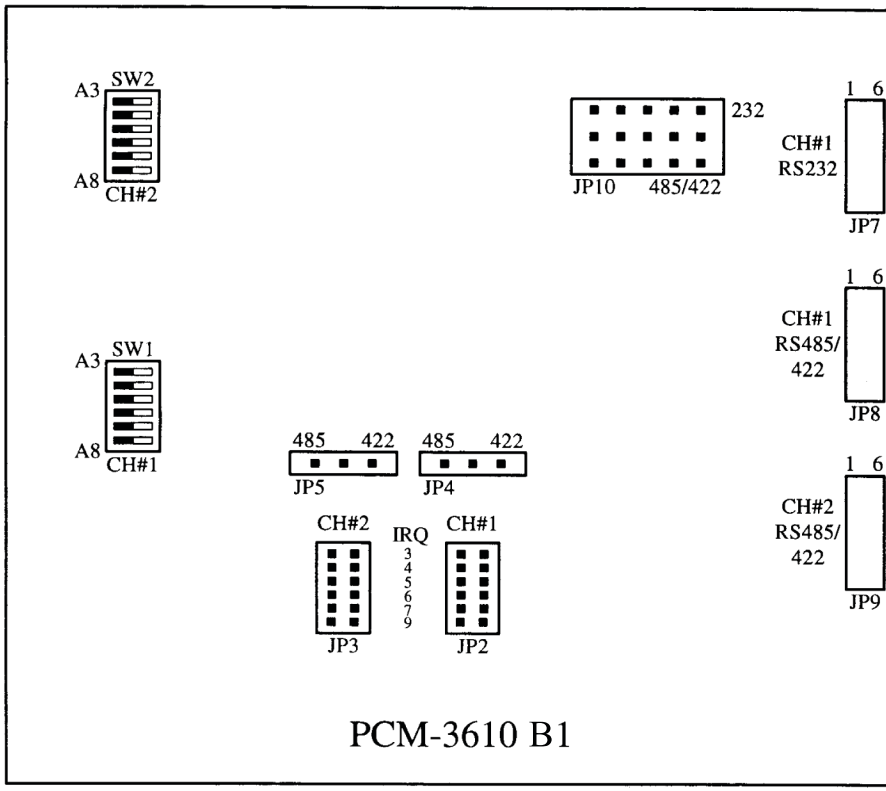
RS-422 JP5 right to RS-422

RS-485 JP5 leR to RS-485

Base address (SW1 and SW2)

Two 6-position DIP switches select the I/O port base address for each channel. SW1 controls Ch. 1, and SW2 controls Ch. 2. See the figure below for swach locations.

You can set the base address anywhere from hex 200 to 3F8. The default set ings are 3F8 (COM1) for Channel 1 and 2F8 (COM2) for Channel 2. The following table shows switch set ings for various base addresses:



Module I/O addresses (SW1 and SW2)

Range (hex)	Switch position					
	1	2	3	4	5	6
200 - 207	0	0	0	0	0	0
208 - 20F	●	0	0	0	0	0
2E8- 2EF	●	0	●	●	●	0
* 2F8 - 2FF (COM2)	●	●	●	●	●	0
3E8 - 3EF	●	0	●	●	●	●
* 3F8 - 3FF (COM1)	●	●	●	●	●	●

O = On ● = 0ff * = defaults

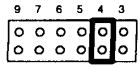
NOTE: *Switches 1-6 control the PC bus address lines as follows:*

Switch	1	2	3	4	5	6
Line	A3	A4	A5	A6	A7	A8

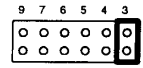
Interrupt level-IRQ (JP2 and JP3)

You can set each port for any interrupt level from 3 to 9, except 8. Jumper JP2 controls Ch. 1, and JP3 controls Ch. 2. Simply short the pins on the jumper corresponding to the interrupt level (see figure below).

JP2 (Ch 1, COM1) IRQ



JP3 (Ch, 2, COM2) IRQ



If you use the ports as standard COM1 and COM2, you will need to set Channel 1 to IRQ 4 and Channel 2 to IRQ 3.

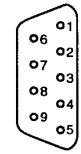
Note: *If your CPU module or card has serial intefface ports, you wiD need to adjust the 1/0 port addresses (or disable the ports) to avoid conflicts.*

Connector pin assignments

You access the PCM-361 0's ports through two extemal male DB-9 connectors. Ground pins are not connected to the DB-9 connector housing for the sake of isolation. With channel 1 you must attach the external cable to the proper proper connector (JP7 or JP8) depending on whether you are using RS-232 or RS-485. See page 2 for details.

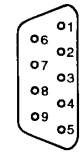
RS-422/485 pin assignments appear below:

RS~22

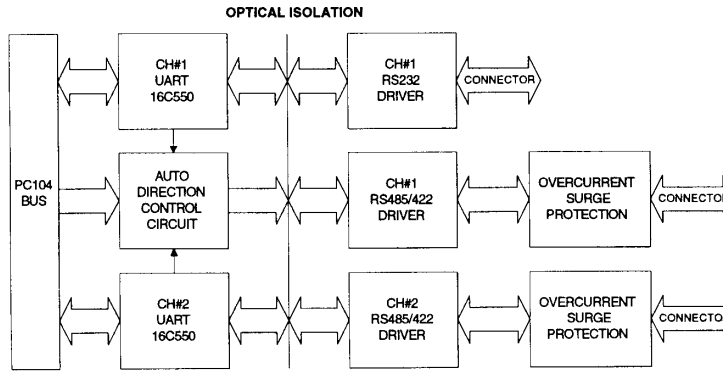


RS~85	**Pin description**
1	TX-(DATA-) or send data - (DTE)
2	TX+(DATA+) or send data + (DTE)
3	RX+ or receive data + (DTE)
4	RX- or receive data - (DTE)
5	GROUND
6	RTS- or ready to send -
7	RTS+ or ready to send +
8	CTS+ or clear to send +
9	CTS- or clear to send -

RS-232 pin assignments appear below:



RS-232	**Pin description**	
1	DCD	receive line signal detector
2	RD	received data
3	TD	transmitted data
4	DTR	dataterminal ready
5	GND	ground
6	DSR	data set ready
7	RTS	request to send
8	CTS	clear to send
9	RI	ring indicator



Block diagram

Hardware installation

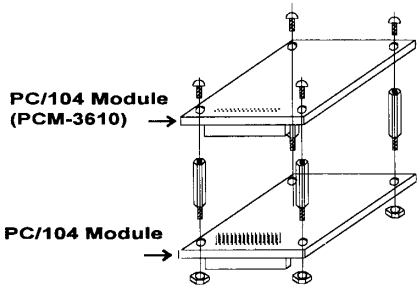
Warning! *TURN OFFyour PC power supply whenever you instaR or remove the PCM-3610 or connect and disconnect cables.*

Installing the module on a CPU card

1. Tum the PC's power off. Tum the power off to any peripheral devices such as printers and monitors.
2. Disconnect the power cord and any other cables from the back of the computer.
3. Remove the system unit cover (see the user's guide for your chassis if necessary).
4. Remove the CPU card from the chassis (if necessary) to gain access to the card's PC/104 connector.
5. Screw the brass spacer (included with the module) into the threaded hole on the CPU card. Do not tighten too much, or the threads may be damaged.
6. Carefully align the pins of the PCM-3610 with the PC/ 104 connector. Slide the module into the connector. The module pins may not slide all the way into the connector; do not push too hard or the module may be damaged.
7. Secure the module to the CPU card to the threaded hole in the CPU card using the included screw.
8. Attach any accessories to the PCM-3610.
9. Reinstall the CPU card and replace the system unit cover. Reconnect the cables you removed in step 2. Tum the power on.

Connecting to another PC/104 module

1. Insert the pins of connector JP6 (on the end of the PCM3610 module) into the piggyback connector on the other PC/104 module.



2. Screw the PCM-3610 to the brass spacer.

This completes the hardware installation. Install the software driver as described in the following section.

Signal wiring

RS-422 has separate transmit and receive lines so both devices can transmit at the same time. The transmit lines from one device connect to the receive lines on the other device. Typical connections are as follows:

Computer A		Computer B	
1	TX-	4	RX-
2	TX+	3	RX+
3	RX+	2	TX+
4	RX-	1	TX-
5	GND	5	GND
6	RTS-	9	CTS-
7	RTS+	8	CTS+ .
8	CTS+	7	RTS+
9	CTS-	6	RTS-

In RS-485 the two devices share a single pair of data lines. One device transmits while the other receives. Typical connections are as follows:

Device A		Device B	
1	TX- (DATA-)	1	TX- (DATA-)
2	TX+ (DATA+)	2	TX+ (DATA+)
5	GND	5	GND

Programming

Programming with COM1 or COM2

If you set the PCM-3610's ports as COM1 and COM2, you can send and receive data using the nommal communication functions found in high-level languages. The following examples use BASIC to demonstrate PCM-3610 programming.

The BASIC communication process starts with the `OPEN ~COMn: , , ..."` statement. This statement assigns a buffer for communication purposes and sets up the communication parameters.

Command format

```
OPEN "COMn: [speed][,parity][,data][,stop]
[,RS][,CS[n]][,DS[n]][,CD[n]][,LF][,PE]"
AS [#]filenum
```

Example:

```
OPEN "COM1:9600,N,8,,CS,DS,CD" AS #1
```

Where:

COMn: n is 1 or 2, indicating either COM1 or COM2

speed: An integer constant specifying the baud rate in bits per second

parity: One of the following characters:
S: space
0: odd
M: mark
E: even
N: none

data: An integer constant indicating the number of data bits. Valid values are 4, 5, 6, 7 and 8. The default is 7.

stop: The number of stop bits. Valid values are 1 and 2. The default is 2 for 75 and 110 bps, 1 for all others.

RS: Suppresses RTS

CS: Controls CTS

DS: Controls DSR

CD: Controls CD

LF: Sends a line feed following each carriage return

PE: Enables parity checking

filenum: filenum is an integer expression which evaluates to a valid file number

You must put the speed, parity, data and stop parameters in this position and order, but you can put the RS, CS, DS, CD, LF and PE parameters in any order. The n argument in the CS, DS and CD parameters specifies the number of milliseconds to wait for the signal before returning a “device timeout” error. n may range from 0 to 65535. If you omit n I; or set it equal to 0, then the line status is not checked at all.

Refer to the IBM BASIC reference manual for more detailed information.

Programming example—standard COM ports

You can use the following BASIC program to test the PCM-3610's send and receive functions.

```
10      \************************************************
20      \* Program: DEMO01.BAS (for RS485 mode)          *
30      \* Description: This demo program transmits a    *
40      \* string through COM1 and receives it through   *
50      \* COM2                                          *
60      \*                                               *
70      \* Jumper settings                               *
hO      \* JP4: RS485, JPS: RS485                        *
80      \* JP10: RS485/RS422                             *
90      \*                                               *
100     \* Signal wiring                                 *
110     \* COMl                          COM2            *
120     \* 1 TX- (DATA-)   < - - - >     1 TX- (DATA-)   *
130     \* 2 TX+ (DATA+)   < - - - >     2 TX+ (DATA+)   *
140     \* 5 GND           < - - - >     5 GND           *
150     \************************************************
160     \Set the proper parameters
170     \COM1 L COM2: baud rate=9600 ; no parity check;
180     \Data bit=8; stop bit=1
190     \Ignore the CTS, RTS and DSR signals.
200     OPEN "COM1:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #1
210     OPEN "COM2:9600,N,8,1,RS,CS,DS,CD" FOR RANDOM AS #1
220     INPUT "INPUT COMMAND:":CMD$
230     IF CMD$="Q" OR "q" THEN CLOSE:END ELSE GOSUB 250
240     GOSUB 300:GOTO 220
250     \****** Transmit data sub-routine ******
260     PRINT #1, CMD$
270     RETURN
300     \****** Receive data sub-routine ******
310     T=TIMER:TEMP$="":RX$=""
320     IF TIMER>T+.5 THEN PRINT "TIMEOUT ERROR":RETURN
330     IF LOC(2)>0 THEN TEMP$=INPUT$(1,#2) ELSE GOTO 320
340     RX$=RX$+TEMP
350     IF TEMP$=CHR$(13) THEN GOTO 360 ELSE GOTO 320
360     PRINT "RECEIVE DATA:";RX$:RETURN
```

Using other 1/0 port addresses

If you are going to use l/O ports other than COM1 or COM2, you will need to directly program the registers of the i-~ PCM3610's 16C550 chip.

See page 7 for information on the fommat and programming of these registers. See page 8 if you have trouble finding a ~. free l/O port base address.

You can use the following program as a base as you develop your own driver. The program exchanges data (the numbers 0 to 256) between two ports. It uses l/O port addresses hex 2E8 and 3E8. Set JP4, JP5 and JP10 for RS485 or RS-422 mode (described on page 2).

Programming exampleÑarbitrary 1/0 ports

```
10      \****************************************
20      \Clear the screen
30      CLS
40      \Set the I/O port base addresses for
50      \both cards
60      PORT1%=hH2E8
70      PORT2%=hH3E8
80      \Read all registers once to
90      \clear any random data
100     FOR I=PORT1% TO PORT1%+6
110     DUM=INP(I)
12 0    NEXT I
130     FOR I=PORT2% TO PORT2%+6
140     DUN=INP(I)
150     NEXT I
160     \Initialize the registers of
170     tportl. First, set DLAB = 1 so the
180     \desired baud rate can be programmed.
190     OUT PORT1%+3,&H80
200     \Write the value of divisor into
210     \registers: hex 180 = dec 384 = 300 BAUD
220     OUT PORT1%,&H80:0UT PORT1%+l,&H1
230     \Set word length = 8 bits, stop bits = 2,
240     \even parity, DLAB = 0.
250     OUT PORT1%+3, hHlF
260     \Do the same thing for port2.
270     OUT PORT2%+3,hH80
280     OUT PORT2%,hH80:0UT PORT2%+1,&H1
290     OUT PORT2%+3,&H1F
300     \Loop over data (0-255) and send it
310     \from portl to port2
320     FOR BYTE=0 TO 255
330     \Wait until the transmitter buffer
340     \is empty.
350     IF (INP (PORT1%+S) AND 32) =0 GOTO 350
360     \Output the data through portl.
370     OUT PORT1%, BYTE
380     \See if the data is available by checking
390     \the Data Ready bit.
400     IF (INP(PORT2%+5) AND 1)=0 GOTO 400
410     J=INP(PORT2%)
420     \Print out the data byte received
430     PRINT ~port ":HEX$(PORT2%)" = ":HEX$(J)
440     \If the value sent <> the received value then error
450     IF J<>BYTE GOTO 620
460     NEXT BYTE
470     \Loop over data (0-255) and send it
480     \from port2 to portl.
490     FOR BYTE=0 TO 255
500     \See if the transmitter buffer is empty.
510     IF (INP(PORT2%+5) AND 32)=0 GOTO 510
520     OUT PORT2%, BYTE
530     \See if the data is available by
540     \checking the Data Ready bit.
550     IF ( INP ( PORT1%+S) AND 1 ) =0 GOTO 550
560     J=INP(PORT1%)
570     PRINT "port":HEXS (PORT1%)"= ":HEXS (J)
580     IF J<>BYTE GOTO 620
590     NEXT BYTE
600     \If everything is OK, then stop.
610     END
620     PRINT "Data transmission error!":BEEP:END
```

Programming exampLE—communication

The following pair of example programs show how you can set up communication between two computers. The first program sends data then receives data. The second program receives data then sends data. Run the first program on one computer and the second on another.

Program for first computer

```
10        \********** STEP 1: INITIALIZATION ***********
20        \Clear screen
3 0       CLS
40        \Define variables A to Z as integer
50        DEFINT A-Z
60        \Set port base address (must match hardware)
70        PORT = &H3FB
80        \Set baud rate to 300
90        OUT PORT + 3, LH80
100       OUT PORT, LH80
110       OUT PORT, 1
120       OUT PORT + 3, LH1F
130       \******** STEP 2: SEND DATA ********
150       FOR I = 65 TO 90
160       \
170       GOSUB 200
18 0      NEXT I
190       GOTO 260
200       STATUS = INP(PORT + 5) AND &H20
210       IF STATUS = 0 THEN 200
220       OUT PORT, I
230       FOR J = 0 TO 1200: NEXT J
240       RETURN
250       \****** STEP 3: RECEIVE DATA ******
260       FOR I = 65 TO 90: GOSUB 280: NEXT       I
270       END
280       STATUS = INP(PORT + 5)
290       IF (STATUS AND LH1E) THEN 280
300       IF (STATUS AND LH1) = 0 THEN 280
310       D = INP(PORT)
320       PRINT "DATA= "; CHR$ (D)
330       RETURN
```

Program for second computer

```
10        \********** STEP1: INITIALIZATION **********
20        \Clear screen
3 0       CLS
40        \Define variables A TO Z as iDteger
50        DEFINT A-Z
60        \Set port base address (must match hardware)
70        PORT = LH2F8
80        \Set baud rate to 300
90        OUT PORT + 3, LH80
100       OUT PORT, &H80
110       OUT PORT, l
120       OUT PORT + 3, &HlF
130       \***** STEP 2: RECEIVE DATA FROM ANOTHER PC *****
140       FOR I = 65 TO 90: GOSUB 190: NEXT I
150       PRINT: PRINT: PRINT
160       PRINT.DATA RECEIVES END, THEN DATA SEND BEGINNING."
170       PRINT: PRINT "PRESS ANY KEY. . ."
180       IF INXEY$ = "" THEN 180 ELSE 260
190       STATUS = INP(PORT + 5)
200       IF STATUS AND LH1E THEN GOTO 190
210       IF (STATUS AND LH1) = 0 THEN 190
220       d = INP(PORT)
230       PRINT "DATA= ": CHR$(d)
240       RETURN
250       \*************** STEP 3: SEND DATA ***************
260       FOR I = 65 TO 90
270       d = I
280       GOSUB 310
290       NEXT I
300       END
310       STATUS = INP(PORT + 5) AND LH20
320       IF STATUS = 0 THEN 310
3 3 0     OUT PORT, d
340       FOR J = 0 TO 1200: NEXT J
3 50      RETURN
```

C language test program

You can use the following C program to test the PCM-3610's send and receive functions.

```
/*************************************************/
/*    Program: DEMO01.C (For RS485/RS422)        */
/*    Description: This demo program transmits a string  */
/*    to COM1 and receives a string Erom COM2    */
/*    Compiler: Turbo C 2.0                      */
/*                                               */
/*    RS-485 jumper and switch settings, signal wiring   */
/*    SW1  -  3F8        COM1:          COM2:    */
/*    sW2  -  2F8        1 DATA-   <->  1 DATA-  */
/'    JP2  -  IRQ4       2 DATA+   <->  2 DATA+  */
/'    JP3  -  IRQ3       5 GND     <->  3 GND    */
/'    JP4  -  485                                */
/'    JP5  -  485                                */
/'    JP10 -  485/422                            */
/'                                               */
/'    RS-422 jumper and switch settings, signa1 wiring   */
/'    SW1  -  3F8        COM1:          COM2:    */
/'    sW2  -  2F8        1 TX-     <->  4 RX-    */
/'    JP2  -  IRQ4       2 TX+     <->  3 RX+    */
/'    JP3  -  IRQ3       3 RX+     <->  2 TX+    */
/'    JP4  -  422        4 RX-     <->  1 TX-    */
/'    JP5  -  422        5 GND     <->  3 GND    */
/'    JP10 -  485/422                            */
/*************************************************/

#include <dos.h>
#include <io.h>
#include <stdio.h>
#include <conio.h>

#define    TIME_OUT 10000

static     int base0 = 0x3f8;      /* Base address of port 0 */
static     int basel = 0x2f8;      /* Base address of port 1 */
ste-tic    char rec[l6]: /* Buffer for received string '/
static     char cmdll61 i /* Buffer for transmitted string /

void main ( )
{
  int i; /* Counter for character being sent/received */
  char flag; /* Flag for end of output/input data */
  int timeout; /* Timeout counter */

  outport ( (base0+2), 0xc9); /* enable port 0 FIFO */
  outport((basel+2), 0xc9); /* enable port 1 FIFO */
  /* set communication parameters for port 0 */
  outp(base0+3, 0x80) i /* Set DLAB=1 */
  /. Set baud = 115200 */
  outp(base0, 0x01);
  /* Set data=8, stop=1, no parity */
  outp(base0+3, 0x03);
  /* Disable port 0 interrupt */
  outp(base0+1, 0x00);
  /* Set communication parameters for port 1 */
  outp(basel+3, 0x80) 1 /* Set DLAb=1 */
  /* Set baud = 115200 */
  outp(basel, 0x01);
  /* Set data=8, stop=1, no parity */
  outp (basel+1, 0 );
  outp(basel+3, 0x03 );
  /* Disable port 1 interrupt */
  outp(basel+l, 0x00);
```

```
printf("\nEnter ~ string to be transmitted "
       "(1S characters or less) or Q to quit:");
gets (cmd);
while (cmd[O] != "q" && cmd[0] != "Q")
{
       i=O;
       cmd[strlen(cmd)] = 0xOd;
       flag=1;
       while (flag)
       {
           outportb(baseO, cmd[i] );  /* Send data */
           i (cmd[i] == 0x0d)
           flag=o;
       i++;
       }
       i=O;
       flag=1;
       timeout=TIME _ OUT;
       while (flag)
       {
           /* Check if receiver data is ready */
           if ((inportb(basel+5) & 1) !=0)
           {
               rec [i]=inportb (basel ); /* Receive data */
               if (rec[i] ==0x0d)
               {
                    rec[i+1]='\O';
                    flag=0;
                    printf ("\nReceived data: %s\n", rec);
                    }
                    i++;
               }
               else
               {  /* Check timeout */
                    timeout--;
                    if (timeout == 0)
                    {
                        flag = 0;
                        printf("\nTimeout error\n");
                        }
                    }
               }
printf ("\nEnter a string to be transmitted "
       "(15 characters or less) or Q to quit:");
gets (cmd);
}
}
```

Register strcture and format

This section gives short description of each of the module's registers. For more infommation please refer to the data book for the STARTECT 1 6C550 UART chip.

All registers are one byte. Bit O is the least significant bit, and bit 7 is the most significant bit. The address of each register is specified as an offset from the port base address (BASE), selected with DIP switch SW1 or SW2.

DLAB is the "Divisor Latch Access Bit", bit 7 of BASE+3.

BASE+0 Receiver buffer register when DLAB=0 and the operation is a read.

BASE+O Transmitter holding register when DLAB=0 and the operation is a write.

BASE+O Divisor latch bbs 0 - 7 when DLAB=1.

BASE+1 Divisor latch bHs 8 -15 when DLAB=1.

The two bytes BASE+O and BASE+1 together fomm a 1 6-bit number, the divisor, which detemmines the baud rate. Set the divisor as follows:

Baud rate	Divisor
50	2304
75	1 536
110	1047
133.5	857
150	768
300	384
600	192
1200	96
1800	64
2000	58
2400	48
3600	32
4800	24
7200	16
9600	12
19200	6
38400	3
56000	2
115200	1

BASE+1 Interrupt Status Register (ISR) when DLAB=0
- bit 0 Enable received-data-available interrupt
- bit 1 Enable transmitter-holding-register-empty interrupt
- bit 2 Enable receiver-line-status interrupt
- bit 3 Enable modem-status interrupt

BASE+2 FIFO Control Register (FCR)
- bit 0 Enable transmit and receive FIFOs
- bit 1 Clear contents of receive FIFO
- bit 2 Clear contents of transmit FIFO

bit 3 Change RXRDY and TXRDY from mode O to mode 1.
bits 6-7 Set trigger level for receiver FIFO interrupt.

Bit7	Bit6	FIFO trigger level
0	0	01
0	1	04
1	0	08
1	1	14

BASE+3 Line Control Register (LCR)
bit 0 Word length select bit
bit 1 Word length select bit 1

Bit1	Bit0	Word length (bits)
0	0	5
0	1	6
1	0	7
1	1	8

bit 2 Number of stop bits
bit 3 Parity enable
bit 4 Even parity select
bit 5 Stick parity
bit 6 Set break
bit 7 Divisor Latch Access Bit (DLAB)

BASE+4 Modem Control Register (MCR)
bit 0 DTR
bit 1 RTS

BASE+5 Line Status Register (LSR)
bit 0 Receiver data ready
bit 1 Overrun error
bit 2 Parityerror
bit 3 Framing error
bit 4 Break interrupt
bit 5 Transmitter holding register empty
bit 6 Transmitter shiff register empty
bit 7 At least one parity error, framing error or break indication in the FIFO

BASE+6 Modem Status Register (MSR)
bit 0 Delta CTS
bit 1 Delta DSR
bit 2 Trailing edge ring indicator
bit 3 Delta received line signal detect
bit 4 CTS
bit 5 DSR
bit 6 Ri
bit 7 Received line signal detect

BASE+7 Temporary data register

Standard PC I/O port assignments

The following chart shows the I/O addresses used by standard PC peripheral devices.

I/O address (hex)	Assignment
000-1 FF	used by base system board
200	not used
201	game control
202-277	not used 278-27F second printer port
280-2F7	not used
2F8-2FF	COM2
300-377	not used
378-37F	printer port
380-3AF	not used 3B0-3BF monochrome adapter and printer
3C0-3CF	not used
3D0-3DF	color and graphics adapters
3E0-3EF	not used 3F0-3F7 floppy diskette drive
3F8-3FF	COM1: