PCM-3640
PC/104 4-port RS-232 Module

Introduction

The PCM-3640 is a PC/104-compatible 4-port RS-232 serial interface module. It works with PC/104 CPU modules or CPU cards which accept PC/104 expansion modules. It provides four independent serial interfaces, accessed through male DB-9 connectors.

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 ~ COM4 port addresses.

Features

- Four RS-232 serial interfaces
- High speed data transmission—up to 115,200 Bps.
- Switch selectable addresses (COM1 ~ COM4 or any other address from hex 200 to 3F8)
- 16 bytes FIFOs
- Jumper selectable interrupt level
- Eight LEDs indicate status of TX, RX lines (red LED represents TX, green LED represents RX)
- Supported by PC-ComLib serial communication programming library (optional)

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
- **I/O connectors**: Four male DB-9
- **Interrupt level**: IRQ 3, 4, 5, 6, 7 or 9
- **Clock input**: 1.8432 MHz
- **Power consumption**: +5 V @ 220 mA max.

Initial inspection

We carefully inspected the PCM-3640 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 used to configure the PCM-3640:

<table>
<thead>
<tr>
<th>Switch</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>SW1</td>
<td>I/O base address (enhanced mode)</td>
</tr>
<tr>
<td>JP1</td>
<td>Channel 1 Interrupt level</td>
</tr>
<tr>
<td>JP2</td>
<td>Channel 2 Interrupt level</td>
</tr>
<tr>
<td>JP3</td>
<td>Channel 3 Interrupt level</td>
</tr>
<tr>
<td>JP4</td>
<td>Channel 4 Interrupt level</td>
</tr>
</tbody>
</table>

Board Layout

[Diagram of board layout]
Default jumper settings

The PCM-3640 will be shipped in standard mode, with the following I/O address and IRQ settings:

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O address</th>
<th>IRQ no.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>3F8</td>
<td>IRQ4</td>
</tr>
<tr>
<td>Port 2</td>
<td>2F8</td>
<td>IRQ3</td>
</tr>
<tr>
<td>Port 3</td>
<td>3E8</td>
<td>IRQ12</td>
</tr>
<tr>
<td>Port 4</td>
<td>2E8</td>
<td>IRQ15</td>
</tr>
</tbody>
</table>

Jumper and Switch settings

The PCM-3640 can be used in two modes: standard or enhanced mode. In standard mode the I/O addresses are compatible with the standard PC communication ports, COM1 ~ COM4. In enhanced mode you can select a different base address. The offset of each port from the base address is fixed.

Standard / Enhanced mode selection

Switch 7 of DIP switch SW1 selects between standard and enhanced mode.

Standard mode

In standard mode, the I/O address of the ports are as follows:

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O address</th>
<th>Interrupt No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>3F8</td>
<td>Selectable (see p.3)</td>
</tr>
<tr>
<td>Port 2</td>
<td>2F8</td>
<td>Selectable (see p.3)</td>
</tr>
<tr>
<td>Port 3</td>
<td>3E8</td>
<td>Selectable (see p.3)</td>
</tr>
<tr>
<td>Port 4</td>
<td>2E8</td>
<td>Selectable (see p.3)</td>
</tr>
</tbody>
</table>

Enhanced mode

Base address selection(SW1)

In enhanced mode, you can select a different base address. The base address determines the address for each of the four ports. The I/O addresses for the four ports are as follows:

<table>
<thead>
<tr>
<th>Port</th>
<th>I/O address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 1</td>
<td>Base + 00H</td>
</tr>
<tr>
<td>Port 2</td>
<td>Base + 08H</td>
</tr>
<tr>
<td>Port 3</td>
<td>Base + 10H</td>
</tr>
<tr>
<td>Port 4</td>
<td>Base + 18H</td>
</tr>
</tbody>
</table>

You use switches 1~6 of DIP switch SW1, a 7-position DIP switch, to set the base address. You can set the base address anywhere from hex 200 to 3F8.

To set the base address, you have to calculate the base address as follows:

**NOTE:** On the PCM-3640 the address line A9 does not appear on the DIP switch as it is permanently hard-wired to HEX 200 on the card.

The following table shows different base address settings.

<table>
<thead>
<tr>
<th>Port base address (SW1)</th>
<th>Base Address</th>
<th>A3</th>
<th>A4</th>
<th>A5</th>
<th>A6</th>
<th>A7</th>
<th>A8</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>200-207</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td>208-20F</td>
<td>○</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td></td>
<td>●</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
<tr>
<td></td>
<td></td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
<td>○</td>
</tr>
</tbody>
</table>

*3F8-3FF

Note: If your CPU module or card has serial interface ports, you will need to adjust the I/O port addresses (or disable the ports) to avoid conflicts.
Interrupt level selection (JP1 ~ JP4)

You can set the interrupt level for each port from 3 to 15, except 8, 13 and 14. Jumpers JP1, JP2, JP3 and JP4 sets the interrupt level for port 1, port 2, port 3 and port 4 respectively.

Simply short the pins on the jumper corresponding to the interrupt level required (as illustrated below).

Note: Do not use interrupts that are used by other cards/ports, unless you have made provision for interrupt sharing in your programs.

RS-232 Signal wiring

Since the RS-232 interface is not strictly defined, many devices have their own connection methods which may ignore some signal lines or define reserved lines to other functions. It is best to refer to the user’s manual for your device for installation instructions. You may find the following helpful.

In general, DTE (Data Terminal Equipment) refers to the device that is leading the communication. Examples include PC’s, terminals and some printers. DCE refers to the device being communicated with or controlled. Examples include modems, DSU’s (digital service units), printers and lab/factory equipment.

In some situations you may be able to get by with just three lines: data on TxD, a Signal Ground and a handshaking line. Examples are printer or plotter connections, troubleshooting and situations where you require only one-wire communication.

Terminal or PC (DTE) connections

<table>
<thead>
<tr>
<th>PCM-3640 (DTE): (DB-9)</th>
<th>Terminal (DTE):DB-25</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin</td>
<td>Signal</td>
</tr>
<tr>
<td>3</td>
<td>TxD</td>
</tr>
<tr>
<td>2</td>
<td>RxD</td>
</tr>
<tr>
<td>7</td>
<td>RTS</td>
</tr>
<tr>
<td>8</td>
<td>CTS</td>
</tr>
<tr>
<td>6</td>
<td>DSR</td>
</tr>
<tr>
<td>5</td>
<td>GND</td>
</tr>
<tr>
<td>4</td>
<td>DTR</td>
</tr>
<tr>
<td>1</td>
<td>DCD</td>
</tr>
</tbody>
</table>

Modem connections

<table>
<thead>
<tr>
<th>PCM3640: DB-9 Male Modem (DCE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pin</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>
Connecting to another PC/104 module

1. Insert the pins of connector JP6 (on the end of the PCM-3640 module) into the piggyback connector on the other PC/104 module.

2. Screw the PCM-3640 to the brass spacer.
   This completes the hardware installation.

Programming

Programming with COM1 or COM2

If you set the PCM-3640's ports as COM1 and COM2, you can send and receive data using the normal communication functions found in high-level languages. The following examples use BASIC to demonstrate PCM-3640 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
  - O: 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

---

### Hardware installation

**Warning!**

TURN OFF your PC power supply whenever you install or remove the PCM-3640 or connect and disconnect cables.

Installing the module on a CPU card

1. Turn the PC's power off. Turn 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-3640 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-3640.
9. Reinstall the CPU card and replace the system unit cover. Reconnect the cables you removed in step 2. Turn the power on.
You must put the speed, parity, data and stop parameters in this position and order, but you can put the RS, CS, 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 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-3640's send and receive functions.

```basic
10 '******************************
20 ' Program: DEMO01.BAS
30 ' Description: This demo program transmits a string through COM1 and receives it through
40 ' string through COM2.
50 ' COM2
60 '********************************************************
70 ' stiff = 8; stop bit=1
80 ' Set port base address (must match hardware)
90 ' Write the value of divisor into
100 ' Output the data through port1.
110 ' Wait until the transmitter buffer
120 ' Loop over data (0-255) and send it
130 ' Print out the data byte received
140 ' If the value sent <> the received value then
150 ' See if the data is available by checking
160 ' See if the transmitter buffer is empty.
170 ' Check if the CTS, RTS and DSR signals. (See page 7 for information on the format and programming
detailed information.)
180 'Set proper parameters
190 ' Read all registers once to
200 'Both cards
210 'Clear screen
220 'Clear the screen
230 'Input data from COM2 to COM1
240 'Do the same thing for port2.
250 'Power-up
260 'Program
270 'Receive data from COM1 to COM2
280 'Send data from COM2 to COM1
290 '****** Transmit data sub-routine ******
300 '****** Receive data sub-routine ******
310 'If everything is OK, then stop.
320 'If everything is OK, then stop.
330 'Clear the screen
340 'Clear the screen
350 'Clear the screen
360 'Clear the screen
370 'Clear the screen
380 'Clear the screen
390 'Clear the screen
400 'Clear the screen
410 'Clear the screen
420 'Clear the screen
430 'Clear the screen
440 'Clear the screen
450 'Clear the screen
460 'Clear the screen
470 'Clear the screen
480 'Clear the screen
490 'Clear the screen
500 'Clear the screen
510 'Clear the screen
520 'Clear the screen
530 'Clear the screen
540 'Clear the screen
550 'Clear the screen
560 'Clear the screen
570 'Clear the screen
580 'Clear the screen
590 'Clear the screen
600 'Clear the screen
610 'Clear the screen
620 'Clear the screen
630 'Clear the screen
640 'Clear the screen
650 'Clear the screen
660 'Clear the screen
670 'Clear the screen
680 'Clear the screen
690 'Clear the screen
700 'Clear the screen
710 'Clear the screen
720 'Clear the screen
730 'Clear the screen
740 'Clear the screen
750 'Clear the screen
760 'Clear the screen
770 'Clear the screen
780 'Clear the screen
790 'Clear the screen
800 'Clear the screen
810 'Clear the screen
820 'Clear the screen
830 'Clear the screen
840 'Clear the screen
850 'Clear the screen
860 'Clear the screen
870 'Clear the screen
880 'Clear the screen
890 'Clear the screen
900 'Clear the screen
910 'Clear the screen
920 'Clear the screen
930 'Clear the screen
940 'Clear the screen
950 'Clear the screen
960 'Clear the screen
970 'Clear the screen
980 'Clear the screen
990 'Clear the screen
```

**Programming example—arbitrary I/O ports**

If you are going to use I/O ports other than COM1 or COM2, you will need to directly program the registers of the PCM-3640's 16C550 chip.

See page 7 for information on the format and programming of these registers. See page 8 if you have trouble finding a free I/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 I/O port addresses hex 2E8 and 3E8. Set JP4, JP5 and JP10 for RS485 or RS-422 mode (described on page 2).

```basic
10 '******************************
20 ' Program: DEMO01.BAS
30 ' Description: This demo program transmits a string through COM1 and receives it through COM2.
40 ' string through COM2.
50 ' COM2
60 '********************************************************
70 ' stiff = 8; stop bit=1
80 ' Set port base address (must match hardware)
90 ' Write the value of divisor into
100 ' Output the data through port1.
110 ' Wait until the transmitter buffer
120 ' Loop over data (0-255) and send it
130 ' Print out the data byte received
140 ' If the value sent <> the received value then
150 ' See if the data is available by checking
160 ' See if the transmitter buffer is empty.
170 ' Check if the CTS, RTS and DSR signals. (See page 7 for information on the format and programming
detailed information.)
180 'Set proper parameters
190 ' Read all registers once to
200 'Both cards
210 'Clear screen
220 'Clear the screen
230 'Input data from COM2 to COM1
240 'Do the same thing for port2.
250 'Power-up
260 'Program
270 'Receive data from COM1 to COM2
280 'Send data from COM2 to COM1
290 '****** Transmit data sub-routine ******
300 '****** Receive data sub-routine ******
310 'If everything is OK, then stop.
320 'If everything is OK, then stop.
330 'Clear the screen
340 'Clear the screen
350 'Clear the screen
360 'Clear the screen
370 'Clear the screen
380 'Clear the screen
390 'Clear the screen
400 'Clear the screen
410 'Clear the screen
420 'Clear the screen
430 'Clear the screen
440 'Clear the screen
450 'Clear the screen
460 'Clear the screen
470 'Clear the screen
480 'Clear the screen
490 'Clear the screen
500 'Clear the screen
510 'Clear the screen
520 'Clear the screen
530 'Clear the screen
540 'Clear the screen
550 'Clear the screen
560 'Clear the screen
570 'Clear the screen
580 'Clear the screen
590 'Clear the screen
600 'Clear the screen
610 'Clear the screen
620 'Clear the screen
630 'Clear the screen
640 'Clear the screen
650 'Clear the screen
660 'Clear the screen
670 'Clear the screen
680 'Clear the screen
690 'Clear the screen
700 'Clear the screen
710 'Clear the screen
720 'Clear the screen
730 'Clear the screen
740 'Clear the screen
750 'Clear the screen
760 'Clear the screen
770 'Clear the screen
780 'Clear the screen
790 'Clear the screen
800 'Clear the screen
810 'Clear the screen
820 'Clear the screen
830 'Clear the screen
840 'Clear the screen
850 'Clear the screen
860 'Clear the screen
870 'Clear the screen
880 'Clear the screen
890 'Clear the screen
900 'Clear the screen
910 'Clear the screen
920 'Clear the screen
930 'Clear the screen
940 'Clear the screen
950 'Clear the screen
960 'Clear the screen
970 'Clear the screen
980 'Clear the screen
990 'Clear the screen
```

**Program for first computer**

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.

```basic
10 '************ STEP 1: INITIALIZATION ************
20 'Clear screen
30 'CLS
40 'Define variables A to Z as integer
50 'DEFINT A-Z
60 'Set port base address (must match hardware)
70 'PORT = &H3F8
80 'Set baud rate to 300
90 'OUT PORT + 3, &H80
100 'OUT PORT, &H80
110 'OUT PORT, 1
120 'OUT PORT + 3, &H1F
130 '****** STEP 2: SEND DATA ********
140 'FOR I = 65 TO 90
150 ' FOR I = 65 TO 90
160 'END
170 'GOSUB 200
180 'NEXT I
190 'GOTO 260
```

**Program for second computer**

```basic
10 '************ STEP 1: INITIALIZATION ************
20 'Clear screen
30 'CLS
40 'Define variables A to Z as integer
50 'DEFINT A-Z
60 'Set port base address (must match hardware)
70 'PORT = &H3F8
80 'Set baud rate to 300
90 'OUT PORT + 3, &H80
100 'OUT PORT, &H80
110 'OUT PORT, 1
120 'OUT PORT + 3, &H1F
130 '****** STEP 2: SEND DATA ********
140 'FOR I = 65 TO 90
150 ' FOR I = 65 TO 90
160 'END
170 'GOSUB 200
180 'NEXT I
190 'GOTO 260
```
char flag; /* Flag for end of output/input data */
int timeout; /* Timeout counter */
outport((base0+2), 0xc9);  /* enable port 0 FIFO */
outport((base1+2), 0xc9);  /* enable port 1 FIFO */
/* Set communication parameters for port 0 */
outp(base0+3, 0x80);  /* Set DLAB=1 */
/* Set baud = 115200 */
outp(base0, 0x01);
outp(base1+1, 0);
/* Set data=8, stop=1, no parity */
outp(base0+3, 0x03);
/* Disable port 0 interrupt */
outp(base0+1, 0x00);

while (cmd[0] != 'q' && cmd[0] != 'Q')
{
    i=0;
    cmd[strlen(cmd)] = 0x0d;
    flag=1;
    while (flag)
    {
        outportb(base0, cmd[i]);   /* Send data */
        if (cmd[i] == 0x0d)
            flag=0;
        i++;
    }

    i=0;
    flag=1;
    timeout=TIME_OUT;
    while (flag)
    {
        /* Check if receiver data is ready */
        if ((inportb(base+5) & 1) !=0)
        {
            rec[i]=inportb(base);  /* Receive data */
            if (rec[i] == 0x0d)
            {
                rec[i+1]="\0";
            }
        }

        printf("\nEnter a string to be transmitted "
        "(15 characters or less) or Q to quit:"
        );
        gets(cmd);
    }
}

C language test program
You can use the following C program to test the PCM-3640’s send and receive functions.

C language test program
You can use the following C program to test the PCM-3640’s send and receive functions.

/                                            
/  Program:  DEMO01.C                          
/  Description: This demo program transmits a string    
/  to COM1 and receives a string from COM2    
/  Compiler: Turbo C 2.0                        
/                                            
*******************************************************************************/
#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 base1 = 0x2f8;  /* Base address of port 1 */
static char rec[16]; /* Buffer for received string */
static char cmd[16]; /* Buffer for transmitted string */

void main()
{
    int i; /* Counter for character being sent/received */
flag=0;
printf("Received data: %s\n", rec);
}
i++;
}
else
{ /* Check timeout */
timeout--;
if (timeout == 0)
{
flag = 0;
printf("Timeout error\n");
}
}
printf("Enter a string to be transmitted 
(15 characters or less) or Q to quit: ");
gets(cmd);

Register structure and format

This section gives short description of each of the module’s registers. For more information please refer to the data book for the STARTECH 16C550 UART chip.

All registers are one byte. Bit 0 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.

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

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

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

The two bytes BASE+0 and BASE+1 together form a 16-bit number, the divisor, which determines the baud rate. Set the divisor as follows:

<table>
<thead>
<tr>
<th>Baud rate</th>
<th>Divisor</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2304</td>
</tr>
<tr>
<td>75</td>
<td>1536</td>
</tr>
<tr>
<td>110</td>
<td>1047</td>
</tr>
<tr>
<td>133.5</td>
<td>857</td>
</tr>
<tr>
<td>150</td>
<td>768</td>
</tr>
<tr>
<td>300</td>
<td>384</td>
</tr>
<tr>
<td>600</td>
<td>192</td>
</tr>
<tr>
<td>1200</td>
<td>96</td>
</tr>
<tr>
<td>1800</td>
<td>64</td>
</tr>
<tr>
<td>2000</td>
<td>58</td>
</tr>
<tr>
<td>2400</td>
<td>48</td>
</tr>
<tr>
<td>3600</td>
<td>32</td>
</tr>
<tr>
<td>4800</td>
<td>24</td>
</tr>
<tr>
<td>7200</td>
<td>16</td>
</tr>
<tr>
<td>9600</td>
<td>12</td>
</tr>
<tr>
<td>19200</td>
<td>6</td>
</tr>
<tr>
<td>38400</td>
<td>3</td>
</tr>
<tr>
<td>56000</td>
<td>2</td>
</tr>
</tbody>
</table>

BASE+1

Interrupt Status Register (ISR) when DLAB=0

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>FIFO trigger level</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>01</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>04</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>08</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>

BASE+2

FIFO Control Register (FCR)

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 1</th>
<th>Clear contents of receive FIFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Clear contents of transmit FIFO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Change RXRDY and TXRDY from mode 0 to mode 1</td>
</tr>
</tbody>
</table>

BASE+3

<table>
<thead>
<tr>
<th>Bit 6-7</th>
<th>Set trigger level for receiver FIFO interrupt</th>
</tr>
</thead>
</table>

BASE+4

Line Control Register (LCR)

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Word length select bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
</tr>
</tbody>
</table>

BASE+5

Modem Control Register (MCR)

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 1</th>
<th>DTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>DTR</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>RTS</td>
</tr>
</tbody>
</table>

BASE+6

Line Status Register (LSR)

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Receiver data ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Overflow error</td>
</tr>
<tr>
<td>1</td>
<td>Parity error</td>
</tr>
<tr>
<td>2</td>
<td>Framing error</td>
</tr>
<tr>
<td>3</td>
<td>Break interrupt</td>
</tr>
<tr>
<td>4</td>
<td>Transmitter holding register empty</td>
</tr>
<tr>
<td>5</td>
<td>Transmitter shift register empty</td>
</tr>
<tr>
<td>6</td>
<td>At least one parity error, framing error or break indication in the</td>
</tr>
</tbody>
</table>

PCM-3640 User's Manual 7
### PC/104 Bus signal assignments

<table>
<thead>
<tr>
<th>Pin</th>
<th>J1/P1 Row A</th>
<th>J1/P1 Row B</th>
<th>J2/P2 Row C</th>
<th>J2/P2 Row D</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>--</td>
<td>--</td>
<td>0V</td>
<td>0V</td>
</tr>
<tr>
<td>1</td>
<td>IOCHCK*0V</td>
<td>SBHE*</td>
<td>MEMCS16*</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SD7</td>
<td>RESETDRV</td>
<td>LA23</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SD6</td>
<td>+5V</td>
<td>LA22</td>
<td>IRQ10</td>
</tr>
<tr>
<td>4</td>
<td>SD5</td>
<td>IRQ9</td>
<td>LA21</td>
<td>IRQ11</td>
</tr>
<tr>
<td>5</td>
<td>SD4</td>
<td>-5V</td>
<td>LA20</td>
<td>IRQ12</td>
</tr>
<tr>
<td>6</td>
<td>SD3</td>
<td>DRQ2</td>
<td>LA19</td>
<td>IRQ15</td>
</tr>
<tr>
<td>7</td>
<td>SD2</td>
<td>-12V</td>
<td>LA18</td>
<td>IRQ14</td>
</tr>
<tr>
<td>8</td>
<td>SD1</td>
<td>ENDXFR*</td>
<td>LA17*</td>
<td>DACK0*</td>
</tr>
<tr>
<td>9</td>
<td>SD0</td>
<td>+12V</td>
<td>MEMR*</td>
<td>DRQ0*</td>
</tr>
<tr>
<td>10</td>
<td>IOCHRDY (KEY)²</td>
<td>MEMW*</td>
<td>DACK5*</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>AEN</td>
<td>SMEMW*</td>
<td>SD8</td>
<td>DRQ5</td>
</tr>
<tr>
<td>12</td>
<td>SA19</td>
<td>SMEMR*</td>
<td>SD9</td>
<td>DACK6*</td>
</tr>
<tr>
<td>13</td>
<td>SA18</td>
<td>IOW*</td>
<td>SD10</td>
<td>DRQ6</td>
</tr>
<tr>
<td>14</td>
<td>SA17</td>
<td>IOR*</td>
<td>SD11</td>
<td>DACK7*</td>
</tr>
<tr>
<td>15</td>
<td>SA16</td>
<td>DACK3*</td>
<td>SD12</td>
<td>DRQ7</td>
</tr>
<tr>
<td>16</td>
<td>SA15</td>
<td>DRQ3</td>
<td>SD13</td>
<td>+5V</td>
</tr>
<tr>
<td>17</td>
<td>SA14</td>
<td>DACK1*</td>
<td>SD14</td>
<td>MASTER*</td>
</tr>
<tr>
<td>18</td>
<td>SA13</td>
<td>DRQ1</td>
<td>SD15</td>
<td>0V</td>
</tr>
<tr>
<td>19</td>
<td>SA12</td>
<td>REFRESH*</td>
<td>SD12</td>
<td>(KEY)²</td>
</tr>
<tr>
<td>20</td>
<td>SA11</td>
<td>SYSCLK</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>21</td>
<td>SA10</td>
<td>IRQ7</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>22</td>
<td>SA9</td>
<td>IRQ6</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>23</td>
<td>SA8</td>
<td>IRQ5</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>24</td>
<td>SA7</td>
<td>IRQ4</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>25</td>
<td>SA6</td>
<td>IRQ3</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>26</td>
<td>SA5</td>
<td>DACK2*</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>27</td>
<td>SA4</td>
<td>TC</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>28</td>
<td>SA3</td>
<td>BALE</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>29</td>
<td>SA2</td>
<td>+5V</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>30</td>
<td>SA1</td>
<td>OSC</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>31</td>
<td>SA0</td>
<td>0V</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>32</td>
<td>0V</td>
<td>0V</td>
<td>--</td>
<td>--</td>
</tr>
</tbody>
</table>

### FIFO

- BASE+6: Modern 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.

<table>
<thead>
<tr>
<th>I/O address (hex)</th>
<th>Assignment</th>
</tr>
</thead>
<tbody>
<tr>
<td>000-1FF</td>
<td>used by base system board</td>
</tr>
<tr>
<td>200</td>
<td>not used</td>
</tr>
<tr>
<td>201</td>
<td>game control</td>
</tr>
<tr>
<td>280-2FF</td>
<td>not used</td>
</tr>
<tr>
<td>2F8-2FF</td>
<td>COM2</td>
</tr>
<tr>
<td>300-377</td>
<td>not used</td>
</tr>
<tr>
<td>378-37F</td>
<td>printer port</td>
</tr>
<tr>
<td>380-3AF</td>
<td>not used</td>
</tr>
<tr>
<td>3B0-3BF</td>
<td>monochrome adapter and printer</td>
</tr>
<tr>
<td>3C0-3CF</td>
<td>not used</td>
</tr>
<tr>
<td>3D0-3DF</td>
<td>color and graphics adapters</td>
</tr>
<tr>
<td>3F0-3F7</td>
<td>floppy diskette drive</td>
</tr>
<tr>
<td>3F8-3FF</td>
<td>COM1:</td>
</tr>
</tbody>
</table>