User manual for

PC-74

High Performance Analog I/O Boards for IBM PC, PC XT, PC AT, PS/2 Model 30 and compatible Computer Systems.

Second edition.

January 1990

March 1990 Printing

Table of Contents

Preface

This manual is written for users of the PC-74[1] series of analog I/O boards. It provides all information necessary to successfully program and operate all boards in the series, as well as all information required to use the supplied driver software in conjunction with several languages.

This manual assumes :

- That you have a basic knowledge of electronic circuitry and measurement techniques.
- That you are familiar with the host PC which you are using.
- That you are capable of writing your own programs.

The manual contains the following sections.

Chapter 1 - Introduction.

- Chapter 1 contains an overview of the PC-74 series of boards, their capabilities and the capabilities of the supplied software.

Chapter 2 - Architecture.

- Chapter 2 discusses the basic operation of the PC-74 board.

Chapter 3 - Configuration.

- Chapter 3 discusses the selection of various board parameters and the configuration of the board for various operating requirements.

Chapter 4 - Interconnection.

- Chapter 4 describes the connection of the PC-74 series of boards to the host computer and to user inputs.

1 For the rest of this manual, "PC-74" will be used to refer to all of the boards in the series. Where information is specific to a particular board, this will be stated in the text.

Chapter 5 - Programming the PC-74.

- Chapter 5 describes the register structure of the PC-74 series of boards and provides the information required to program these registers.

Chapter 6 - Using the PC-74 driver software.

- Chapter 6 describes the use of the supplied software drivers.

Chapter 7 - Calibration.

- Chapter 7 describes the procedures and equipment required to calibrate the PC-74 series of boards. Calibration software included with the PC-74 is also described.

Appendix A - Hardware Specifications.

- Appendix A provides complete electrical specifications for the PC-74 series of boards.

Appendix B - Software Specifications.

- Appendix B contains complete technical specifications for the various memory models used by the PC-74 driver software.

Appendix C - PC-74 Component Layout.

- Appendix C contains a layout diagram of the PC74. This allows jumpers and trimpots to be easily located.

Appendix D - Problem Determination guide.

- Appendix D contains information which may help you if you are experiencing problems with your PC-74.

Appendix E - PC-77 Screw Terminal Panel.

- Appendix E contains the user manual for the PC-77 screw terminal board. If you are using a PC-77, you should read this appendix.

Appendix F - Differences from previous driver versions.

- Appendix F discusses the diffences between version 1.00 and 1.01 of the driver software. If you are upgrading from version 1.00, you should read this appendix.

Chapter 1

Introduction

1.1. Overview

The PC-74 series of boards are half size, low cost high accuracy analog and digital I/O boards for the IBM PC/XT/AT, PS/2 model 30 and compatible series of computers. They are plug-compatible with the DT2811 series of I/O boards. The family consists of four boards :

- PC-74LA. Gain programmable to 1, 10, 100 or 500, 30KHz throughput, plug compatible with DT2811-PGL.
- PC-74LC. Gain programmable to 1, 10, 100 or 500, 80KHz throughput (single channel, multichannel throughput 30KHz), plug compatible with DT2811-PGL.
- PC-74HA. Gain programmable to 1, 2, 4 or 8, 30KHz throughput, plug compatible with DT2811-PGH.
- PC-74HC. Gain programmable to 1, 2, 4 or 8, 80KHz throughput (single channel: multichannel throughput 30KHz), plug compatible with DT2811-PGH.

1.2. Features

These boards can be plugged into any of the fully bussed slots in the PC/XT/AT or PS/2 model 30 computer backplane.

1.2.1. A/D subsystem

The A/D subsystem's major component is a monolithic analog to digital converter, which accepts analog voltage inputs from sensors, such as pressure transducers and thermocouples, and converts them into 12 bit digital codes.

This code is transmitted to the host processor, which processes it according to the software in use at the time. The A/D gain may be set by software to 1, 2, 4 or 8 for the HA and HC boards (for high level inputs), or be set by software to 1, 10, 100 or 500 for the LA and LC boards (for low level inputs).

The A/D section allows for jumper selection of either 16 single-ended or 8 differential inputs, and can be configured for unipolar (input range 0-5V) or bipolar (input ranges (+/-5V and +/-2.5V) operation. Resolution is 12 bits. For unipolar outputs, the output code is straight binary, and for bipolar, offset binary.

The A/D may be operated in either single conversion or continuous conversion mode. In single conversion mode the board performs a single conversion on the selected input channel and stops on completion of this conversion. In continuous conversion mode conversions are performed at a set rate. This rate is set by programming the PC-74's internal timer.

A/D conversions may be monitored by either polled I/O or by interrupts. In polled I/O mode the software continuously polls the board's status register to check for completion of the current A/D conversion. In interrupt mode, the board automatically generates a hardware interrupt on completion of each conversion.

Key specifications

- A/D resolution : 12 Bits
- Nonlinearity : Less than +/-0.75 LSB
- A/D full scale input ranges : Jumper selectable between unipolar (full scale range from 0V to +5V) or bipolar (+/-5V or +/-2.5V range).
- Number of A/D inputs : Jumper selectable 16 single ended or 8 differential.
- A/D throughput rate : 30KHz (80KHz on single channel with LC and HC models).

1.2.2. D/A Subsystem

The D/A subsystem consists of two 12-bit D/A converters, DAC0 and DAC1. Digital outputs are received by the host processor and converted to an analog voltage output required by the application in hand. The two DACs are independent of one another, and can both operate at a throughput of up to 50KHz. Output ranges are independently configurable as 0-+5V unipolar, or two bipolar ranges: +/-5v and +/-2.5V.

Key specifications

- D/A resolution : 12 Bits
- D/A nonlinearity : Within 0.01% FSR
- Full scale output ranges : 0 to +5V, -5V to +5V or -2.5V to +2.5V.
- D/A throughput rate : 50 KHz.

1.2.3. Digital I/O subsystem

The digital I/O subsystem is an interface for the transfer of digital data from and to the PC bus to and from one or more peripheral devices connected to the PC-74. There are two digital ports, ports 0 and 1. Port 0 is the input port, and port 1 is the output port.

1.2.4. Interface logic

The PC-74 is accessed via I/O operations performed by the host processor. Of the 10 bit address received by the board, the most significant 7 bits select the board, and the least significant 3 bits select the register to be accessed.

The PC-74 occupies 8 byte locations: five byte locations for the A/D subsystem, two for the D/A subsystem, and one for the digital I/O subsystem. The base address of the board can be selected to be located anywhere between 200 (hex) and 7F8 (hex).

The PC-74 operates from the +5V, +12V and -12V lines of the PC bus.

1.3. Software support

Supplied with the PC-74 board are a set of real time device drivers for use with a wide variety of software. These device drivers are written in C and assembler, and are callable from the following languages :

- Microsoft C (V5.1).
- Microsoft FORTRAN (V5).
- Microsoft QuickBasic (V4.5)
- Turbo C (V1.5)
- Turbo Pascal (V5)

1.4. Throughput

The throughput of the PC-74 series of boards is dependant on several factors, principally whether the a single channel of data or many channels are being converted, and whether DMA or program transfer techniques are used to read data from the A/D converter.

1.4.1. DMA

DMA is Direct Memory Access and, as the name implies, data from the A/D is transferred to the PC's memory directly, without the data acquisition software in use taking any action (other than setting the hardware up initially, and waiting for the DMA to complete). In this case, the processing power of the host PC is of no consequence, and the throughput of the PC-74 will be at its maximum.

1.4.2. Program transfer.

If program transfer techniques are used (Polled I/O or interrupts), the situation becomes more complex. In this case the maximum possible throughput is limited by

the processing power of the CPU in the host PC, and the efficiency of the software in use. In general, throughput of greater than 20 Khz is very seldom achieved.

Note that the software drivers supplied with the PC-74, although efficient, are written as general purpose, 'idiot proof' routines. Custom assembly language routines can easily be written which will outperform these.

1.4.3. Single channel conversions.

If only a single channel of data is to be converted, then DMA can be used. In this case the throughput of the the HC and LC boards is always 80 KHz, and the HA and LA boards always 30 KHz.

1.4.4. Multi-channel Conversions.

Multi-channel conversions must always be done by program transfer, and are hence limited by the speed of the host PC.

Throughput also is dependant on the gain at which each channel to be converted is operating. Throughput for each possible gain setting is as follows:

Gain	Throughput
1, 2, 4, 8 and 10	30 KHz (HC and LC), 20 KHz (HA and LA)
100 and 500	2.5 KHz

1.5. Getting Started

If you want to get started quickly and have notchanged any of the factory installed jumpers on the PC-74, here's what to do :

i. Install the PC-74 in your computer. (Chapter 3 provides brief instructions on this, but if you are not sure, it is better to get someone who is qualified to do this.)

ii. Remove any other 'exotic' boards. (Other engineering and scientific boards, prototype boards, tape back-up boards etc.) The PC-74 can be configured to avoid any conflicts with such boards, but this procedure assumes that the PC-74 is configured as it came from the factory.

iii. Connect up a voltage source to any (or all) of the input channels. (You can also loop the analog outputs back to the inputs). The pin-out of the PC-74 connector is shown in figure 4.1 later in the manual.

iv. Run the DEMO1.EXE program on the driver disk. This should print out the following message :

```
PC-74 Driver Version 1.01
PC-74 diagnostics report the following :

PC-74 found, operating correctly.
```

- If the program does not do this, the PC-74 may be conflicting with another board, or the jumpers on the PC-74 have been changed. You will have to configure the PC-74 fully, as described in chapter 3.

v. If the PC-74 was found, run the program DEMO2.EXE on the distribution disk. This displays the voltage on all the PC74 input channels, and allows you to vary the analog outputs. This program assumes that the PC-74 is in its factory standard configuration.

1.6. Accessories

In order to assist in applying the PC-74, several accessories are available. Only a brief description is given here. Consult your dealer for full details.

1.6.1. PC-77

The PC-77 is a screw terminal board, specifically designed for the PC74. It provides direct access to all PC-74 I/O lines. Connection to the PC-74 is made via ribbon cable. The PC-77 is fully mechanically and electrically compatible with the DT707. In addition, the PC-77 provides cold-junction compensation with up-range open input detection for thermocouples, as well operation in conjunction with 4 - 20 mA inputs.

1.6.2. PC-81

The PC-81 is a input expander board. Multiple PC-81s may be used to expand the input channel capability of the PC-74 to more than 65000 channels. Each PC-81 has 64 screw terminal inputs.

1.6.3. PC-22

The PC-22 is a Euro-card format single channel signal conditioning module. It provides programmable gain and filtering functions.

1.6.4. PC-68

The PC-68 is a Euro-card format four channel strain gage signal conditioning board. It provides four independent channels with user programmable excitation, and high performance instrumentation amplifier. The PC-68 can also be used simply as a four channel ultra-high performance instrumentation amplifier board.

[This page intentionally left blank.]

Chapter 2

Architecture

This chapter describes the architecture of the PC-74 series of boards. The block diagram in figure 2.1 highlights the major elements contained on the board, and their interrelationship. There are five major subsections. These are the following :

2.1. D/A Subsystem

The D/A subsystem contains a two D/A converters and their associated circuitry, including buffer registers. When a the high byte of a D/A converter is written, the data in the buffer register is transferred to the D/A, and hence to the analog output.

2.2. A/D Subsystem

The A/D subsystem contains several separate components :

- The input multiplexer. The multiplexer selects either one of eight differential input channels or one of sixteen single ended input channels. This channel is selected by a channel address, which is written to a channel address register.
- The instrumentation amplifier. The instrumentation amplifier amplifies either the difference between two differential inputs, or the difference between a single ended input and ground. The degree of amplification can be selected under software control.
- The sample and hold unit. The sample and hold unit holds the output of the instrumentation amplifier steady for the duration of the A/D converter's conversion process.
- The A/D converter performs the actual A/D conversion. An A/D conversion is begun by a A/D strobe. This is generated by the timing and control section, described later.

Data may be transferred from A/D either by polled I/O, DMA or interrupts.

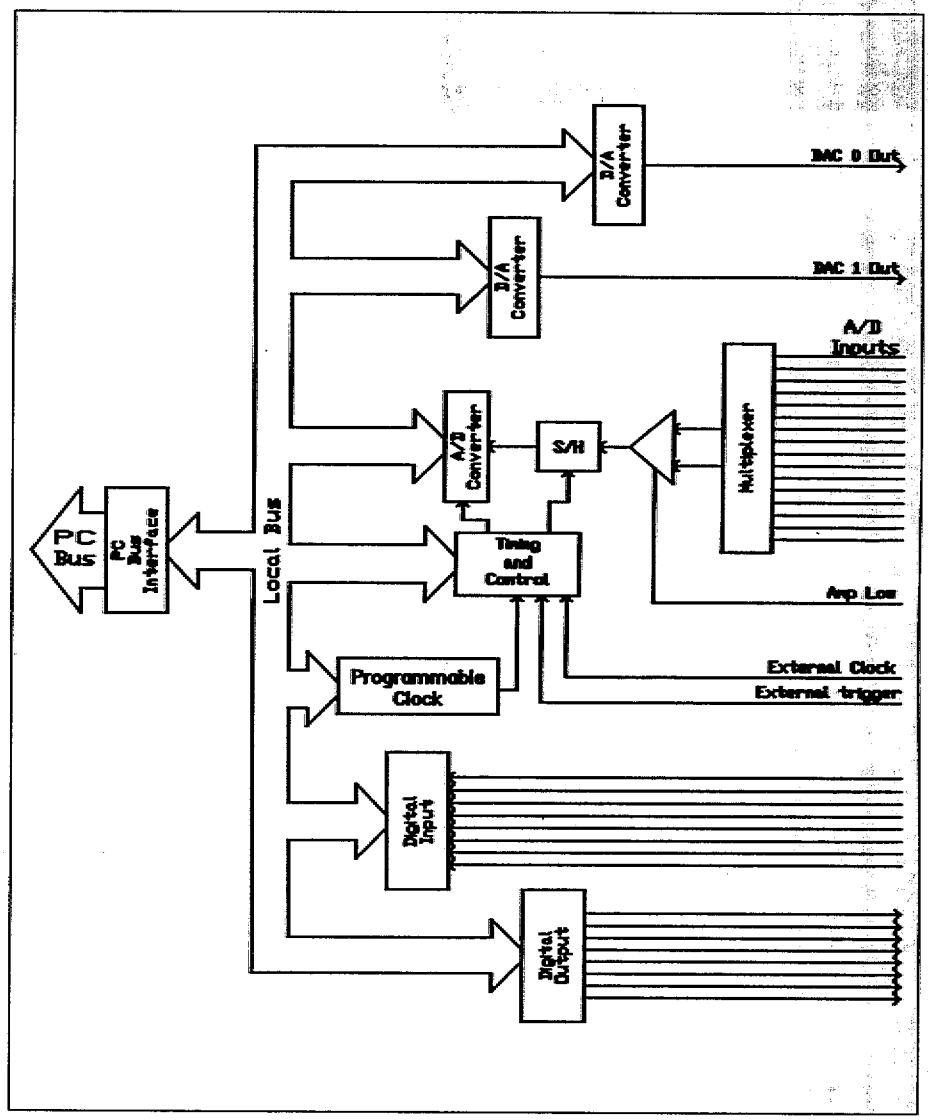


Fig 2-1.
PC-74 block diagram.

2.2.1. Polled I/O.

Polled I/O is the simplest possible method of data transfer. It proceeds as follows :

i. The A/D converter is set the appropriate channels and gain, and a conversion started.

ii. The program the continuously monitors the status register, until the A/D conversion completes.

iii. The data from the A/D conversion is then read, and stored in the PC's memory by the program.

iv. If block of data is to read (the board is set for continuous conversion), then the program waits for the next conversion to complete. This process repeats until however many samples are required have been read.

Polled I/O has the advantage of extreme simplicity, but has two disadvantages.

- Transfer speed is limited by the speed of the CPU in the host PC.
- While polled I/O is being done, the CPU is totally dedicated to this process, and cannot deal with anything else (such as keyboard input).

Polled I/O is generally used for single conversions, or continuous conversions at low sampling rates (less than 30 KHz)

2.2.2. Interrupts.

Interrupt based data transfer proceeds as follows :

i. The PC hardware is first set up with the address of an interrupt service routine (ISR). This ISR is a simple sequence of instructions for reading data from the A/D. The A/D converter is then set for the appropriate channels and gain, and a conversion started. The program can then continue with any other task, such as checking the keyboard for input.

ii. When the A/D converter completes its conversion, an interrupt is generated.

iii. This interrupt causes the CPU to stop what it is doing, and execute the interrupt service routine. This reads the data from the A/D, and stores it in the PC's memory.

iv. This process repeats until however many samples are required have been read.

Interrupt based I/O has the advantage that the CPU can perform other tasks while the A/D conversion is in progress, but has certain disadvantages of its own.

- As in the case of polled I/O, transfer speed is limited by the speed of the CPU in the host PC. Maximum transfer rates are generally considerable less than for polled I/O.
- Interrupt service routines are complex, and subject to several problem not normally encountered. If you wish to use interrupts, we recommend that you use the driver software supplied with the PC-74.

Interrupt driven I/O is normally used for low (less then 3 KHz) sample rates.

2.2.3. DMA

DMA stands for Direct Memory Access. It proceeds as follows :

i. The PC hardware is first set up with the address of the memory into which the A/D samples are to be put, and the number of samples to be obtained. The A/D converter is then set for the appropriate channels and gain, and a conversion started. The program can then continue with any other task, such as checking the keyboard for input.

ii. When the A/D converter completes its conversion, a DMA cycle is initiated.

iii. This DMA cycle reads the data from the A/D, and stores it in the PC's memory, without the CPU taking any action.

iv. This process repeats until however many samples are required have been read.

v. The program checks the PC's hardware to see if all the required samples have been transferred. Note that this checks can be done at any time, unlike for polled I/O, where the A/D status must be checked quickly enough to ensure that data is read before the next A/D conversion completes.

The primary advantage of DMA operation is the very high transfer rate, but DMA can can only operate on a single channel.

2.3. Bus interface.

The bus interface is responsible for two functions :

i. The decoding of the board's base address. The board's base address is set by jumpers.

ii. The generation of interrupts. Interrupts can be generated under two conditions :

ii.i. A/D interrupts are enabled, DMA is not enabled, and an A/D conversion cycle completes.

ii.ii. A/D interrupts are enabled and an A/D error occurs.

A/D errors may occur as a result of two conditions :

- Completion of an A/D conversion prior to the data from the previous conversion being read.
- An attempt to start an an A/D conversion prior to the previous conversion completing.

If both DMA and interrupts are enabled, interrupts will be generated on A/D errors. As an A/D error routinely occurs on DMA termination, this can be used as a method of signaling the end of a DMA operation.

2.4. Timing and control

The timing and control subsection is responsible for the generation of A/D strobes. It is A/D strobes which cause the A/D converter to begin a conversion. A simplified block diagram of this section is shown in figure 2.2. There are five major subsections :

2.4.1. A/D strobe multiplexer.

This multiplexer selects between continuous and single conversions. If single conversions are selected, then the strobe to the A/D converter is generated by a write operation to the gain/channel register. In this case every write operation to this register initiates a conversion. If continuous conversions are selected, then the strobes to the A/D converter are generated by the clock divider, described below.

2.4.2. Clock divider.

The clock divider divides the clock signal from the clock selection multiplexer by a programmable ratio. No strobes are generated unless the divider is enabled, so allowing the start of a set continuous conversion to synchronized either to external events, or program execution. Note that this divider is active regardless of whether the clock signal is internally or externally generated.

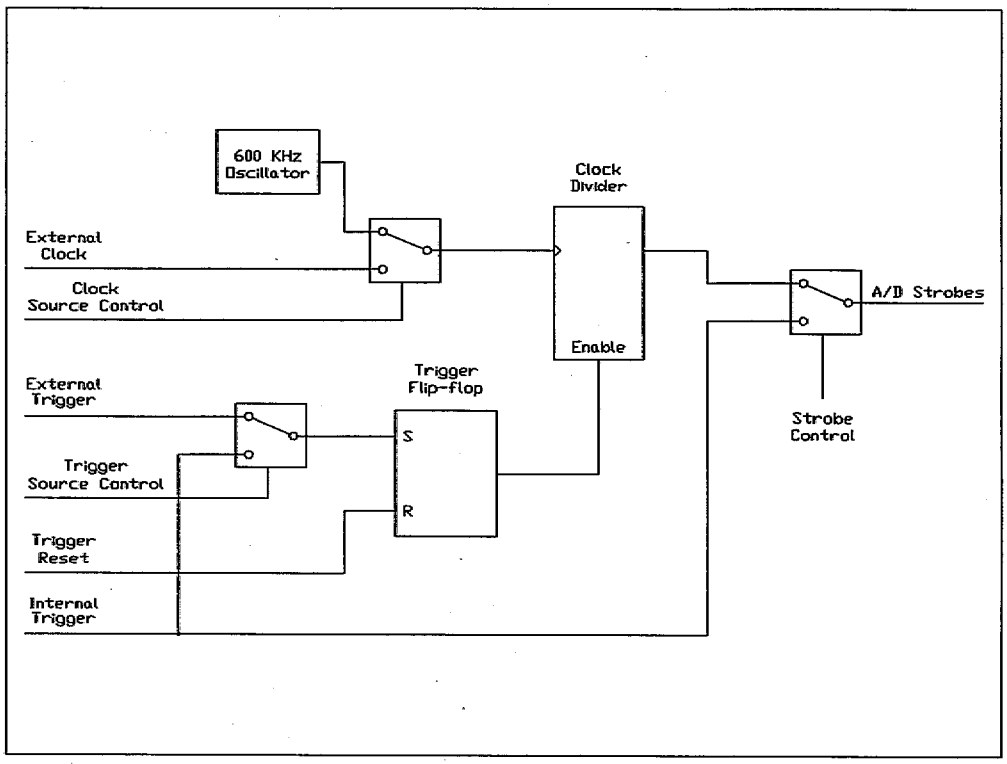


Fig. 2.2. Timing and control section.

2.4.3. Trigger flip-flop.

The trigger flip-flop is used to enable the clock divider. If the flip-flop is not set, then no clock signal will be generated by the divider. This flip-flop is et by a trigger signal from the trigger selection multiplexer.

2.4.4. Clock selection multiplexer.

The clock selection multiplexer selects between internal and external clocks.

- The external clock is obtained from the J1 connector.
- The internal clock is crystal controlled, and operates at 600 KHz.

2.4.5. Trigger selection multiplexer.

The trigger selection multiplexer selects between internal and external triggers. An internal trigger is caused by a write operation to the gain/channel register, and an external trigger by a negative edge on the trigger line of the J1 connector.

2.5. Modes of operation.

The multiplexers described above, can configure the clock and control section into four modes. These are the following :

2.5.1. Mode 0 - Single conversion.

In this mode, an A/D strobe is generated on each write operation to the Gain/channel register in the PC-74. The sequence of operations for this mode is as follows :

i. The A/D control/status register is loaded with mode 0.

ii. The gain/status register is then written with the number of the first channel and the gain for that channel. This generates an A/D strobe, hence starting the A/D converter.

iii. The resulting A/D data is then read in, by either polled I/O or interrupts.

iv. The next A/D conversion may then be started by writing the number of the next channel and its gain to the gain/channel register.

2.5.2. Mode 1 - Continuous conversion, internal trigger and clock.

In this mode, A/D strobes are generated by the internal clock, once the trigger flip-flop has been set. The sequence of events is as follows :

i. The A/D control/status register is loaded with mode 1. This resets the trigger flip-flip, disabling A/D strobes. The timer/counter register is loaded with the division ratio which will give the required sample rate.

ii. The gain/status register is then written with the number of the first channel and the gain for that channel. This sets the trigger flip, enabling the divider. A/D strobes are then generated at the rate set by the internal clock (600 KHz), and the code written to the divider.

iii. On each occasion that the A/D completes a conversion, the resulting A/D data is then read in, by polled I/O, DMA or interrupts.

iv. A/D strobes will continue to be generated until a new mode value is written to the control/status register.

2.5.3. Mode 2 - Continuous conversion, external trigger.

This mode of operation is very similar to mode 1, with the exception that the trigger flip-flop must be set by negative edge on the external trigger input of the PC-74. The sequence of events is as follows :

- The A/D control/status register is loaded with mode 2. This resets the trigger flip-flip, disabling A/D strobes. The timer/counter register is loaded with the division ratio which will give the required sample rate.
- The gain/status register is then written with the number of the first channel and the gain for that channel.

- Conversions are initiated by a negative edge on the external trigger input. This sets the trigger flip, enabling the divider. A/D strobes are then generated at the rate set by the internal clock (600 KHz), and the code written to the divider.
- On each occasion that the A/D completes a conversion, the resulting A/D data is then read in, by polled I/O, DMA or interrupts.
- A/D strobes will continue to be generated until a new mode value is written to the control/status register.

2.5.4. Mode 3 - Continuous conversion, external trigger, clock.

This mode of operation is very similar to mode 2, with the exception that the rate a which A/D strobes is generated is set by the frequency on the external oscillator input of the PC-74 and the division ratio of the timer/counter. The sequence of events is as follows :

i. The A/D control/status register is loaded with mode 2. This resets the trigger flip-flip, disabling A/D strobes. The timer/counter register is loaded with the division ratio which will give the required sample rate.

ii. The gain/status register is then written with the number of the first channel and the gain for that channel.

iii. Conversions are initiated by a negative edge on the external trigger input. This sets the trigger flip, enabling the divider. A/D strobes are then generated at the rate set by the external clock, and the code written to the divider.

iv. On each occasion that the A/D completes a conversion, the resulting A/D data is then read in, by polled I/O, DMA or interrupts.

v. A/D strobes will continue to be generated until a new mode value is written to the control/status register.

2.6. Digital I/O.

The digital I/O section of the PC-74 consists of two ports. port 0 is the input port and port 1 the output port. Both ports are 8 bit, and are ALS TTL compatible.

[This page intentionally left blank.]

Chapter 3

Configuring the PC-74 board

3.1. Introduction

The PC-74 board can be configured in many different ways to suit each user's individual requirements. This configuration is set by the position of the various mini-jumps on the board. Each set of mini-jumps controls a specific aspect of the operation of the board. These are as follows :

i. DT2811 Compatibility. For specific applications, it is possible to totally disable all PC-74 extensions to the DT2811 standard. This is very seldom required. In this mode DMA operation is not available, and the DMA enable bit in the control/status register cannot be written. This jumper is factory preset to allow PC-74 extensions.

ii. Bus interface. The base address of the board, the DMA level and the interrupt level used by the board can be set. As supplied by the factory, the base address is set to 300H, the DMA level to 1, and the interrupt level to 2. This allows operation in a standard PC/XT/AT which contains only conventional boards (Multifunction boards, disk controller boards, display boards etc), but may require modification if exotic boards (other scientific boards, certain backup systems etc) are installed. Note that both the DMA request and the interrupt line are electronically disconnected from the PC bus unless specifically enabled by software, even if an interrupt level selection jumper or a DMA level selection jumper is installed.

iii. D/A operation. The output range of both D/A may be selected independently by mini-jumps. As supplied by the factory, both D/A outputs are set for bipolar output, and the range from -5 to +5 volts.

iv. A/D operation. Two aspects of A/D operation may be configured. These are as follows :

iv.i. The input range of the A/D may be set, by jumper, for uni- or bipolar operation, for input voltage ranges 0 - +5V, +/-2.5V or +/-5V.

iv.ii. The A/D may be set for either differential, single ended or pseudo-differential operation. As supplied by the factory, the A/D is set for single-ended mode operation. The above set of factory installed jumpers are designed to suit most applications, and are required for correct operation of the demonstration, diagnostics and calibration software supplied with the board.

3.2. Changing the jumper settings

The jumpers may be located either from the diagram in appendix C, or from the labels on the PC-74 board itself.

In order to change the jumper settings follow the procedure below :

i. Switch off the computer.

ii. Remove the board.

iii. Change the required jumpers.

iv. Replace the board in the PC.

v. Power up, and run a program (such as DEMO1), which executes the PC-74 diagnostics routines.

3.3. DT2811 Compatibility jumper setting.

The PC-74 allows bit 3 of the control status register to be written. On the DT2811, this bit is reserved, and cannot be written. The compatibility jumper disables writes to the this bit, and hence gives total DT2811 compatibility. This is in fact very seldom required (to date, we are aware of no software, including the software supplied with the DT2811, which requires this). This is controlled by jumpers 16 and 35, as shown in table 3.1.

Table 3.1. Compatibility mode jumpers.

Compatibility Mode	Jumpers	
	W16	W35
Normal (PC-74) mode	Out	In
DT2811 Mode	In	Out

Table 3.2. Base address jumper settings.

Base Address	Jumpers						
	W17	W19	W20	W21	W27	W24	W23
200H	In	In	In	In	In	In	In
208H	In	In	In	In	In	In	Out
210H	In	In	In	In	In	Out	In
218H	In	In	In	In	In	Out	Out
220H	In	In	In	In	Out	In	In
228H	In	In	In	In	Out	In	Out
230H	In	In	In	In	Out	Out	In
238H	In	In	In	In	Out	Out	Out
240H	In	In	In	Out	In	In	In
248H	In	In	In	Out	In	In	Out
250H	In	In	In	Out	In	Out	In
258H	In	In	In	Out	In	Out	Out
260H	In	In	In	Out	Out	In	In
268H	In	In	In	Out	Out	In	Out
270H	In	In	In	Out	Out	Out	In
278H	In	In	In	Out	Out	Out	Out
280H	In	In	Out	In	In	In	In
288H	In	In	Out	In	In	In	Out
290H	In	In	Out	In	In	Out	In
298H	In	In	Out	In	In	Out	Out
2A0H	In	In	Out	In	Out	In	In
2A8H	In	In	Out	In	Out	In	Out
2B0H	In	In	Out	In	Out	Out	In
2B8H	In	In	Out	In	Out	Out	Out
2C0H	In	In	Out	Out	In	In	In
2C8H	In	In	Out	Out	In	In	Out
2D0H	In	In	Out	Out	In	Out	In
2D8H	In	In	Out	Out	In	Out	Out
2E0H	In	In	Out	Out	Out	In	In
2E8H	In	In	Out	Out	Out	In	Out
2F0H	In	In	Out	Out	Out	Out	In
2F8H	In	In	Out	Out	Out	Out	Out

Table 3.3. Base address jumper settings (cont).

Base Address	Jumpers						
	W17	W19	W20	W21	W27	W24	W23
300H	In	Out	In	In	In	In	In
308H	In	Out	In	In	In	In	Out
310H	In	Out	In	In	In	Out	In
318H	In	Out	In	In	In	Out	Out
320H	In	Out	In	In	Out	In	In
328H	In	Out	In	In	Out	In	Out
330H	In	Out	In	In	Out	Out	In
338H	In	Out	In	In	Out	Out	Out
340H	In	Out	In	Out	In	In	In
348H	In	Out	In	Out	In	In	Out
350H	In	Out	In	Out	In	Out	In
358H	In	Out	In	Out	In	Out	Out
360H	In	Out	In	Out	Out	In	In
368H	In	Out	In	Out	Out	In	Out
370H	In	Out	In	Out	Out	Out	In
378H	In	Out	In	Out	Out	Out	Out
380H	In	Out	Out	In	In	In	In
388H	In	Out	Out	In	In	In	Out
390H	In	Out	Out	In	In	Out	In
398H	In	Out	Out	In	In	Out	Out
3A0H	In	Out	Out	In	Out	In	In
3A8H	In	Out	Out	In	Out	In	Out
3B0H	In	Out	Out	In	Out	Out	In
3B8H	In	Out	Out	In	Out	Out	Out
3C0H	In	Out	Out	Out	In	In	In
3C8H	In	Out	Out	Out	In	In	Out
3D0H	In	Out	Out	Out	In	Out	In
3D8H	In	Out	Out	Out	In	Out	Out
3E0H	In	Out	Out	Out	Out	In	In
3E8H	In	Out	Out	Out	Out	In	Out
3F0H	In	Out	Out	Out	Out	Out	In
3F8H	In	Out	Out	Out	Out	Out	Out

Table 3.4. Base address jumper settings (cont).

Base Address	Jumpers						
	W17	W19	W20	W21	W27	W24	W23
600H	Out	In	In	In	In	In	In
608H	Out	In	In	In	In	In	Out
610H	Out	In	In	In	In	Out	In
618H	Out	In	In	In	In	Out	Out
620H	Out	In	In	In	Out	In	In
628H	Out	In	In	In	Out	In	Out
630H	Out	In	In	In	Out	Out	In
638H	Out	In	In	In	Out	Out	Out
640H	Out	In	In	Out	In	In	In
648H	Out	In	In	Out	In	In	Out
650H	Out	In	In	Out	In	Out	In
658H	Out	In	In	Out	In	Out	Out
660H	Out	In	In	Out	Out	In	In
668H	Out	In	In	Out	Out	In	Out
670H	Out	In	In	Out	Out	Out	In
678H	Out	In	In	Out	Out	Out	Out
680H	Out	In	Out	In	In	In	In
688H	Out	In	Out	In	In	In	Out
690H	Out	In	Out	In	In	Out	In
698H	Out	In	Out	In	In	Out	Out
6A0H	Out	In	Out	In	Out	In	In
6A8H	Out	In	Out	In	Out	In	Out
6B0H	Out	In	Out	In	Out	Out	In
6B8H	Out	In	Out	In	Out	Out	Out
6C0H	Out	In	Out	Out	In	In	In
6C8H	Out	In	Out	Out	In	In	Out
6D0H	Out	In	Out	Out	In	Out	In
6D8H	Out	In	Out	Out	In	Out	Out
6E0H	Out	In	Out	Out	Out	In	In
6E8H	Out	In	Out	Out	Out	In	Out
6F0H	Out	In	Out	Out	Out	Out	In
6F8H	Out	In	Out	Out	Out	Out	Out

Table 3.5. Base address jumper settings (cont).

Base Address	Jumpers						
	W17	W19	W20	W21	W27	W24	W23
700H	Out	Out	In	In	In	In	In
708H	Out	Out	In	In	In	In	Out
710H	Out	Out	In	In	In	Out	In
718H	Out	Out	In	In	In	Out	Out
720H	Out	Out	In	In	Out	In	In
728H	Out	Out	In	In	Out	In	Out
730H	Out	Out	In	In	Out	Out	In
738H	Out	Out	In	In	Out	Out	Out
740H	Out	Out	In	Out	In	In	In
748H	Out	Out	In	Out	In	In	Out
750H	Out	Out	In	Out	In	Out	In
758H	Out	Out	In	Out	In	Out	Out
760H	Out	Out	In	Out	Out	In	In
768H	Out	Out	In	Out	Out	In	Out
770H	Out	Out	In	Out	Out	Out	In
778H	Out	Out	In	Out	Out	Out	Out
780H	Out	Out	Out	In	In	In	In
788H	Out	Out	Out	In	In	In	Out
790H	Out	Out	Out	In	In	Out	In
798H	Out	Out	Out	In	In	Out	Out
7A0H	Out	Out	Out	In	Out	In	In
7A8H	Out	Out	Out	In	Out	In	Out
7B0H	Out	Out	Out	In	Out	Out	In
7B8H	Out	Out	Out	In	Out	Out	Out
7C0H	Out	Out	Out	Out	In	In	In
7C8H	Out	Out	Out	Out	In	In	Out
7D0H	Out	Out	Out	Out	In	Out	In
7D8H	Out	Out	Out	Out	In	Out	Out
7E0H	Out	Out	Out	Out	Out	In	In
7E8H	Out	Out	Out	Out	Out	In	Out
7F0H	Out	Out	Out	Out	Out	Out	In
7F8H	Out	Out	Out	Out	Out	Out	Out

3.4. Bus Interface jumper settings.

3.4.1. Base address

The base address setting is controlled by seven jumpers, which short out the jumper locations 17, 19 to 21, 23 to 24, and 27. As supplied by the factory, the address is set to 218H. The board occupies 8 consecutive locations. Tables 3.2, 3.3, 3.4 and 3.5 shows which of these jumpers should be installed for a particular base address.

3.4.2. Interrupt level

The interrupt setting block consists of jumper locations 31 to 34. The interrupt level may be set to level 2, 3, 5 or 7.

3.4.2.1. Jumper settings

The interrupts jumper settings are described in table 3.6.

Table 3.6. Interrupt jumper settings.

Interrupt Level	Jumpers			
	W31	W32	W33	W34
2	In	Out	Out	Out
3	Out	In	Out	Out
5	Out	Out	In	Out
7	Out	Out	Out	In
None	Out	Out	Out	Out

NOTE. Only one of the jumpers in the above block may be installed at any one time.

3.4.2.2. Selection of interrupt level

In a standard PC, interrupts are allocated as follows :

level 3	Used by COM2: (if installed)
level 4	Used by COM1: (if installed)
level 5	Used by fixed disks (XT and AT)
level 7	Used by LPT1: (if installed)

An interrupt level which is not already used must be selected. As level 3 is least often used, this is normally selected. Note that unless the interrupts are specifically enabled by software, the interrupt output from the board is tri-stated (does not have any effect on the PC bus).

3.4.3. DMA level

The DMA setting block consists of jumper locations 25, 26, 28 and 29. The DMA level may be set to level 1 or 3.

3.4.3.1. Jumper settings

The interrupts jumper settings are described in table 3.7.

Table 3.7. DMA jumper settings.

DMA Level	Jumpers			
	W25	W26	W28	W29
1	In	Out	In	Out
3	Out	In	Out	In
None	Out	Out	Out	Out

NOTE. Only one set of the jumpers in the above block may be installed at any one time.

3.4.3.2. Selection of DMA level

In a standard PC, DMA channels are allocated as follows :

level 1 Unused

level 3 Used by fixed disks (XT and AT)

A DMA level which is not already used must be selected. As level 1 is least often used, this is normally selected. Note that unless DMA is specifically enabled by software, the DMA output from the board is tri-stated (does not have any effect on the PC bus).

3.5. D/A jumper selections.

The D/A may be jumpered for several output ranges. The D/A jumpers are installed in the locations numbered 1 to 8 as shown in table 3.8 and 3.9.

Table 3.8. DAC0 range jumpers.

DAC0 Output Range	Jumpers			
	W5	W6	W7	W8
0 to +5V	Out	In	Out	In
-2.5V to +2.5V	In	Out	Out	In
-5V to +5V	In	Out	In	Out

Table 3.9. DAC1 range jumpers.

DAC1 Output Range	Jumpers			
	W1	W2	W3	W4
0 to +5V	Out	In	Out	In
-2.5V to +2.5V	In	Out	Out	In
-5V to +5V	In	Out	In	Out

3.6. A/D configuration.

3.6.1. A/D gain setting

The gain of the A/D may be set either to 1, 2, 4, and 8 (PC-74HA and HC) or to 1, 10, 100 and 500 (PC-74LA and LC), and any one of these gains selected from software.

Note that the accuracy of the system is dependant on the gain, as follows :

Gain	System Accuracy
1	Within +/-.03% FSR
2	Within +/-.035% FSR
4	Within +/-.04% FSR
8	Within +/-.05% FSR
10	Within +/-.05% FSR
100	Within +/-.07% FSR
500	Within +/-.15% FSR

3.6.2. A/D jumper settings

i. The A/D may be configured for either 16 channel singled ended operation, for 8 channel differential operation, or for pseudo-differential operation. This is done by jumper settings. The jumper settings are given in table 3.10.

ii. The A/D may also be configured for input range. these jumper setting are given in table 3.11.

Table 3.10. A/D input mode jumpers.

A/D Input Mode	Jumpers					
	W13	W14	W15	W18	W22	W30
Single-ended	In	Out	Out	In	In	In
Differential	Out	In	Out	Out	Out	Out
Pseudo-differential	Out	Out	In	In	In	In

3.6.3. Fixed jumper settings

Table 3.11. A/D input range jumpers.

A/D Input Range	Jumpers	
	W9	W10
0V to +5V	Out	In
-2.5V to 2.5V	In	In
-5V to +5V	In	Out

Certain jumpers are fixed, depending on the board version. These jumper are set at the factory, and do not require modification. They are shown in table 3.12.

Table 3.12. Fixed jumpers.

PC-74 Version	Jumpers	
	W11	W12
HA or HC	Out	In
LA or LC	In	Out

Chapter 4

Interconnections

4.1. Introduction

The PC-74 family of boards plug into IBM PC/XT/AT or compatible expansion slots at connector P1, and connects to the user's circuitry at connector J1. This chapter describes these two connectors.

4.2. Connections to the IBM backplane.

The PC-74 series of boards can be plugged into any slot of the IBM backplane, with the exception of the J8 slot of the XT. This particular slot requires the -CARDSLCT signal, which is not used on other slots. All communication to and from the host processor is carried out via this connector.

4.3. User connection.

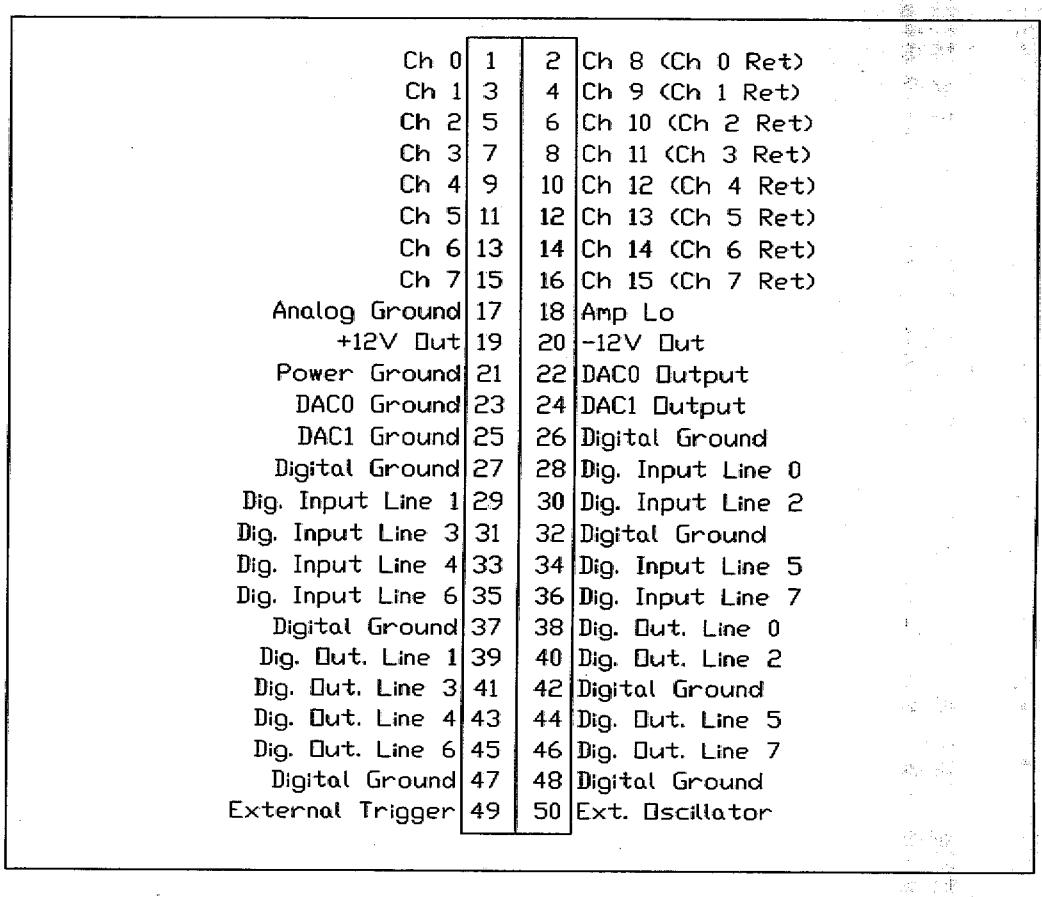
The PC-74 is connected to the user interface via a male 50 way 3M-type connector. This connector accommodates the following signals :

- 16 single ended or 8 differential lines of analog input.
- 2 lines of analog output.
- 16 digital I/O lines.
- External trigger and oscillator.

J1 also provides + and -12V power supply, with limited current output.

Figure 4.1. shows these connections, together with their pin assignments.

Fig. 4.1.
PC-74 I/O
connector.



Ch 0	1	2	Ch 8 (Ch 0 Ret)
Ch 1	3	4	Ch 9 (Ch 1 Ret)
Ch 2	5	6	Ch 10 (Ch 2 Ret)
Ch 3	7	8	Ch 11 (Ch 3 Ret)
Ch 4	9	10	Ch 12 (Ch 4 Ret)
Ch 5	11	12	Ch 13 (Ch 5 Ret)
Ch 6	13	14	Ch 14 (Ch 6 Ret)
Ch 7	15	16	Ch 15 (Ch 7 Ret)
Analog Ground	17	18	Amp Lo
+12V Out	19	20	-12V Out
Power Ground	21	22	DAC0 Output
DAC0 Ground	23	24	DAC1 Output
DAC1 Ground	25	26	Digital Ground
Digital Ground	27	28	Dig. Input Line 0
Dig. Input Line 1	29	30	Dig. Input Line 2
Dig. Input Line 3	31	32	Digital Ground
Dig. Input Line 4	33	34	Dig. Input Line 5
Dig. Input Line 6	35	36	Dig. Input Line 7
Digital Ground	37	38	Dig. Out. Line 0
Dig. Out. Line 1	39	40	Dig. Out. Line 2
Dig. Out. Line 3	41	42	Digital Ground
Dig. Out. Line 4	43	44	Dig. Out. Line 5
Dig. Out. Line 6	45	46	Dig. Out. Line 7
Digital Ground	47	48	Digital Ground
External Trigger	49	50	Ext. Oscillator

4.3.1. Signal definitions.

i. CH0 - CH15. These are the analog input lines. Note that in differential mode, CH8 functions as the return line for CH0 etc.

ii. ANALOG GROUND. One analog ground line is provided. The analog input lines are measured relative to AGND.

iii. AMP LO. This forms the return line line for the analog inputs in pseudo-differential mode.

iv. DAC0 OUTPUT. This is the analog output line for DAC0.

v. DAC0 Ground. This is the ground return line for DAC0. It is internally connected to analog ground.

vi. DAC1 OUTPUT. This is the analog output line for DAC1.

vii. DAC1 Ground. This is the ground return line for DAC1. It is internally connected to analog ground.

viii. +12. This line provides a +12 V power supply to the user's interface. Maximum permissible current draw is 20 mA. The return line for this current is POWER GROUND.

ix. -12. This line provides a -12 V power supply to the user's interface. Maximum permissible current draw is 20 mA. The return line for this current is POWER GROUND.

x. Power Ground. This is the ground return line for the power supplies. It is internally connected to analog ground.

xi. Digital Ground. This is the ground return line for the digital inputs and outputs. There are seven of these lines. Any digital circuitry tied to the digital lines should be referenced to these lines. It is internally connected to analog ground.

xii. Dig. Input Line 0 - 7. These eight lines form the digital input port.

xiii. Dig. Out. Line 0 - 7. These eight lines form the digital output port.

xiv. External Trigger. This line is used to trigger a series a A/D conversions. It is LS-TTL compatible.

xv. Ext. Oscillator. This line may be used to provide a clock to the A/D converter. It is LS-TTL compatible.

4.3.2. Recommended analog input schemes.

Analog signals can be input into the PC-74 in three possible ways - single ended, differential or pseudo-differential.

4.3.2.1. Single ended inputs.

In single ended connections, input signals share a common low side. This is analog ground. This input mode has the advantage of giving the maximum number of inputs. Its major disadvantage is the loss of common mode rejection obtainable from differential mode. Single ended inputs are very sensitive to noise. Their use is not

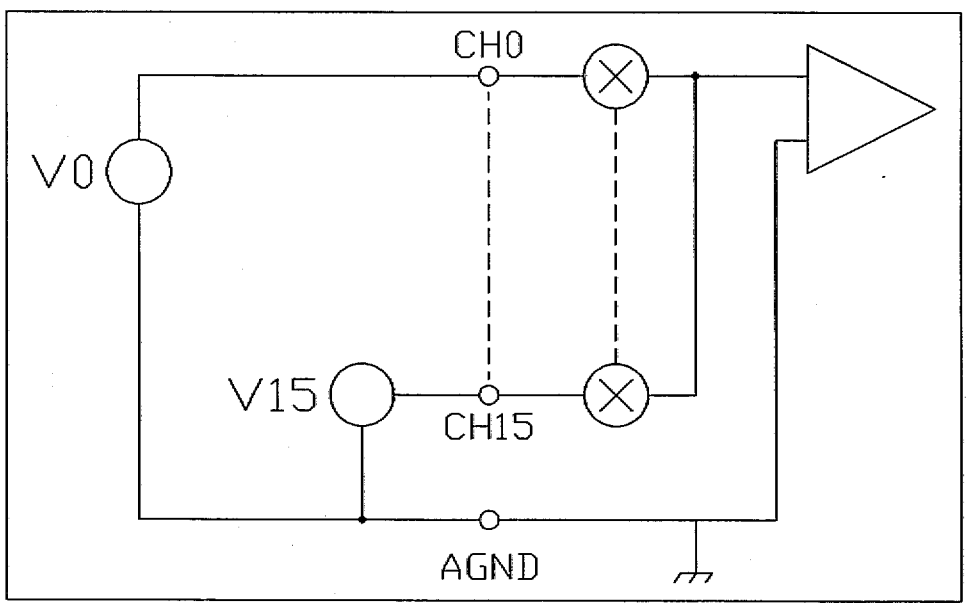


Fig. 4.2. Single ended inputs.

recommended with lead lengths of greater than 18 inches, or gains of greater than 5. Single ended input connections are shown in figure 4.2.

4.3.2.2. **Differential inputs.**

In a differential input mode, two multiplexer switches are used per channel. This means that only 8 input channels are available. The advantage however is greatly improved noise performance. Differential input connections are shown in figure 4.3.

NOTE. All signal inputs must be referred to analog ground. This can be done by connecting a 1 to 10 kilohm resistor from the low end of each input to analog ground.

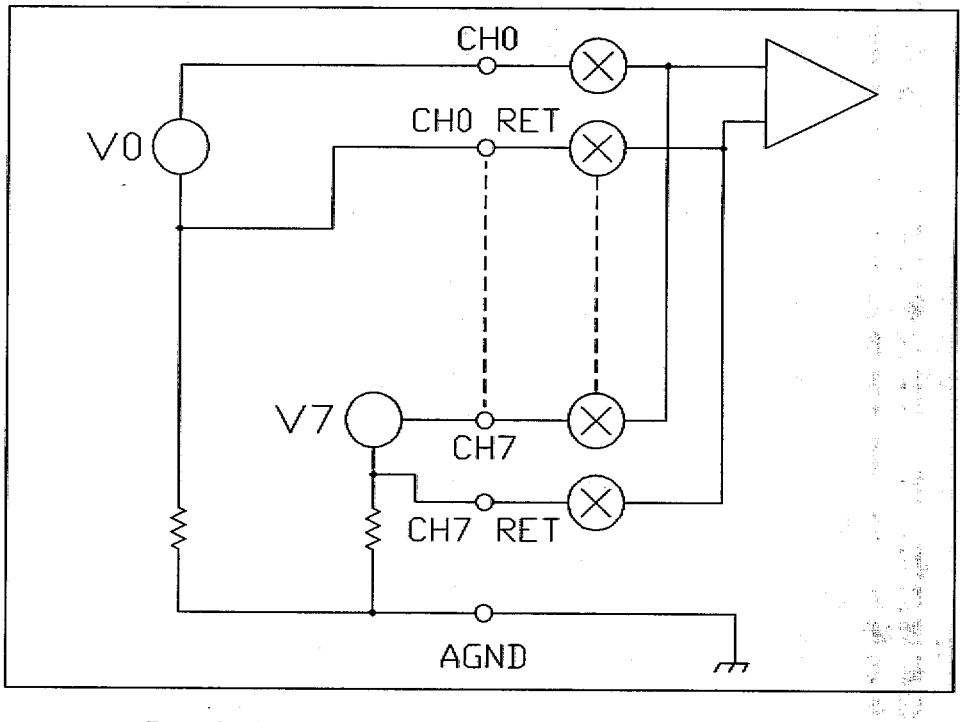


Fig. 4.3. Differential connection.

4.3.2.3. **Pseudo-differential inputs.**

Pseudo-differential input is a variation on single-ended inputs, providing some degree of common mode noise rejection, but still giving 16 input channels. The return sides of all signals are tied to Amp Lo (the low side of the instrumentation amplifier). The Amp Lo connection (pin 18, J1) is brought out to the input connector (e.g., the PC-77). Amp Lo is externally connected to analog ground, as indicated below. Pseudo-differential input connections are shown in figure 4.4.

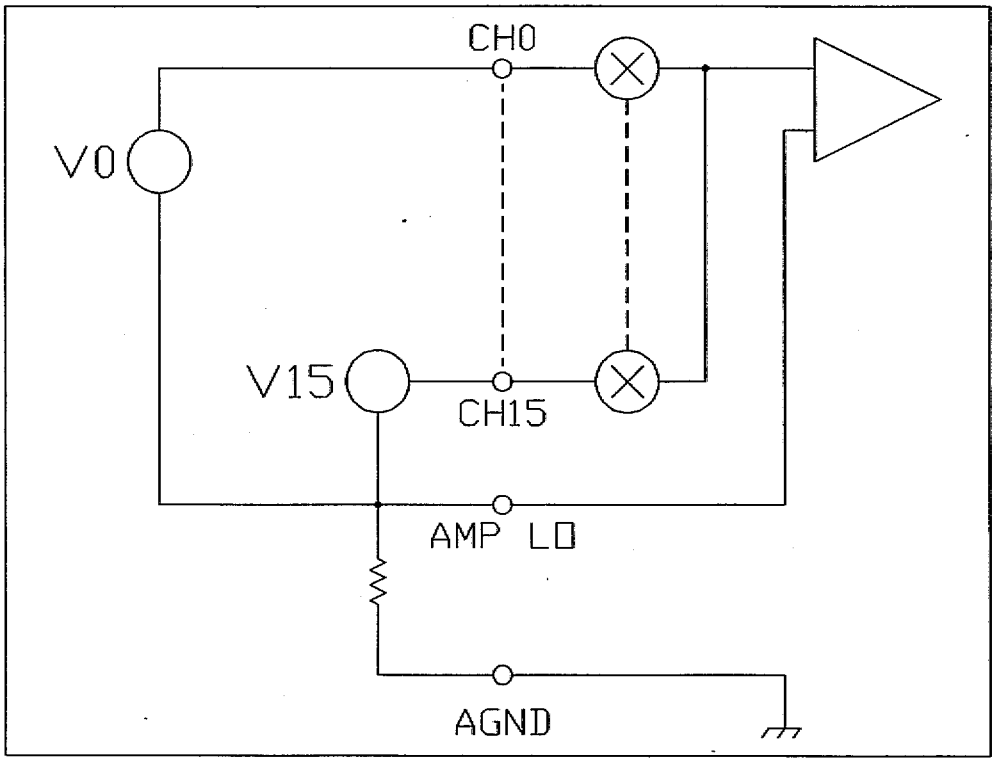


Fig. 4.4. Psuedo-differential connection.

Warning : Overloading any analog input may cause other input channels to become inaccurate or noisy.

4.3.3. Analog output

The analog output lines are referenced to the DAC0 and DAC1 ground lines. It may be jumpered to give one of three voltage outputs.

4.3.4. Digital I/O

The digital I/O lines are referenced to digital ground, as are the external trigger and oscillator lines. Note that the digital output lines are always active, and no other digital output should be connected to them.

4.4. Connection guidelines.

The PC-74 is a very high performance I/O subsystem, and was designed to to have lower input noise levels than any similar analog I/O board currently available. Its performance may however be severely affected by incorrect connection techniques. This especially true of noise levels.

4.4.1. Twisted pair input lines.

When using differential inputs, noise levels may be significantly reduced by twisting the low and high input leads of each channel together. This should NOT be done for single ended inputs.

4.4.2. Shielded input lines.

Wherever possible, leads should be shielded. Optimally, each twisted pair in a differential input system should be individually shielded. The shield should be tied to analog ground at the instrument end of the connection only.

4.4.3. Grounding.

If user circuitry is connected to the PC-74 it is of the utmost importance to keep the digital and analog ground separate.

4.4.4. Input voltages.

To maintain stated accuracy, all inputs to the PC-74 must be within 120% of full scale. Note if a channel is operating at a gain of 500, and the A/D is jumpered for -2.5 to +2.5 V operation, this means that the input voltage must be less than +- 6 mV !

Chapter 5

Programming the PC-74

5.1. Introduction

At the lowest level, the PC-74 can be programmed using I/O input and output instructions. This chapter contains the information required to do this. Although not difficult, this is time consuming, and requires detailed knowledge of the PC-74, as well as the operation of the host PC and its operating system. In order to simplify this process, a set of device drivers is provided along with the board. The use of these allows access to all board functions. These drivers are described in the next chapter.

Fig. 5.1. PC-74 register Structure.

Offset from Base	Register Name
0	A/D Control/Status
1	A/D Gain/Channel
2	A/D, DAC0 Low Byte
3	A/D, DAC0 High Byte
4	DAC1 Low Byte
5	DAC1 High Byte
6	Digital I/O Port
7	Timer Control

5.2. Register structure.

The PC-74 uses 8 consecutive address locations in I/O space. The layout of these registers is shown in fig. 5.1.

Offset from base	Function
0	A/D control/status (ADCSR) (R/W)
1	A/D gain/channel (ADGCR)(R/W)
2	A/D, DAC0 low byte (ADDAT/DADAT0)(R/W)
3	A/D, DAC0 high byte (ADDAT/DADAT0)(R/W)
4	DAC1 low byte (DADAT1)(W)
5	DAC1 high byte (DADAT1) (W)
6	DIO port0/DIO port1 (DIOP1/DIOP2) (R/W)
7	Timer/counter (TMRCTR) (R/W)

Note also that the addresses above are given as offsets from the base address of the board. This base address is jumper selected as described in the previous chapter.

Each register will now be described in detail.

5.2.1. ADCSR - A/D control status register (offset 0)

The ADCSR controls the operation of the A/D subsystem, and contains the current A/D's status information. The bit functions of the ADCSR are shown in figure 5.2.

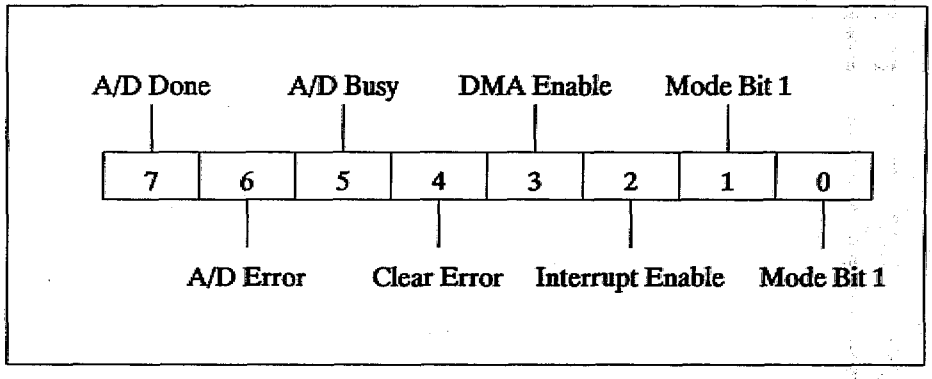


Fig. 5.2. A/D Control/Status register.

Bit 7 - A/D done, read

Set by the A/D converter to indicate completion of a conversion. If bit 2 of the ADCSR is set (interrupts are enabled), then an interrupt will be generated when bit 7 is set. If bit 3 of the ADCSR is set (DMA is enabled), then a DMA cycle will be generated when bit 7 is set. Bit 7 is cleared on power up, and by a read of the ADDAT/DADATO (high byte).

Bit 6 - A/D error, read

Set when an error occurs during an A/D conversion. Such conditions can result from two causes: either the current conversion has been completed while data from the previous conversion has not been completely read (ADDATA/DADATO high byte not read), which is a data overflow error, or an attempt has been made to initiate a new conversion while the current conversion is still in progress, which is a trigger error.

Note that on DMA operations it is normal to to have a data overflow error on completion of the operation. There should not however be any error while the transfer is in progress.

If bit 2 of the ADCSR is set (interrupts are enabled) then an interrupt will be generated if bit 6 is set. Bit 6 is cleared on power up, and by writing a 1 to bit 4 of the ADCSR.

Bit 6 is cleared on power up and by a write to bit 4 of this register.

Bit 5 - A/D busy, read

Set by an A/D strobe, and indicates that a conversion is in progress. The bit is cleared at end of conversion, and any trigger while the bit is set will cause an error condition.

Bit 4- Clear error, write

Bit 6 of ADCSR (A/D ERR) is cleared by writing 1 to this bit, which reads as 0.

Bit 3 - DMA enable, read/write

This bit is controlled by the program in use. Setting it will enable both DMA operation. All DMA is disabled, and the DMA request line to the PC bus is tri-stated, when this bit is cleared. Note that this bit can only be set if the DT2811 compatibility jumpers are set in the PC-74 mode. This is described in chapter 3. If these jumpers are in the DT2811 mode, writes to this bit are disabled, and it always reads as 0. Bit 3 is cleared on power up.

Bit 2 - Interrupt enable, read/write

This bit is controlled by the program in use. Setting it will enable both A/D done and A/D error interrupts. All interrupts are disabled when this bit is cleared. Bit 2 is cleared on power up.

Bit 1 - Mode bit 1, read/write

See bit 0.

Bit 0 - Mode bit 0, read/write

Mode bits 0 and 1 are used to control the four modes of the A/D subsystem - see the following table.

Mode	Bit 1	Bit 0	Function
0	0	0	Single conversion on ADGCR load
1	0	1	Continuous conversion, internal clock. Clk enabled on ADGCR load
2	1	0	Continuous conversion with internal clock & external trigger
3	1	1	Continuous conversion with external oscillator & external trigger

Chapter 2, Architecture, explains this in more detail.

5.2.2. ADGCR - A/D gain/channel register (offset 1)

The ADGCR controls the gain of the instrumentation amplifier, which in turn controls the gain of the input to the A/D subsystem. For single channel operations (mode 0) the ADCGR must be loaded with the gain and channel number selected for each conversion. The bit functions of the ADGCR are shown in figure 5.3.

Bits 7 & 6 - Gain select 1 & 0, read/write

These bits select the gain of the instrumentation amplifier. See the following table for bit configuration for the four possible gains.

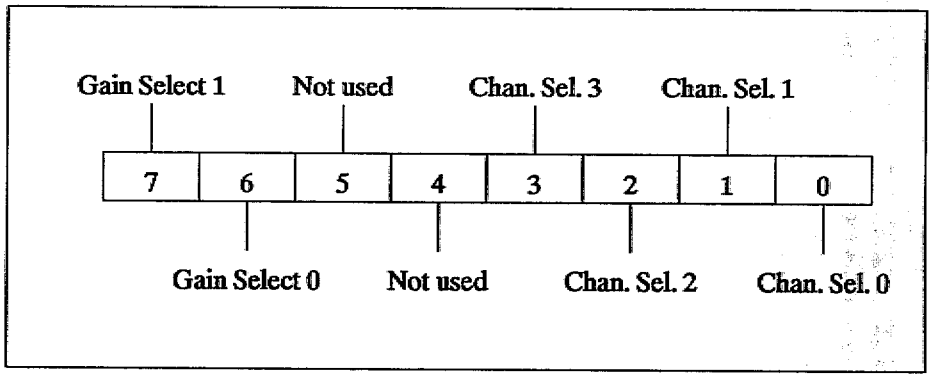


Fig. 5.3. A/D Gain/Channel register.

GS1	GS0	Gain HA/HC	Gain LA/LC
0	0	1	1
0	1	2	10

1	0	4	100
1	1	8	500

Bits 4 & 5 - Reserved

These bits read as 0, and writes to them are ignored.

Bits 0-3 - Channel select

These bits specify the input channel for the current A/D conversion. In single ended or pseudo-differential mode, bits 3-0 are used for the channel selection, but in differential mode, bit 3 is ignored, and input channel selection is by bits 2-0. The following table shows A/D channel selection bit settings in both single ended and differential modes.

Channel select bits				Channel selected	
3	2	1	0	SE	DI
0	0	0	0	0	0
0	0	0	1	1	1
0	0	1	0	2	2
0	0	1	1	3	3
0	1	0	0	4	4
0	1	0	1	5	5
0	1	1	0	6	6
0	1	1	1	7	7
1	0	0	0	8	0
1	0	0	1	9	1
1	0	1	0	10	2
1	0	1	1	11	3
1	1	0	0	12	4
1	1	0	1	13	5
1	1	1	0	14	6
1	1	1	1	15	7

5.2.3. ADDAT/DADAT0 - A/D & DAC0 register (low byte)

This register has a dual function. In A/D conversion, the A/D converter loads this register with digital data at the end of the conversion. To retrieve converted data, the user must perform a read operation to ADDAT/DADATO. In the case of D/A conversion, this register is loaded with the digital code for the D/A conversion. Bit 0 is the LSB. The layout of this register is shown in figure 5.4.

Bits 7-0 - A/D data (AD) & DAC0 data (DA)

These bits are the low byte of the 12-bit code which is returned from an A/D conversion, or else the low byte of a 12-bit code which is software-loaded into DAC0 ready for D/A conversion.

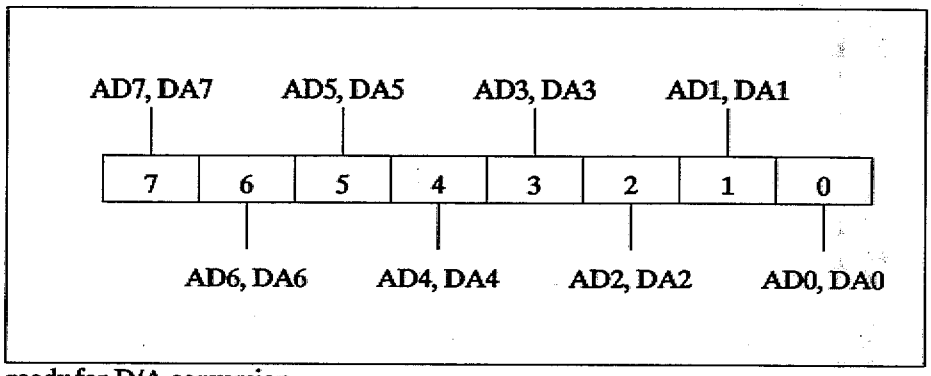


Fig. 5.4. A/D and DAC0 data register (low byte).

5.2.4. ADDAT/DADT0 - A/D & DAC0 register (high byte)

This register operates in a similar way as ADDAT/DADAT0 low byte. A read operation performed at end of a A/D conversion to retrieve converted data will clear A/D Done (ADCSR bit 7). Writing D/A data to this register begins a D/A conversion on DAC0. Data is right justified, with bit 3 (either AD11 or DA11) MSB. The layout of this register is shown in figure 5.5.

Bits 3-0 - A/D data (AD) and DAC0 data (DA)

The four higher bits of 12-bit data from an A/D conversion, or the four higher bits of the 12-bit code which is loaded into DAC0 by the software for a D/A conversion.

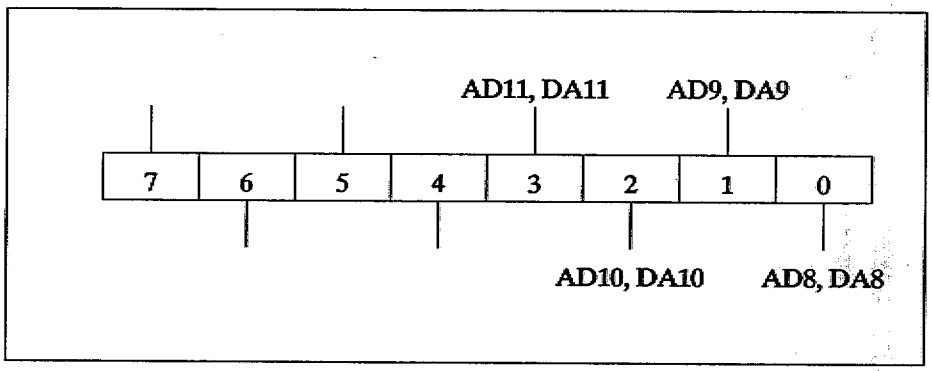


Fig. 5.5. A/D and DAC0 data register (high byte).

Bits 7-4 - Not used.

Bits 7 - 4 always read back as 0.

5.2.5. DADAT1 - DAC1 register low byte (offset 4)

This register is used to hold the eight lower bits (low byte) of the 12-bit code loaded into DAC1 by software for D/A conversions. Bit 0 is the LSB. The layout of this register is shown in figure 5.6.

Bits 7-0 - DAC1 data (DA)

These bits are the LSB of DAC1 data.

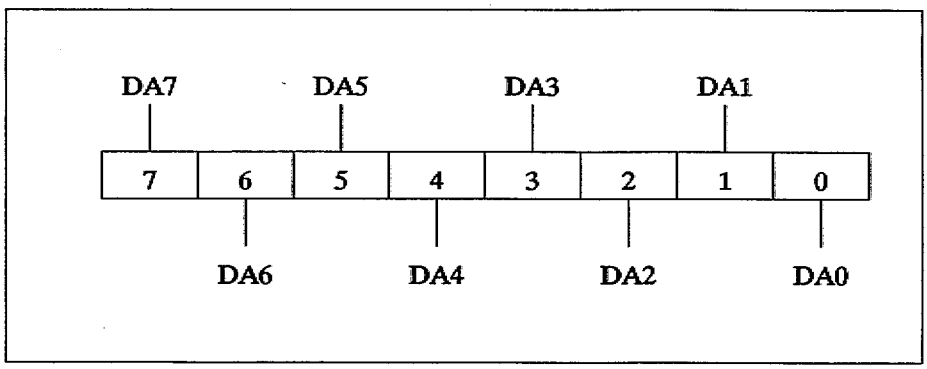


Fig. 5.6. DAC1 data register (low byte).

5.2.6. DADAT1 - DAC1 register high byte (offset 5)

DADAT1 high byte holds the four higher bits of the software-loaded 12-bit code for D/A conversion. Bit 3 is the MSB. Data is right justified. The layout of this register is shown in figure 5.7.

Bits 3-0 - DAC1 data (DA)

These four bits are the MSB of the DAC1 data.

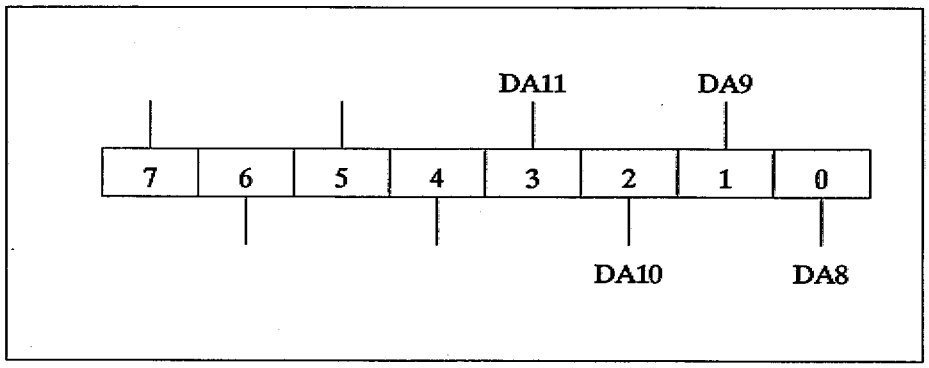


Fig. 5.7. DAC1 data register (high byte).

Bits 7-4 - Not used.

5.2.7. Digital output port 1 (offset 6)

The DIO port 1 register receives digital data which is to be transferred from the host processor to the PC-74. The layout of this register is shown in figure 5.8.

Bits 7-0 - Digital input data

Bit 7 is DIO7, and bit 0 is DIO0. Port 1 bit 1 connects to pin 28 on the J1 connector, bit 1 to pin 29, and so on.

5.2.8. Digital input port 0 (offset 6)

DIO port 0 register receives digital data from the PC-74 to be transferred to the host processor. The layout of this register is shown in figure 5.9.

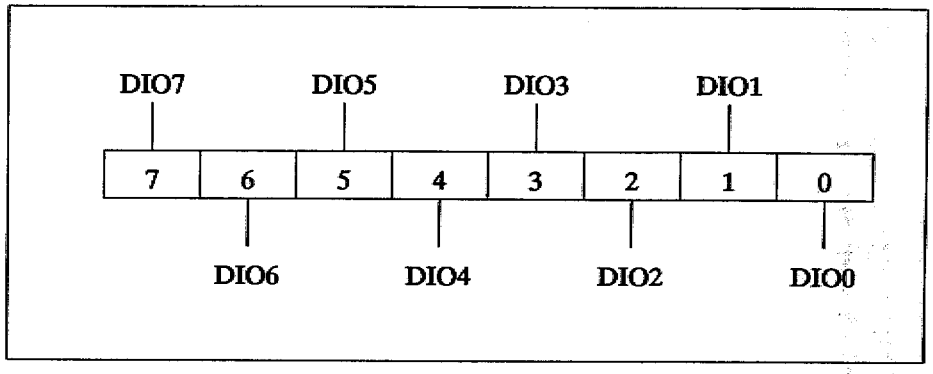


Fig. 5.8. Digital output port.

Bits 7-0 - Digital input data

Bit 0 is DIO0, and is associated with pin 38 on the J1 connector: bit 1 (DIO1) connects to pin 39, and so on.

5.2.9. TMRCTR - Timer/counter (offset 7)

TMRCTR is an 8-bit register containing the digital code for the selection of the PC-74's clock frequency. It specifies a factor by which the base frequency (pulse rate) of the internal clock or the external clock is divided, thus deriving the selected operating frequency. The layout of this register is shown in figure 5.10.

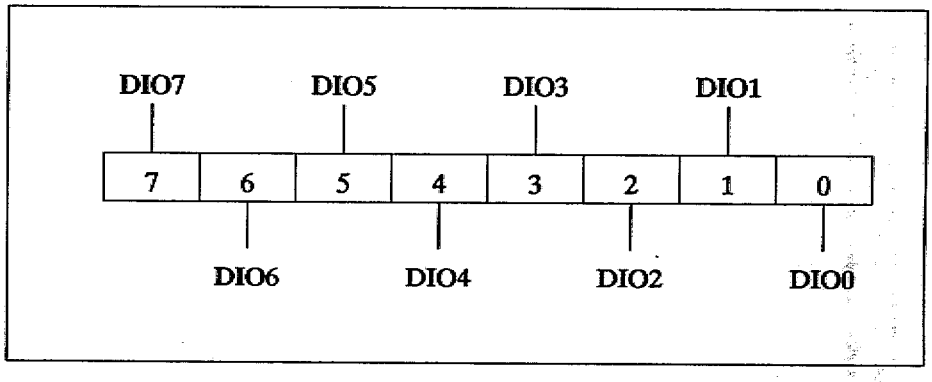


Fig. 5.9. Digital input port.

Bits 6 & 7 - Reserved

These bits read back as 0, and are ignored on write.

Bits 5-0 - Timer frequency control

These bits define the frequency by which the base frequency is divided to give the required clock circuit operating frequency. By various configurations of the TMRCTR bits, various divisor values and hence various operating frequencies can be obtained from the internal clock pulse base frequency of 600KHz. The following table shows the configurations.

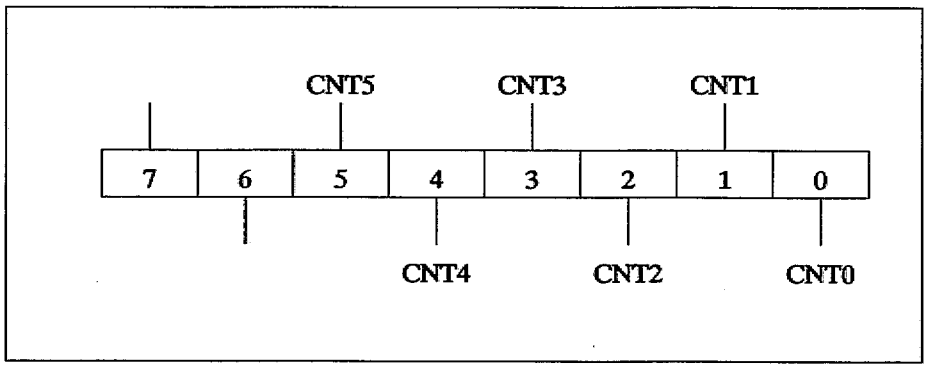


Fig. 5.10.
Timer/counter register.

TMRCTR code (hex)	Base frequency divisor	Clock frequency (Hz)
00	1	600K
01	10	60K
02	100	6K
03	1000	600
04	10000	60
05	100000	6
06	1000000	0.6
07	10000000	0.06
08	10	60K
09	100	6K
0A	1000	600
0B	10000	60
0C	100000	6
0D	1000000	0.6
0E	10000000	0.06
0F	100000000	0.006
10	2	300K
11	20	30K

12	200	3K
13	2000	300
14	20000	30
15	200000	3
16	2000000	0.3
17	20000000	0.03
18	3	200K
19	30	20K
1A	300	2K
1B	3000	200
1C	30000	20
1D	300000	2
1E	3000000	0.2
1F	30000000	0.02
20	4	150K
21	40	15K
22	400	1.5K
23	4000	150
24	40000	15
25	400000	1.5
26	4000000	0.15
27	40000000	0.015
28	5	120K
29	50	12K
2A	500	1.2K
2B	5000	120
2C	50000	12
2D	500000	1.2
2E	5000000	0.12
2F	50000000	0.012
30	6	100K
31	60	10K

32	600	1K
33	6000	100
34	60000	10
35	600000	1
36	6000000	0.1
37	60000000	0.01
38	12	50K
39	120	5K
3A	1200	500
3B	12000	50
3C	120000	5
3D	1200000	0.5
3E	12000000	0.05
3F	120000000	0.005

Output frequencies above 80KHz are not recommended. Performance can be improved by disabling the system clock during clocked conversion cycles, and, when using polled I/O, disabling the system interrupts.

5.3. Initialization

In order to initialize the A/D, the following steps should be performed. Note that the function Clean, supplied with the PC-74 driver (described in chapter 6), performs this function. This sequence should be followed before writing a new mode value to the control/status register.

i. Write 0 to the A/D control/status register.

ii. Wait at least 100 uS.

iii. Read the high and low byte of the A/D data register.

iv. Write 10 hex to the A/D control/status register.

The A/D subsystem is then ready for operation.

[This page intentionally left blank.]

Chapter 6

PC-74 Driver Software.

6.1. Introduction.

The PC-74 driver software has been written to allow the easy use of all PC-74 functions from various high level languages. Also included with the driver are :

- Source code (in C) for the entire driver system.
- Demonstration software in a variety of languages.

6.1.1. Languages supported.

The following languages are supported :

- TURBO C (V 1.5 or later).
- TURBO PASCAL (version 4.0 or later). The TURBO PASCAL driver does not support interrupts.
- Microsoft C version 5.1 or later.
- Microsoft FORTRAN version 5 or later.
- Microsoft QuickBasic version 4.5 or later.
- Almost all other compiled languages can also be used in conjunction with one or the other of the object modules.

6.2. Function Quick Reference.

This section provides a quick reference to the various driver functions. Detailed descriptions are provided later.

Function	Description
Version	Returns the driver version number.

Set_base	Sets the board base address.
Get_base	Returns the board base address.
Set_gain	Sets channel gain.
Get_gain	Returns the channel gain setting.
Diag	Diagnostics function.
Rtc_off	Switches the PC real-time clock off.
Rtc_on	Switches the PC real-time clock on.
Clean	Readies the board for a A/D conversion.
Init	Initializes the board.
D_in	Digital input.
D_out	Digital output.
Clk_md	Sets the A/D clock and trigger modes.
Ad_clock	Sets the A/D conversion clock rate.
Read_clock	Reads the A/D conversion rate.
Ad_in	Obtains a single A/D sample.
Ad_in_1, Ad_in_2	Obtains a single A/D value for systems with PC-81 channel expansion boards.
Da_out	Outputs a value to a D/A converter.
S_chan	Obtains a block of samples from a single channel.
M_chan	Obtains a block of samples from a sequence of channels.
Mi_chan	Obtains a block of samples from a sequence of channels using interrupts. This is a back-ground task.
Int_chk	Checks for completion of Mi_chan.
Int_close	Shuts down the interrupt system after Mi_chan.
Sd_chan	Obtains a block of samples from a single channel under DMA. This is a back-ground task.
Dma_init	Initializes the DMA system.
Dma_chk	Checks for completion of Sd_chan.
Dma_close	Shuts down the DMA system after Sd_chan.

6.3. Library contents.

The PC-74 driver system consists of five basic modules. Four of these modules are written in C, and one in assembler. These modules are compiled and linked in different ways for the various supported languages. The modules are the following :

- PC74S.C. This module is the basic module of the set, and must always be used. It included simple program transfer operations, as well certain general purpose utility functions. Global variables are also declared in this module. It may be compiled from PC74S.C using either TURBO C version 1.5 or Microsoft C version 5.1 or later. Most other C compilers should also be able to compile this module.
- PC74C.C. This module contains code for DMA operations. It may be compiled using either TURBO C version 1.5 or Microsoft C version 5.1 or later.
- PC74I.C. This module contains interrupt operation support code. It should only be included if interrupt operation are used. It may be compiled using either TURBO C version 1.5 or Microsoft C version 5.1 or later.
- PC74D.C. This is the diagnostics module, and contains only the diagnostics routines for the PC-74. It need only be included if you make use of the diagnostics routines. It may be compiled using either TURBO C version 1.5 or Microsoft C version 5.1 or later.
- PC74A.ASM. This is the BIOS access module, and contains routines to allow the driver to access the PC's BIOS in a language independent way. The routines in the module can, if you wish, be replaced by routines from your compiler's run-time library.

  The PC74A module may be compiled using Microsoft MASM version 5 or later. If the symbol "lg" is defined, an object file suitable for use with programs using medium, large or huge memory models will be produced. If the symbol "lg" is not defined, an object file suitable for use with programs using tiny, small or compact memory models will be produced.

6.4. Source code.

Source code for the entire driver system can be found in the \SOURCE directory of the distribution disk. Also in this directory is a complete make file for all the library files. This file complies driver modules for all possible languages for all possible memory models. Normally you will wish to edit this file to generate only those modules you actually require. To generate all possible modules, you will require Microsoft C (V5.1), Microsoft MASM (V5), as well as TURBO C. NOTE : Although we encourage you to modify the supplied source code to suit your purposes, we cannot provide technical support for such modifications.

6.5. Library Modules.

The driver code is supplied in the form of library modules for several languages, object modules, two QuickLibraries and a TURBO PASCAL unit. For any given program, you will only require one of these, depending on the functions you require, and the memory model in use.

6.5.1. Library module naming.

There are 12 library modules, all of which are different versions of a single basic module. This basic module is described below. The basic library name is as follows:

- 74M. This is the driver library module, and consists of 4 object modules. These are described below.

This module comes in 12 different forms, to suit different languages and memory models.

The actual module names are formed by combining the base name given above with a prefix and a postfix. The prefix gives the language for which the module is written, and the postfix the memory model.

6.5.1.1. Module Prefix.

The following prefixes are used.

i. P : This indicates that the module is intended for languages which support the PASCAL naming convention. (for more information on calling/naming conventions, see appendix B). Note : These modules cannot be used in conjunction with TURBO PASCAL. Special modules are provided for this purpose. See below.

ii. C : This indicates that the module is intended for languages which support the C naming convention. (for more information on calling/naming conventions, see appendix B).

6.5.1.2. Module Postfix.

i. _S : Small memory model.

ii. _C : Compact memory model.

iii. _M : Medium memory model.

iv. _L : Large memory model.

v. _H : Huge memory model.

For more information on memory models, see appendix B.

Example : The main module for a C program which uses the small model would be C74M_S.LIB.

6.6. Object Modules.

Each library file discussed above consists of one or more object files. These can be extracted for use on their own, if required. this is done by the LIB program supplied with your compiler.

6.6.1. Object module naming.

The actual module names are formed by combining the base module name given above with a prefix and a postfix. The prefix gives the language for which the module is written, and the postfix the memory model. These prefixes and postfixes are the same as described for the library modules above.

Example : The diagnostics module for a C program which uses the small model would be C74D_S.OBJ.

6.7. Microsoft C.

All supplied modules are directly compatible with Microsoft C version 5.1. You can either link your program to any of the C74m_x.LIB library files supplied (depending on memory model), or simply compile the C source code files. For example, to compile and link DEMO4.C in the large memory model use :

```
cl /AL demo4.c c74m_l.lib
```

6.7.1. Required files

PC74.H, C74M_x.LIB (depending on memory model)

6.7.2. Examples

DEMO4.C, DEMO5.C, DEMO6.C

6.8. Microsoft QuickC

Microsoft QuickC version 1 is fully supported for both the stand-alone and integrated environment operation. In order to operate within the integrated environment, you must load the PC-74 QuickLibrary, C74M_M.QLB, at startup. This is done as follows :

```
qc /lc74m_m.qlb
```

Your C program must also include the PC74.H file.

If you are using QuickC is stand-alone mode (via the QCL command), you must link your program with the supplied C74M_M.LIB library. If you chose to generate a .EXE file from within QuickC, this is done automatically by QuickC. Note this applies to medium model programs only.

If you are using a version of QuickC other than 1, you may have to regenerate the C74M_M.QLB QuickLibrary. The MA74 make file contains the commands required to do this.

6.8.1. Required files

PC74.H, C74M_M.QLB and C74M_M.LIB

6.8.2. Examples

DEMO4.C, DEMO5.C, DEMO6.C

6.9. Turbo C.

All supplied source code modules are directly compatible with Turbo C version 1.5. Note however that the supplied library files are NOT compatible with all versions of Turbo C. To use Turbo C, you should recompile all the C modules under your version of Turbo C. To compile correctly under Turbo C, you must define the "tcc" symbol.

If you are using the interrupt module (PC74I.C) then you must also link to PC74AS.OBJ (tiny, small or compact memory model) or PC74AL.OBJ (medium, large or huge memory model).

6.9.1. Required files

PC74.H, REG_74.H, PC74S.C, PC74D.C, PC74I.C, PC74C.C and either PC74AS.OBJ or PC74AL.OBJ.

6.9.2. Examples

DEMO4.C, DEMO5.C, DEMO6.C

6.10. TURBO PASCAL.

Turbo pascal requires special naming conventions which are not compatible with standard languages. For this reason special versions of object modules 74S, 74D and 74C are generated by compiling with Turbo C. The object modules used are T74S_L.OBJ, T74D_L.OBJ and T74C_L.OBJ. The MA74 file contains the command used to generate these three files. Note that in order to compile for Turbo Pascal, the "tpc" symbol must be defined.

Note : The interrupt module, 74I, cannot be used in conjunction with TURBO PASCAL, and the PC74A module is hence not required.

The three special modules are combined into a single TURBO PASCAL unit, PC74.TPU.

This module, which is compiled from PC74.PAS, contains all three driver modules. Note that the global variables are defined in this module, and should NOT be declared in any other unit or program. In order to use the Turbo Pascal Module, all that is required is to declare the PC74 unit under the "USES" clause of the Turbo Pascal program.

6.10.1. Required files

PC74.PAS, PC74.TPU

6.10.2. Examples

DEMO1.PAS, DEMO2.PAS, DEMO3.PAS

6.11. Microsoft QuickBasic

Microsoft QuickBasic version 4.5 is fully supported for both the stand-alone and integrated environment operation. In order to operate within the integrated environment, you must load the PC-74 QuickLibrary, C74M_L.QLB, at startup. This is done as follows :

```
qb /l c74m_l.qlb
```

Your BASIC program must also include the PC74.INC file.

If you are using Quickbasic is stand-alone mode (via the BC command), you must link your program with the supplied C74M_L.LIB library. If you chose to generate a .EXE file from within QuickBasic, this is done automatically by QuickBasic.

If you are using a version of QuickBasic other than 4.5, you may have to regenerate the C74M_L.QLB QuickLibrary. The MA74 make file contains the commands required to do this.

6.11.1. Required files

PC74.INC, C74M_L.QLB and C74M_L.LIB

6.11.2. Examples

DEMO7.BAS and DEMO8.BAS

6.12. Microsoft FORTRAN.

Microsoft FORTRAN version 5 is fully supported. In order to use FORTRAN, you must include the two FORTRAN include files, PC74.FI and PC74.FD into your source code. The PC74.FI file contains function and subroutine definitions, and need be included only at the start of the program. PC74.FD contains variable declarations, and must included into each section of the program that uses driver functions.

You must link your programs to the C74M_L.LIB library file. For example, to compile and link DEMO9.FOR :

```
fl demo9.for c74m_l.lib
```

6.12.1. Required files

PC74.FI, PC74.FD, C74M_L.LIB

6.12.2. Examples

DEMO9.FOR, DEMO10.FOR

6.13. C Include files.

Two files which are designed to be included into C source code are included on the distribution disk.

6.13.1. PC74.H

This file is designed for inclusion with C programs which make use of any of the driver modules. It contains prototype declarations for all C functions used.

6.13.2. REG_74.H

This file is designed for inclusion with C programs in which the user accesses the PC-74 directly. It contains definition for the various PC-74 registers.

6.14. Global variables.

The PC-74 driver makes use of two global variables. These are normally defined (storage allocated) in the PC74S driver module, EXCEPT in the case of Turbo Pascal. These two variables must be initialized by the user program. Note that for TURBO PASCAL programs these variables are defined in the Turbo Pascal module PC74.PAS.

Under most languages, these variables can be accessed either directly, of via the Set_base, Get_base, Set_gain and Get_gain functions. However, under QuickBasic, direct access is not possible, and the access routines must be used.

The variables are the following.

6.14.1. base_74.

This is a single integer value, in which is stored the base address of the PC-74 board in use. It should not be altered while a interrupt or DMA operation is in progress.

6.14.2. gain_74.

This is an array of 16 integer values, indexed from 0 to 15. Location n represents the binary value to be written to the PC74's gain register when channel n is read.

6.15. Return codes.

Four return codes are used by the PC-74 driver. These are all signed integers, and are defined in the include files and TURBO PASCAL unit files discussed above.

6.15.1. ok_74 (0).

A return code of 0 indicates that the operation was successfully performed.

6.15.2. par_74 (-1).

A return code of -1 indicates that the function was not performed as one or more of the function parameters were out of range or invalid.

6.15.3. err_74 (-2).

A return code of -2 indicates that a error was encountered while performing an operation. This is normally due to a data overrun, typically as a result of too high a clock rate.

6.15.4. n_comp_74 (1).

A return code of 1 indicates that a interrupt or DMA operation has not yet completed.

6.16. Data Format.

Analog data from the A/D and to the D/A converters is always in the form of right justified offset binary code. (The most negative voltage is represented as 0 Hex, the most positive as FFF hex). The four most significant bits of any A/D data are always 0. The most significant four bits of D/A data are ignored.

Analog voltages may be calculated from digital codes and vice-versa as follows:

i. For unipolar operation (0 to full scale) :

```
Voltage(unip) = (Digital code)(Full scale)/4096
```

ii. For bipolar operation (-full scale to + full scale) :

```
Voltage(bip) = (Digital code - 2048)(Full scale)/2048
```

6.17. OS/2 operation.

The PC-74 driver system supports OS/2 operations, with certain restrictions.

DMA and interrupt operations can be performed only by a device driver under OS/2. As a result, the PC74I and PC74C module cannot be used, and the PC74A module is no longer required.

Certain other driver functions cannot be used. These are specified in the function reference.

A demonstration program, DEMO11.C, illustrates operation under OS/2. To compile this, you will require Microsoft C V5.1, and the OS/2 development toolkit. The make file MP74 generates DEMO11. Note that it is generated as a bound mode application.

The file DEMO11.DEF is a definition file for the DEMO11 program, and illustrates how to declare PC-74 functions as IOPL and exports.

Remember that when writing OS/2 applications that OS/2 requires you to obtain port access prior to calling any driver function. This is however not functional on current versions of OS/2.

6.18. Function Reference.

This section contains full data on all available functions. Where examples are listed, this refers to the demonstration programs in the root directory of the distribution disk.

6.18.1. Ad_clock.

Name :	ad_clock - sets the clock divisor.
C Usage :	#include <PC74.H> void Pascal ad_clock(int clk_val);
PASCAL Usage :	procedure(clk_val : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE SUB ad.clock ALIAS "ad_clock" (BYVAL clk.val%)
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to subroutine ad_clock(clk_val) integer*2 clk_val [VALUE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	Sets the A/D clock divider ratio. This division ratio applies to both the internal (600 KHz) and external clock inputs. The division ratio is calculated as follows : The division ratio consists of a base ratio and a multiplier. The multiplier is set by bits 3-5, as follows: 0 - 1 1 - 10 2 - 2 3 - 3 4 - 4 5 - 5 6 - 6 7 - 12 The base ratio is set by the least significant bits, bits 0-2, as follows :

0 - 1

1 - 10

2 - 100

3 - 1000

4 - 10000

5 - 100000

6 - 1000000

7 - 10000000

Example : A clk_val of 2C hex would give a division ratio of

5 x 10000 = 50000

Using the internal clock, this would give an operating frequency of 12 Hz.

Return value : None

Global variables : base_74

See Also : Read_clock.

Example : Demo5.c, demo6.c, demo3.pas

6.18.2. Ad_in.

Name : ad_in - Gets a single A/D input reading.

C Usage : #include <PC74.H>

int Pascal ad_in(int chan, int *in_val);

PASCAL Usage : procedure ad_in(chan : integer, var in_val : integer);

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION ad.in% ALIAS "ad_in" (BYVAL chan%, SEG in.val%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

```
interface to function ad_in (chan, in_val)
integer*2 ad_in
integer*2 chan [VALUE]
integer*2 in_val [REFERENCE]
end
```

Module : PC74S

OS/2 compatible Yes

Description : This function obtains a sample from channel chan, and places the result in the integer variable in_val.

Return value : ok_74 - operation performed

par_74 - Invalid channel number.

err_74 - A/D hardware error.

Global variables : base_74, gain_74

See Also : ad_in_1, ad_in_2.

Example : demo2.pas

6.18.3. Ad_in_1.

Name : ad_in_1 - Reads a single A/D value.

C Usage : #include <PC74.H>

int Pascal ad_in_1(int chan, int *in_val);

PASCAL Usage : procedure ad_in_1(chan : integer, var in_val : integer);

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION ad.in.1% ALIAS "ad_in_1" (BYVAL chan%, SEG in.val%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

```
interface to function ad_in_1 (chan, in_val)
integer*2 ad_in_1
integer*2 chan [VALUE]
integer*2 in_val [REFERENCE]
end
```

Module : PC74S

OS/2 compatible : Yes

Description : This function is similar to ad_in, but is used to obtain a single A/D sample from an up to 256 channel system which uses a single layer of sub-multiplexed PC-81 expansion boards. The digital output lines are used to address the extender boards.

Return value : ok_74 - operation performed

par_74 - Invalid channel number.

err_74 - A/D hardware error.

Global variables : base_74, gain_74

See Also : ad_in, ad_in_2.

6.18.4. Ad_in_2.

Name : ad_in_2 - Reads a single A/D value.

C Usage : #include <PC74.H>

	int Pascal ad_in_2(int chan, int *in_val);
PASCAL Usage :	procedure ad_in_2(chan : integer, var in_val : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE FUNCTION ad.in.2% ALIAS "ad_in_1" (BYVAL chan%, SEG in.val%)
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to function ad_in_2 (chan, in_val) integer*2 ad_in_2 integer*2 chan [VALUE] integer*2 in_val [REFERENCE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	This function is similar to ad_in, but is used to obtain a single A/D sample from an up to 4096 channel system which uses a double layer of sub-multiplexed PC-81 expansion boards. The digital output lines are used to address the extender boards.
Return value :	ok_74 - operation performed
	par_74 - Invalid channel number.
	err_74 - A/D hardware error.
Global variables :	base_74, gain_74
See Also :	ad_in, ad_in_1.

6.18.5. Clean.

Name :	Clean - Clears the A/D in preparation for a conversion.
C Usage :	#include <PC74.H>
	void Pascal clean();
PASCAL Usage :	procedure clean;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE SUB clean ALIAS "clean"
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to subroutine clean end
Module :	PC74S
OS/2 compatible	Yes

Description : This procedure disables interrupts and DMA, and then waits long enough to ensure that no conversion is in progress. The error flag is then cleared. This procedure need not be called prior to the use of any of the other driver functions, but is supplied to assist users which wish to write programs which interact directly with the hardware.

Note that Clean sets the clock mode to 0.

Return value : None

Global variables : base_74

See Also : Dma_init.

6.18.6. Clk_md.

Name : clk_md - Sets the clock mode.

C Usage : #include <PC74.H>

void Pascal clk_md(int clock_md);

PASCAL Usage : procedure clk_md(clock_md : integer);

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE SUB clk.md ALIAS "clk_md" (BYVAL clock.md%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

```
interface to subroutine clk_md(clock_md)
integer*2 clock_md [VALUE]
end
```

Module : PC74S

OS/2 compatible Yes

Description : This procedure sets the A/D clock and trigger mode. Values are as follows :

0 - Single conversion.

1 - Internal clock, internal trigger.

2 - Internal clock, external trigger.

3 - External clock, external trigger.

Only the lower two bits of the procedure argument are used.

Note that function Clean resets the clock mode to 0.

Return value : None

Global variables : base_74

See Also : Ad_clock.

Example : demo5.c, demo6.c, demo7.c, demo3.pas

6.18.7. Da_out.

Name :	da_out - outputs a value to a D/A.
C Usage :	#include <PC74.H> int Pascal da_out(int chan, int out_val);
PASCAL Usage :	function da_out(chan, out_val : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION da.out% ALIAS "da_out" (BYVAL chan%, BYVAL out.val%)
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function da_out (chan, out_val) integer*2 da_out integer*2 chan [VALUE] integer*2 out_val [VALUE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	Out_val is sent to D/A chan. Values for chan are 0 or 1. Coding is offset binary.
Return value :	ok_74 - operation performed par_74 - Invalid channel number.
Global variables :	base_74
See Also :	
Example :	demo2.pas, demo3.pas

6.18.8. Diag.

Name :	diag - PC-74 diagnostics.
C Usage :	#include <PC74.H> int Pascal diag();
PASCAL Usage :	function diag : integer;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION diag% ALIAS "diag"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function diag integer*2 diag end
Module :	PC74D

OS/2 compatible	Yes
Description :	Executes diagnostic tests on the PC-74. These tests do not require any specific jumper settings, but will destroy the contents of all the PC-74 registers.
Return value :	0 - PC-74 OK. Any other value - Board faulty or not found.
Global variables :	base_74
See Also :	
Example :	Demo1.pas, demo2.pas, demo3.pas, demo4.c, demo5.c, demo6.c, demo7.bas, demo8.bas, demo9.for, demo10.for

6.18.9. D_in.

Name :	d_in - Digital input.
C Usage :	#include <PC74.H> int Pascal d_in();
PASCAL Usage :	function d_in : integer;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION d.in% ALIAS "d_in"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function d_in integer*2 d_in end
Module :	PC74S
OS/2 compatible	Yes
Description :	Obtains the value on the digital input port.
Return value :	Integer representing the digital input value.
Global variables :	base_74
See Also :	d_out.

6.18.10. DMA_chk.

Name :	dma_chk - Checks for DMA completion.
C Usage :	#include <PC74.H> int Pascal dma_chk;
PASCAL Usage :	function dma_chk;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION dma.chk% ALIAS "dma_chk"
FORTRAN Usage:	INCLUDE 'pc74.fi'

	INCLUDE 'pc74.fd'
	interface to function dma_chk integer*2 dma_chk end
Module :	PC74C
OS/2 compatible	No
Description :	Checks whether a DMA operation has completed.
Return value :	ok_74 - operation completed.
	par_74 - Invalid DMA level.
	n_comp_74 - operation not yet completed.
Global variables :	base_74
See Also :	sd_chan, dma_init.

6.18.11. DMA_close.

Name :	dma_close - Close the DMA system.
C Usage :	#include <PC74.H>
	int Pascal dma_close();
PASCAL Usage :	function dma_close;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE FUNCTION dma.close% ALIAS "dma_close"
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to function dma_close integer*2 dma_close end
Module :	PC74C
OS/2 compatible	No
Description :	Shuts down the DMA system. Note that data may not appear in the data buffer until this function is called. This procedure must be called after each DMA operation, regardless of whether the operation was completed.
Return value :	ok_74 - operation completed.
	err_74 - error in operation.
Global variables :	base_74
See Also :	sd_chan, dma_init.
Example :	Demo6.c

6.18.12. DMA_init.

Name :	dma_init - Initializes the DMA system.
C Usage :	#include <PC74.H> void Pascal dma_init();
PASCAL Usage :	procedure dma_init;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE SUB dma.init ALIAS "dma_init"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to subroutine dma_init end
Module :	PC74C
OS/2 compatible	No
Description :	This initializes internal storage prior to any DMA operations. Programs witch use DMA should call this once, prior to any DMA operation.
Return value :	None.
Global variables :	None.
See Also :	sd_chan, dma_chk.
Example :	Demo6.c

6.18.13. D_out.

Name :	d_out - Digital output.
C Usage :	#include <PC74.H> void Pascal d_out(int out_val);
PASCAL Usage :	procedure d_out(out_val : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE SUB d.out ALIAS "d_out" (BYVAL out.val%)
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to subroutine d_out(out_val) integer*2 out_val [VALUE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	Outputs out_val to the digital output port. The MSB is ignored.
Return value :	None

Global variables :	base_74
See Also :	d_in.

6.18.14. Get_base.

Name :	get_base - returns the PC-74 base address as set by Set_base.
C Usage :	#include <PC74.H> int Pascal get_base();
PASCAL Usage :	function get_base : integer
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION get.base% ALIAS "get.base"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function get_base integer*2 get_base end
Module :	PC74S
OS/2 compatible	Yes
Description :	Returns the base address of the current PC-74. This is reflected in the global variable base_74. The base address can be set either via Set_base, or directly. Note that under most languages, except QuickBasic, the value of base_74 can be read directly. Under QuickBasic however, use of the Get_base function is mandatory.
Return value :	Integer representing the base address.
Global variables :	base_74.
See Also :	
Example :	None

6.18.15. Get_gain.

Name :	get_gain - returns the gain setting for a specified channel.
C Usage :	#include <PC74.H> int Pascal get_gain(int index);
PASCAL Usage :	function get_gain(index : integer) : integer
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION get.gain% ALIAS "get_gain" (BYVAL index%)
FORTRAN Usage:	INCLUDE 'pc74.fi'

	INCLUDE 'pc74.fd'
	interface to function get_gain (index) integer*2 get_gain integer*2 index [VALUE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	Returns the gain setting for the input channel specified by index. This is reflected in the global variable gain_74. The gain can be set either via Set_gain, or directly. Note that under most languages, except QuickBasic, the value of gain_74 can be read directly. Under QuickBasic however, use of the Get_gain function is mandatory.
Return value :	Integer representing the gain setting.
Global variables :	gain_74.
See Also :	
Example :	DEMO8.BAS

6.18.16. Init.

Name :	init - initializes the PC-74.
C Usage :	#include <PC74.H>
	void Pascal init();
PASCAL Usage :	procedure init;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE SUB init ALIAS "init"
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to subroutine init end
Module :	PC74S
OS/2 compatible	Yes
Description :	Initializes the PC-74 to a known state. It should be called before any other functions (other than diagnostics).
Return value :	None
Global variables :	base_74
See Also :	
Example :	Demo2.pas, demo3.pas, demo5.c, demo6.c

6.18.17. Int_chk.

Name :	int_chk - checks for interrupt operation completion.
C Usage :	#include <PC74.H>
	int Pascal int_chk();
PASCAL Usage :	Not available.
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE FUNCTION int.chk% ALIAS "int_chk"
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to function int_chk integer*2 int_chk end
Module :	PC74I
OS/2 compatible	No
Description :	Checks whether an interrupt operation has completed, and cleans up the interrupt vectors etc.
Return value :	ok_74 - operation completed.
	err_74 - A/D hardware error.
	n_comp_74 - operation not yet completed.
	par_74 - no interrupt operation in progress.
Global variables :	base_74
See Also :	mi_chan.
Example :	Demo5.c

6.18.18. Int_close.

Name :	int_close - Disable interrupts and restore interrupt vectors.
C Usage :	#include <PC74.H>
	int Pascal int_close();
PASCAL Usage :	Not available.
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE SUB int.close ALIAS "int_close"
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to function int_close integer*2 int_close end

Module :	PC74I
OS/2 compatible	No
Description :	Disable PC-74 interrupts, and restores the interrupt vectors and gain list.
Return value :	None.
Global variables :	base_74
See Also :	mi_chan, int_chk.
Example :	Demo5.c

6.18.19. M_chan.

Name :	m_chan - Multiple channel A/D input.
C Usage :	#include <PC74.H> int Pascal m_chan(int chan_list[], int n_samp, int *f_sample);
PASCAL Usage :	function m_chan(var chan_list; n_samp : integer; var f_sample : integer) : integer;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION m.chan% ALIAS "m_chan" (SEG chan.list%, BYVAL n.samp%, SEG f.sample%)
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function m_chan(c_list, n_samp, f_sample) integer*2 m_chan integer*2 c_list[REFERENCE] integer*2 n_samp[VALUE] integer*2 f_sample[REFERENCE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	This function obtains a sequence of samples from a several channels. A list of channels is given by the parameter chan_list, the number of samples by the parameter n_samp, and the parameter f_sample is the integer variable into which the first sample is written. The channel list is an integer array of any length, and the list must be terminated by a channel number greater than 15. Once the last channel is sampled, the first channel is sampled again, until a total of n_samp conversion have been done.

Example : in order to sample channels 1, 5 and 6 twice, n_samp would be 6, and the channel list as follows :

location 0 - 1

location 1 - 5

location 2 - 6

location 3 - 16

Return value : ok_74 - operation performed

par_74 - Invalid channel number.

err_74 - A/D hardware error.

Global variables : base_74, gain_74

See Also : mi_chan.

Example : Demo3.pas, demo8.bas

6.18.20. Mi_chan.

Name : mi_chan - Multiple channel interrupt controlled input.

C Usage : #include <PC74.H>

int Pascal mi_chan(int chan_list[], int n_samp, int *f_sample, int i_lvl);

PASCAL Usage : Not available.

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION mi.chan% ALIAS "mi_chan" (SEG chan.list%, BYVAL n.samp%, SEG f.sample%, BYVAL i.lvl%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

```
interface to function mi_chan (c_l, n_s, f_s, i_l)
integer*2 mi_chan
integer*2 c_l [REFERENCE]
integer*2 n_s [VALUE]
integer*2 f_s [REFERENCE]
integer*2 i_l [VALUE]
end
```

Module : PC74I

OS/2 compatible No

Description :	This function is identical to m_chan, but operates in interrupt mode. This is useful at low sampling rates, where the host PC can continue to perform other tasks while the data acquisition is done via the interrupt system. Function int_chk is called to determine whether the operation has completed. Note that calling mi_chan only starts the operation. No actual data transfer takes place until after this routine completes. Note that procedure int_close must be called to clean up the interrupt vectors etc, prior to program termination. Values of 2, 3, 5 and 7 are valid for the interrupt level. This level must be the same as set by jumpers on the PC-74 board.
Return value :	ok_74 - operation started. par_74 - Invalid channel number.
Global variables :	base_74, gain_74
See Also :	Int_chk, int_close.
Example :	Demo5.c, demo10.for

6.18.21. Read_clock.

Name :	read_clock - Reads the current A/D clock divisor setting.
C Usage :	#include <PC74.H> int Pascal read_clock();
PASCAL Usage :	function read_clock : integer;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE FUNCTION read.clock% ALIAS "read_clock"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to function read_clock integer*2 read_clock end
Module :	PC74S
OS/2 compatible	Yes
Description :	Reads the clock divisor setting back. Values are described in the section on the ad_clock function.
Return value :	Clock divisor setting.
Global variables :	base_74
See Also :	Ad_clock

6.18.22. Rtc_off.

Name :	rtc_off - disables the PC's clock
C Usage :	#include <PC74.H> void Pascal rtc_off();
PASCAL Usage :	procedure rtc_off;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE SUB rtc.off ALIAS "rtc_off"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to subroutine rtc_off end
Module :	PC74S
OS/2 compatible	No
Description :	Disables the host PC's real-time clock interrupt. This interrupt locks out program transfer and other interrupt routines for up to 200 uS. This can result in lost data and over-run errors at A/D sampling rates higher than 1 KHz. By disabling the real-time clock, higher sampling rates can be achieved.
Return value :	None
Global variables :	None.
See Also :	rtc_on.
Example :	Demo3.pas

6.18.23. Rtc_on.

Name :	rtc_on - enables the PC's clock
C Usage :	#include <PC74.H> void Pascal rtc_on();
PASCAL Usage :	procedure rtc_on;
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC' DECLARE SUB rtc.on ALIAS "rtc_on"
FORTRAN Usage:	INCLUDE 'pc74.fi' INCLUDE 'pc74.fd' interface to subroutine rtc_on end
Module :	PC74S
OS/2 compatible	No

Description : Re-enables the host PC's real-time clock interrupt. This function is used to switch the real-time clock on again after it has been switched off by rtc_off. It is good practice to call this function as soon as possible after a data acquisition function terminates.

Return value : None

Global variables : None.

See Also : rtc_off.

Example : Demo3.pas

6.18.24. S_chan.

Name : s_chan - Obtains a set of samples from a single channel.

C Usage : #include <PC74.H>

int Pascal s_chan(int chan, int n_samp, int *f_sample);

PASCAL Usage : function s_chan(chan, n_samp : integer; var f_sample : integer) : integer;

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION s.chan% ALIAS "s_chan" (BYVAL chan%, BYVAL n.samp%, SEG f.sample%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

```
interface to function s_chan(chan, n_samp,
f_sample)
integer*2 s_chan
integer*2 chan [VALUE]
integer*2 n_samp [VALUE]
integer*2 f_sample [REFERENCE]
end
```

Module : PC74S

OS/2 compatible Yes

Description : This function obtains a sequence of samples from a single channel. The channel is given by parameter chan, the number of samples by the parameter n_samp, and the parameter f_sample is the integer variable into which the first sample is written. For example, if f_sample was the third location in an array, the second sample would go into location 4, the third into location 5 etc.

Return value : ok_74 - operation performed

par_74 - Invalid channel number.

err_74 - A/D hardware error.

Global variables : base_74, gain_74

See Also : sd_chan.

6.18.25. Sd_chan.

Name : sd_chan - Obtains a set of samples from a single channel under DMA control.

C Usage :

```
#include <PC74.H>

int Pascal sd_chan(int chan, int n_samp, int d_lvl, int *buf);
```

PASCAL Usage :

```
function sd_chan(chan, n_samp : integer; d_lvl : integer; var buf : integer) : integer;
```

QuickBasic Usage:

```
REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION sd.chan% ALIAS "sd_chan" (BYVAL chan%, BYVAL n.samp%, BYVAL d.lvl%, SEG buf%)
```

FORTRAN Usage:

```
INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

interface to function sd_chan(chan, n_samp, d_lvl, buf)
integer*2 sd_chan
integer*2 chan [VALUE]
integer*2 n_samp [VALUE]
integer*2 d_lvl [VALUE]
integer*2 buf [REFERENCE]
end
```

Module : PC74C

OS/2 compatible : No

Description :

This function obtains a sequence of samples from a single channel. It is very similar to s_chan, with the exception that the user must specify a DMA level, and that it executes in the back-ground. Function dma_chk is used to check for completion, and function dma_close is used to shut down the DMA system.

Valid setting for d_lvl are 1 or 3. This value must correspond to the setting on the PC-74 board.

Sd_chan requires that the buffer be large enough such that room for all of the samples is available in an area of the buffer which does not cross a 64K boundary. The simplest way to achieve this is use a buffer which is twice the required size. For example, to acquire 1000 samples, use a buffer of 2000 integer locations (4000 byte locations). Note that data may only be placed in the first part of the buffer by function dma_close.

	Note that function dma_init must be called prior to sd_chan.
Return value :	ok_74 - operation started.
	par_74 - Invalid channel number or DMA level.
Global variables :	base_74, gain_74
See Also :	s_chan, dma_chk, dma_close.
Example :	Demo6.c

6.18.26. Set_base.

Name :	set_base - sets the PC-74 base address.
C Usage :	#include <PC74.H>
	void Pascal set_base(base_addr);
PASCAL Usage :	procedure set_base(base_addr : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'
	DECLARE SUB set.base ALIAS "set_base" (BYVAL base.addr%)
FORTRAN Usage:	INCLUDE 'pc74.fi'
	INCLUDE 'pc74.fd'
	interface to subroutine set_base (b_addr) integer*2 b_addr [VALUE] end
Module :	PC74S
OS/2 compatible	Yes
Description :	Sets the base address of the current PC-74, as stored in the global variable base_74. The base address can be set either via Set_base, or directly. Note that under most languages, except QuickBasic, the value of base_74 can be set directly. Under QuickBasic however, use of the Set_base function is mandatory.
Return value :	None
Global variables :	base_74.
See Also :	Get_base
Example :	DEMO7.BAS, DEMO8.BAS

6.18.27. Set_gain.

Name :	set_gain - sets the gain for a specified channel.
C Usage :	#include <PC74.H>
	void Pascal set_gain(int index, int value);
PASCAL Usage :	procedure set_gain(index, value : integer);
QuickBasic Usage:	REM $INCLUDE: 'PC74.INC'

DECLARE SUB set.gain ALIAS "set_gain" (BYVAL index%, BYVAL value%)

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

interface to subroutine set_gain (index, value)
integer*2 index [VALUE]
integer*2 value [VALUE]
end

Module : PC74S

OS/2 compatible Yes

Description : Set the gain for the input channel specified by index to value. This is reflected in the global variable gain_74. The gain can be set either via Set_gain, or directly. Note that under most languages, except QuickBasic, the value of gain_74 can be set directly. Under QuickBasic however, use of the Set_gain function is mandatory.

Return value : None

Global variables : gain_74.

See Also :

Example : DEMO7.BAS, DEMO7.BAS

6.18.28. Version.

Name : version - returns the driver version number.

C Usage : #include <PC74.H>

int Pascal version();

PASCAL Usage : function version : integer

QuickBasic Usage: REM $INCLUDE: 'PC74.INC'

DECLARE FUNCTION version% ALIAS "version"

FORTRAN Usage: INCLUDE 'pc74.fi'

INCLUDE 'pc74.fd'

interface to function version
integer*2 version
end

Module : PC74S

OS/2 compatible Yes

Description : Returns the version number of the driver, encoded as follows :

High byte : Major version number.

Low byte : Minor version number.

	For example, version 2.06 would return 0206 hex.
Return value :	Integer representing the version number.
Global variables :	None.
See Also :	
Example :	Demo1.pas, demo2.pas, demo3.pas, demo4.c, demo5.c, demo6.c, demo7.bas, demo8.bas, demo9.for, demo10.for

Chapter 7

Calibration

7.1. Introduction

This chapter contains information on the calibration procedures for the A/D and D/A subsystems on the PC-74 series of boards.

These procedures should be performed at six month intervals, or whenever the input or output range jumpers are changed.

NOTE : Allow the host PC and the board to warm up for at least one hour before calibration.

7.2. A/D calibration.

A/D calibration is performed by adjusting six trimpots, R3, R5, R8, R11, R16 and R22. These trimpots are easily located from the board layout shown in appendix 3, or the labels on the PC-74 board itself.

7.2.1. Requirements

i. Calibration is done on channel 0. The recommended connector wiring is shown in figure 7.1. These connections are suitable for single ended, pseudo-differential or differential mode calibration.

ii. Calibration is performed with the board jumpered into its intended operating mode.

iii. All cables should be as short as possible.

7.2.2. Equipment required

i. Precision voltage source. Range +10 to -10 V, absolute accuracy better than 0.005%, resolution 100 nV or better.

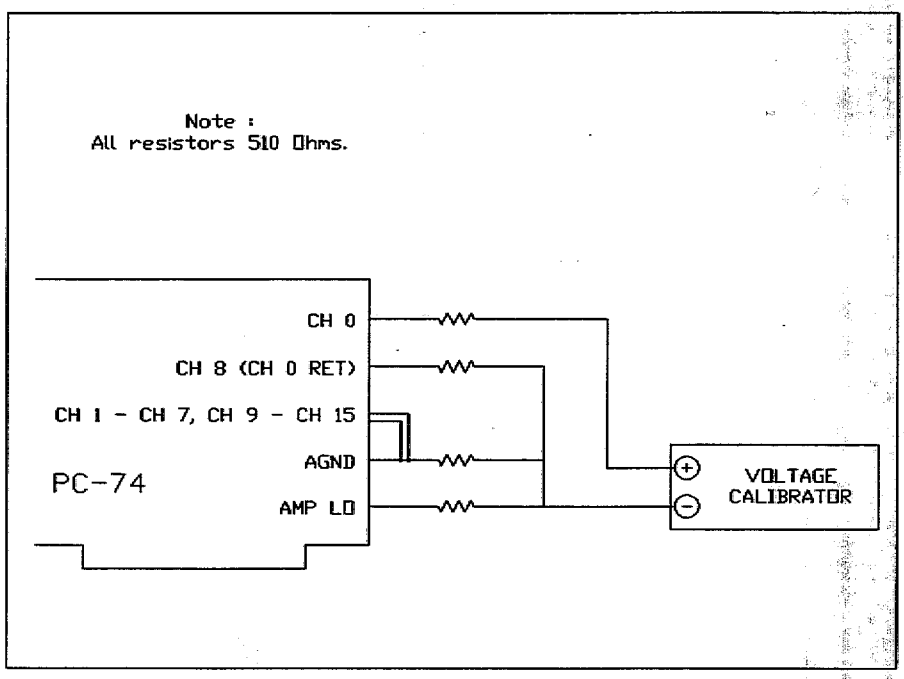


Fig. 7.1. A/D calibration connections.

7.2.3. Procedure

The PC-74 allows each gain to be trimmed individually. Note however, that the offset trimpots, R3 and R22, must be adjusted correctly prior to trimming the individual gains. Note also that for the trim procedure given here, the setting of R3, R22 and R5 interact. It is hence necessary to set R3, R22 and R5 until no further adjustment is required, then to set R8, R11 and R16. The procedure below allows for this.

Table 7.1 contains voltages for (-FS + 1/2 LSB) for all combinations of gain and input range, and table 7.2 supplies (+FS -3/2 LSB) values.

Table 7.1. -FS + 1/2 LSB table for A/D calibration.

A/D Gain	A/D Range		
	0 - 5V	-2.5 - +2.5V	-5 - +5V
1	610.35 uV	-2.499390 V	-4.998779 V
2	305.18 uV	-1.249695 V	-2.499390 V
4	152.59 uV	-624.847 mV	-1.249695 V
8	122.07 uV	-312.424 mV	-624.847 mV
10	12.21 uV	-249.939 mV	-499.878 mV
100	2.44 uV	-24.9939 mV	-49.9878 mV
500	1.22 uV	-4.9988 mV	-9.9976 mV

i. Set the A/D for a gain of 1, apply (+FS - 3/2 LSB) for a gain of 1. Adjust R5 (Gain 0 potentiometer) for an output code of which flickers evenly between FFEH and FFFH.

ii. Apply (-FS + 1/2 LSB) for a gain of 1. Adjust R22 (A/D offset potentiometer) for an output code of which flickers evenly between 000H and 001H.

iii. Apply 0V, set the A/D for its maximum gain, and adjust R3 (instrumentation amplifier offset) for an output code which is 0H (0 - 5 V range) or 800H (-2.5 to +2.5 or -5 to +5 V ranges).

iv. Repeat steps i, ii and iii above until no further adjustment in R3, R5 or R22 is required.

v. Apply (+FS - 3/2 LSB) for a gain of 2 (HA and HC boards) or 10 (LA and LC boards). Adjust R16 (Gain 1 potentiometer) for an output code of which flickers evenly between FFEH and FFFH.

vi. Apply (+FS - 3/2 LSB) for a gain of 4 (HA and HC boards) or 100 (LA and LC boards). Adjust R8 (Gain 2 potentiometer) for an output code of which flickers evenly between FFEH and FFFH.

vii. Apply (+FS - 3/2 LSB) for a gain of 8 (HA and HC boards) or 500 (LA and LC boards). Adjust R11 (Gain 3 potentiometer) for an output code of which flickers evenly between FFEH and FFFH.

Table 7.2. +FS - 3/2 LSB table for A/D calibration.

A/D Gain	A/D Range		
	0 - 5V	-2.5 - +2.5V	-5 - +5V
1	+4.998169 V	+2.498169 V	+4.996338 V
2	+2.499084 V	+1.249084 V	+2.498169 V
4	+1.249542 V	+624.542 mV	+1.249084 V
8	+624.771 mV	+312.271 mV	+624.542 mV
10	+499.817 mV	+249.817 mV	+499.634 mV
100	+49.9817 mV	+24.9817 mV	+49.9634 mV
500	+9.9963 mV	+4.9963 mV	+9.992676 mV

7.2.4. A/D Calibration Software.

The program CAL74.EXE, supplied on the distribution disk, automates the above procedure. Note that for correct operation, the setup information supplied in the first menu must be correct.

7.3. D/A calibration.

D/A calibration is performed by adjusting two trimpots for each DAC, R26 and R28 for DAC0, and R30 and R32 for DAC1. These trimpots are easily located from the board layout shown in appendix C, or the labels on the PC-74 board itself.

7.3.1. Requirements

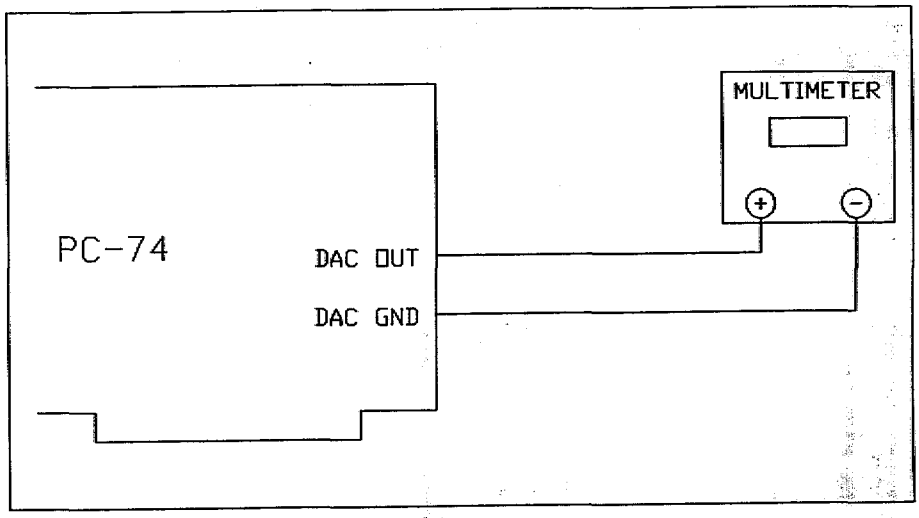


Fig. 7.2. D/A calibration connection.

i. The recommended connector wiring is shown in figure 7.2.

ii. Calibration is performed with the board jumpered into its intended operating mode.

iii. All cables should be as short as possible.

7.3.2. Equipment required

i. Precision Multimeter. Range +10 to -10 V, accuracy 10 uV for voltage.

Table 7.3. D/A calibration output values.

Code Value	D/A Output Range		
	0 - +5V	-2.5 - +2.5V	-5 - +5V
- FS	0 V	-2.5000 V	-5.0000 V
+ FS - 1 LSB	+4.9988 V	+2.4988 V	+4.9976 V
1/2 LSB	610 uV	610 uV	1.22 mV

7.3.3. Procedure

The procedure below gives the steps required to calibrate the D/A converters. Table 7.3 gives -FS voltages, (+FS - 1LSB) and 1/2 LSB values for all output ranges.

i. Set the D/A code to 000H. Use R26 (DAC0) or R30 (DAC1) to adjust the D/A output to within 1/2 LSB of the FS for that range.

ii. Set the D/A code to FFFH. Use R28 (DAC0) or R32 (DAC1) to adjust the D/A output to within 1/2 LSB of (+FS - 1 LSB).

These two adjustments interact to some extent, and hence the sequence should be repeated until no further adjustment in either trimpot is required.

7.3.4. D/A Calibration Software.

The program CAL74.EXE, supplied on the distribution disk, automates the procedure for calibrating the DAC outputs. Note that for correct operation, the setup information supplied in the first menu must be correct.

Appendix A

Hardware Specifications

A.1. Inputs

Input Impedance	On channel 100M/100pF
	Off channel 100M/10pF
Maximum common mode input voltage	10 V
Common mode rejection ratio	70 dB at 60 Hz (gain of 1)
	100 dB at 60 Hz (gain of 500)
Maximum input voltage	+-32 V (power on)
	+-20 V (power off)
Resolution	12 Bits
Nonlinearity	less than 3/4 LSB
Differential nonlinearity	less than 1/2 LSB
Quantization Uncertainty	+-1/2 LSB
A/D offset drift	10 PPM/degree C
Gain drift	30 PPM/degree C
Full scale input ranges	0 to +5V, -2.5 to +2.5V, -5 to +5V
Number of inputs	Jumper selectable 16 single ended or 8 differential.
Throughput rate	80 KHz (HC/LC), 30 KHz (HA, LA), gain of 1.
Monotonicity	0 to 70 degree C

Gain	A/D System Accuracy
1	Within +/-.03% FSR
2	Within +/-.035% FSR
4	Within +/-.04% FSR
8	Within +/-.05% FSR
10	Within +/-.05% FSR
100	Within +/-.07% FSR
500	Within +/-.15% FSR

A.2. D/A Outputs

Resolution	12 Bits
Full scale output ranges	0 to +5V, -2.5 to +2.5V, -5 to +5V
Nonlinearity	less than 0.01% FSR
Differential nonlinearity	less than 1/2 LSB
Quantization Uncertainty	+-1/2 LSB
Settling time	7 uS to +-0.01% of final value.
Offset drift	10 PPM/degree C.
Gain drift	30 PPM/degree C.
Output compliance	+-5 mA.
System Accuracy	0.025%
Monotonicity	0 to 70 degree C

A.3. Digital I/O

Number of ports	Two 8 bit ports
Digital inputs	8, ALS TTL inputs, fan-in 1 ALS load
Digital Outputs	8, ALS TTL outputs, fanout of 20 LS loads.

A.4. Board Timing

A/D may be operated either from an internal or external clock. Conversions may also be triggered either internally or externally.

A.5. IBM PC Interface

Board base address	jumper selectable 200H to 7F8H.
Board interrupt level	jumper selectable 2, 3, 5 or 7.

Board DMA level	jumper selectable 1 or 3.
Power consumption	+5V +- 0.5V, 290 mA
	+12V +- 0.5V, 50 mA
	-12V +- 0.5V, 100 mA

[This page intentionally left blank.]

Appendix B

Memory models : Technical specifications

B.1. Linkable object modules

In order to make use of the linkable object modules supplied on the PC-74 disk, it is necessary to understand certain calling conventions used on 8086 series processors. There are two aspects to this :

i. Memory model. Memory model refers to the length of the pointers used to access data and code. These may be either short (16 bit), long (32 bit) or huge (32 bit). Huge pointers differ from long pointers in that huge pointers can be used on data items of greater than 64K. Long pointers can only address up to 64K. Six memory models are in common use:

i.i. Small model. Short data and code pointers. This means that data pointers consist of a single 16 bit word, and call are of NEAR type. Program and data are in differnent segments, allowing 64K of data and 64K of program.

i.ii. Tiny model. Short data and code pointers. This model is very similar to the small model, but only one segement is used for both data and program. The tiny model is normally only used if the program is intended to be converted to a .COM format.

i.iii. Medium model. Short data pointers and long code pointers. Hence call are of FAR type. Up to 64K of data and 1M of program space is allowed.

i.iv. Compact model. Long data pointers and short code pointers. Hence call are of near type. Up to 1M of data and 64K of program space is allowed. Individual data items may only be up to 64K in size.

i.v. Large model. Long data and code pointers. Data pointers hence consist of 32 bit pointers and calls are of FAR type. Up to 1M of data and 1M

of program space is allowed. Individual data items may only be up to 64K in size.

i.vi. Huge model. Long data and code pointers. Data pointers hence consist of 32 bit pointers and calls are of FAR type. Up to 1M of data and 1M of program space is allowed. The huge model allows individual data item to be greater than 64K in size.

ii. Stack convention. Almost all software makes use of the same convention for placing data on the stack. This is to push the first parameter first, the second second etc. There is however a difference as to when these parameters are removed from the stack (e.i. the stack pointer readjusted).

ii.i. PLM calling convention. In this case it is the responsibility of the called procedure to remove any parameters (normally by a RET N). This calling convention is used by Pascal and Fortran.

ii.ii. C calling convention. In this case the calling program must remove any parameters.

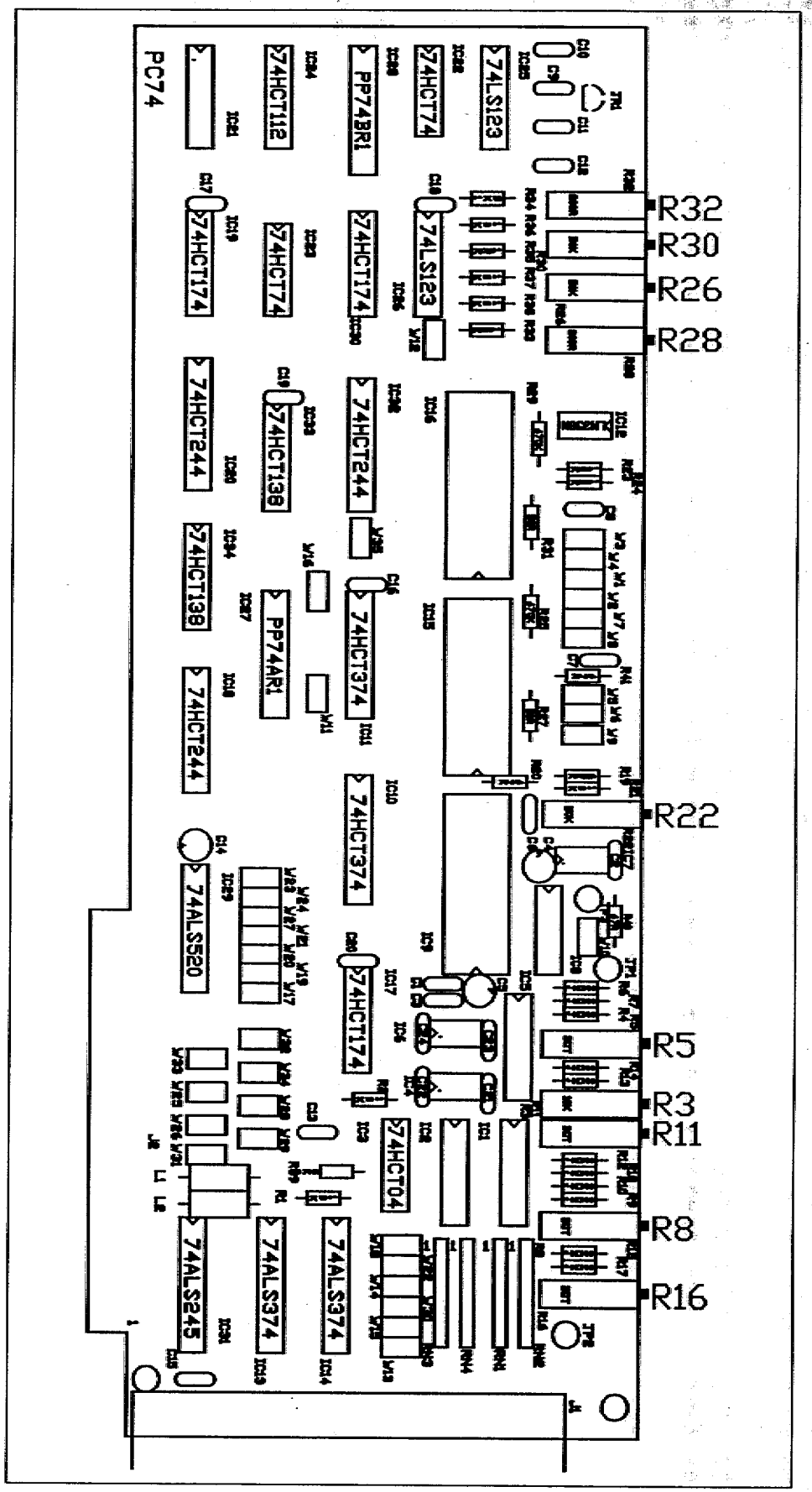
Appendix C

PC-74 Layout Diagram.

This appendix provides a layout diagram of the PC74 board. This shows the locations of :

- All trimpots
- All jumpers
- All test points

Figure C.1.
PC-74 Layout
Diagram.



Appendix D

Problem Determination Guide

D.1. Introduction

If you are experiencing problems, first check the following :

i. Remove the PC-74, and check that all ICs are firmly seated in their sockets, that there is no obvious damage to any components, and that the edge connector fingers on the PC-74 are clean.

ii. Check that the PC-74 is jumpered correctly for your application.

iii. Replace the PC-74, and check that it seats firmly in the host PC's mother-board. Also check that no components are touching a adjacent board.

iv. Check that the ribbon cable is securely plugged into the PC-74, with pin 1 (the side of the ribbon cable with the red stripe) at the top.

D.2. The diagnostics function.

The PC-74 contains a very comprehensive diagnostics program. All of the supplied demo programs as well as the calibration program use this, and can be used to diagnose the PC-74. In fact, the only PC-74 malfunctions which it will not detect are the following :

i. Damaged input multiplexer or instrumentation amplifier.

ii. Damaged D/A output amplifier.

iii. Damaged digital input or output lines.

D.3. Common problems

D.3.1. PC-74 diagnostics report card not found.

This is typically as a result of incorrect jumper settings.

D.3.2. A/D output code all zeros or all ones.

This is typically as a result of floating inputs, or an overload.

If you have exceeded the maximum input voltage (+- 32 V), you may have damaged the input multiplexers. If so, return the board to your dealer for repair.

D.3.3. A/D reading are noisy.

This may be as a result of long leads, an electrically noisy environment, or overloads on other input channels. Note also that if an input channels is overloaded it may saturate in such a way as to give a reading which appears to be in the normal range, but is very noisy.

D.3.4. The first reading in a series is inaccurate.

This is normally as a result of an overload on another input, or long leads, or a very high source impedance.

Appendix E

PC-77 Screw Terminal Panel

E.1. Introduction

E.1.1. PC-77

The PC-77 is a screw terminal board, specifically designed for the PC-74. It provides direct access to all PC-74 I/O lines. Connection to the PC-74 is made via ribbon cable. The PC-77 is fully mechanically and electrically compatible with the DT707.

A circuit diagram of all the variants of the PC-77 is shown in figure E.1.

The PC-77 can be used in conjunction with thermocouples, and 4-20 mA inputs.

E.2. Specifications

E.2.1. General

Dimensions	133 X 223 X 23 mm
Mass	370 grams
Acceptable wire size	18 to 22 AWG

E.2.2. Thermocouple Cold Junction Compensation

Output Voltage	0.5 mV per degree Celsius
Operating Temperature Range	0 to 85 degrees Celsius
Output Impedance	800 Ohm

E.3. Configuring the PC-77

The PC-77 has four jumpers, which must be set depending on input type. These are shown in table E.1.

Table E.1. PC-77 Jumpers.

PC-77 Input Type	Jumpers			
	W1	W2	W3	W4
Voltage	Out	Out	In	In
Thermocouple	In	In	In	In
4-20 mA	Out	Out	In	In

E.4. Using the PC-77

The function of each screw terminal on the PC-77 is shown on board's silkscreen, as well as in figure E.2

The PC-77 can accommodate single ended, pseudo-differential, differential, 4 to 20 mA and thermocouple inputs.

E.4.1. Single ended inputs.

Single ended inputs have a return path via analog ground. This is connected to signal ground via jumper W3.

E.4.2. Pseudo-differential inputs.

Pseudo-differential inputs are very similar to single ended inputs. Note that W4 connects AMP LO to analog ground. This is installed in the factory, but may be removed if necessary.

E.4.3. Differential inputs.

Differential inputs require a return path to analog ground, as discussed in the main text. This may be accomplished on the PC-77 by installing 10 Kilohm resistors in locations R2, R3, R6, R7, R10, R11, R14 and R15. These resistors connect from the return line of each differential input to signal ground.

E.4.4. 4 to 20 mA inputs.

4 to 20 mA inputs are accommodated by current sensing resistors. In order to measure 4-20 mA inputs, you must install the 250 Ohm current sensing resistors. These are R1, R4, R5, R8, R9, R12 and R13, and give an output voltage of 250 mV per mA. In addition, diffential mode operation must be selected on the PC-74.

E.4.5. Thermocouple inputs.

E.4.5.1. Input mode.

You should use a PC-74LA or LC jumpered for differential mode operation if you intend to measure thermocouple voltage.

E.4.5.2. Open inputs.

If you require up-scale open thermocouple detection, then resistors R26-R33 (10 MOhm) should be installed. Open inputs will result in a measured voltage of approximately 2 V. This will result in an input over range for gains of 10 or greater.

E.4.5.3. Cold junction compensation.

Jumpers W1 and W2 must be inserted for the cold-junction compensation circuit to be activated.

The voltage produced by the circuit is 0.5 mV per degree Celsius. The temperature corresponding to this voltage can then be used to compensate for the cold junction on the PC77.

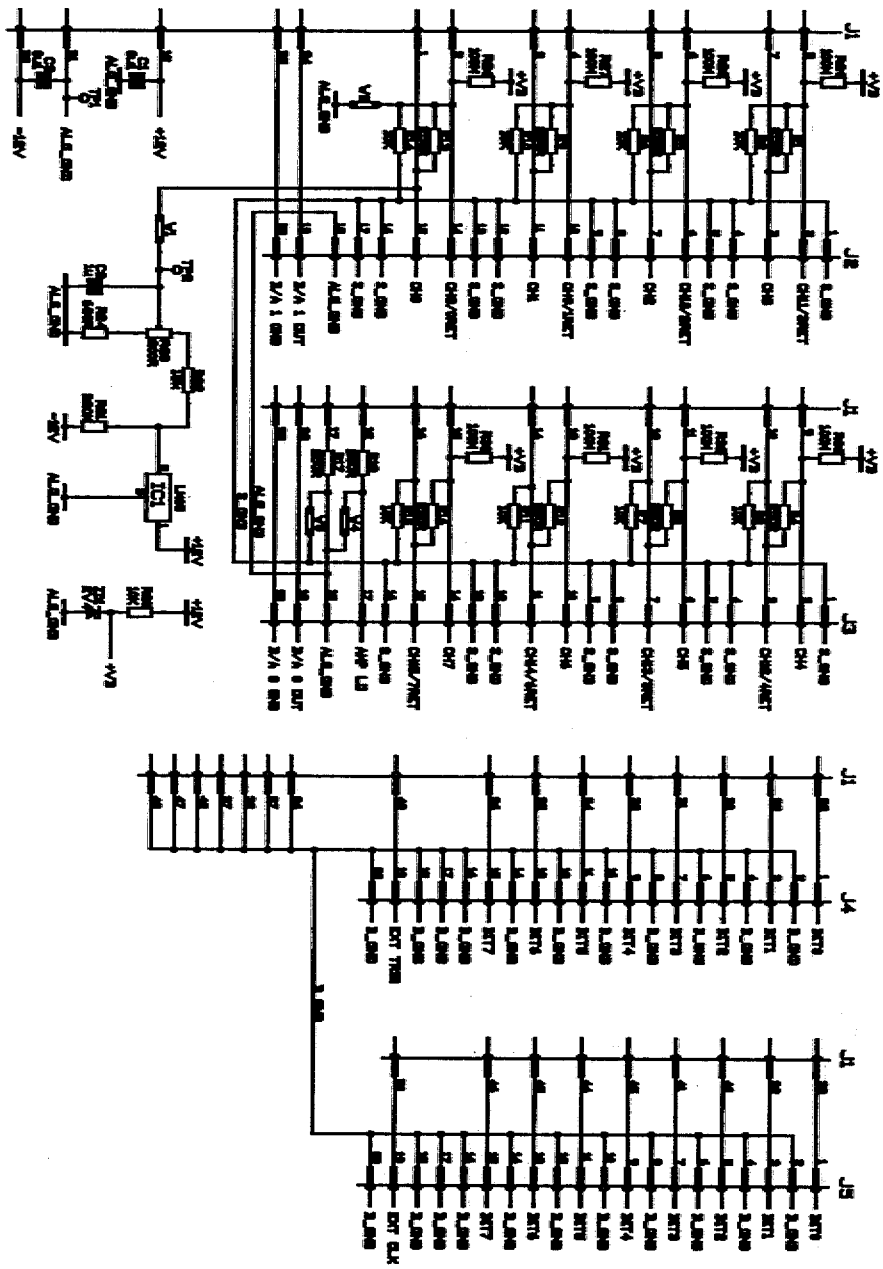
E.4.5.4. Calibration of the cold junction circuit.

This is done as follows :

i. Attach the negative terminal of a voltmeter to TP1 on the PC-77, and the positive lead to TP2.

ii. Adjust R23 such that the voltage between these points is 0.5 mV times the temperature.

Fig. E.1.
PC-77 circuit
diagram.



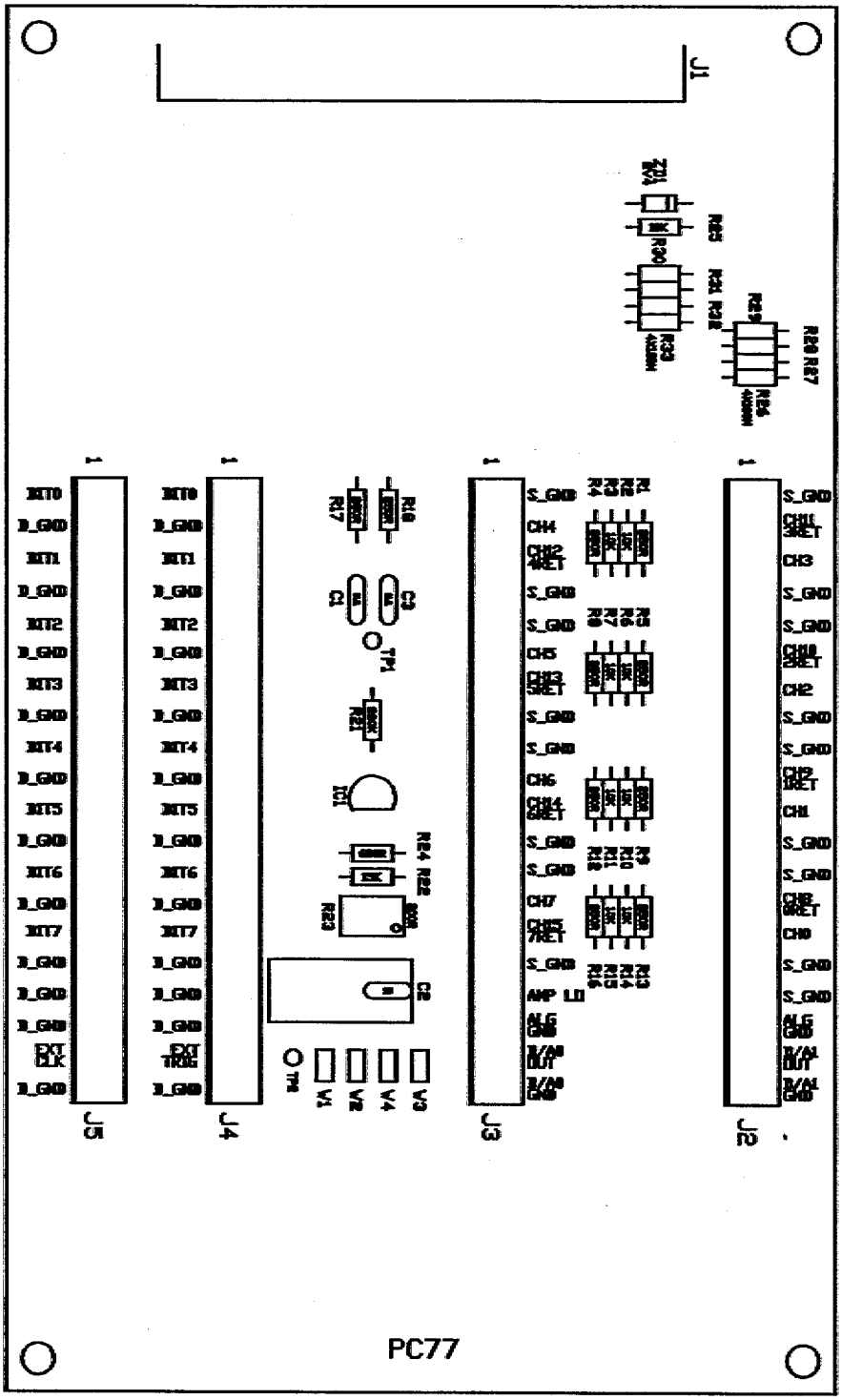


Fig. E.2. PC-77 layout diagram.

[This page intentionally left blank.]

Appendix F

Differences from previous driver versions

F.1. New features

i. Microsoft FORTRAN V5 support.

ii. Microsoft QuickBasic V4.5 support.

iii. Global variable access functions added. The Functions Get_base, Set_base, Get_gain and Set_gain have been added. These allow languages which cannot access the driver's global variables directly to modify the global variables.

F.2. Source code modifications

The driver source code has been modified to compile under either Microsoft C (V5.1), or Turbo C. In order to compile under Turbo C, the symbol "tcc" must be defined.

F.3. Global variables

In previous versions of the driver software, global variables were defined in the main program. This is no longer the case, except in the case of Turbo Pascal. Global variables are now defined in the PC74S module. This change is transparent to C source code which used the data_74 symbol definition to generate the global variable definitions. However, it is necessary to recompile C programs, and to modify programs written in other languages (except Turbo Pascal).

F.4. C calling convention

In previous versions of the compiler, C language versions of the driver used C calling convention. In this version, Pascal calling convention is used. This used to allow driver routines to be placed in IOPL segments under OS/2.

F.5. Language reference

Depending on which language you are using, the following changes must be made to your programs to allow operation with the new driver version.

Microsoft C	Recompile, no source code changes.
Turbo C	Recompile with the "tcc" symbol defined.
Turbo Pascal	Recompile, no source code changes
Other languages	Modify source code to reflect changes in calling convention (if necessary), and change global variables to external.

Index

!

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

V