

ARTISAN®
TECHNOLOGY GROUP

Your **definitive** source for quality pre-owned equipment.

**Artisan Technology Group**
(217) 352-9330 | sales@artisantg.com | artisantg.com

**Full-service, independent repair center**
with experienced engineers and technicians on staff.

**We buy your excess, underutilized, and idle equipment**
along with credit for buybacks and trade-ins.

**Custom engineering**
so your equipment works exactly as you specify.

- Critical and expedited services
- In stock / Ready-to-ship
- Leasing / Rentals / Demos
- ITAR-certified secure asset solutions

**Expert team | Trust guarantee | 100% satisfaction**

All trademarks, brand names, and brands appearing herein are the property of their respective owners.

**Visit our website - *Click HERE***

# PB-ADC3

Optoisolated Analog to Digital Piggyback
for VMOD-2 and IMOD

Manual Order Nr. 14279
**User's Manual**

Issue 3

### Unpacking and Special Handling Instructions

This PepCard product is carefully designed for a long and fault-free life; nonetheless, its life expectancy can be drastically reduced by improper treatment during unpacking and installation.

Observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings, etc. If the product contains batteries for RTC or memory back-up, ensure that the board is not placed on conductive surfaces, including anti-static plastics or sponges. These can cause shorts and damage to the batteries or tracks on the board.

When installing the board, switch off the power mains to the chassis. Do not disconnect the mains as the ground connection prevents the chassis from static voltages, which can damage the board as it is inserted.

Furthermore, do not exceed the specified operational temperature ranges of the board version ordered. If batteries are present, their temperature restrictions must be taken into account.

Keep all of the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the board, re-pack it as it was originally packed.


© 1996 PEP Modular Computer

## REVISION HISTORY
**PB-ADC3 User's Manual**

| Issue | Brief Description of Changes | PCB Index | Date of Issue |
|---|---|---|---|
| 1 | First Issue | 02-01 | January, 1994 |
| 2 | Small corrections throughout manual | 02-01 | Febraury, 1994 |
| 2.0.1 | Corrections to Chapter 4 | 02-01 | March, 1994 |
| 3 | Change to specifications (overvoltage protection) and board index 3 inserted | 03 | July, 1996 |

This document contains proprietary information of **PEP *Modular Computers***. It may not be copied or transmitted by any means, passed to others, or stored in any retrieval system or media, without the prior consent of **PEP *Modular Computers*** or its authorized agents.

The information in this document is, to the best of our knowledge, entirely correct. However, **PEP *Modular Computers*** cannot accept liability for any inaccuracies, or the consequences thereof, nor for any liability arising from the use or application of any circuit, product, or example shown in this document.

**PEP *Modular Computers*** reserve the right to change, modify, or improve this document or the product described herein, as seen fit by **PEP *Modular Computers*** without further notice.


© 1996 PEP Modular Computers

# PEP *Modular Computers*® Two Year Limited Warranty

We grant the original purchaser of **PEP** products the following hardware and system warranty. No other warranties that may be granted or implied by anyone on behalf of **PEP** are valid unless the consumer has the express written consent of **PEP *Modular Computers***.

**PEP *Modular Computers*** warrants their own products (excluding software) to be free from defects in workmanship and materials for a period of 24 consecutive months from the date of purchase. This warranty is not transferable nor extendible to cover any other consumers or long term storage of the product.

This warranty does not cover products which have been modified, altered, or repaired by any other party than **PEP *Modular Computers*** or their authorized agents. Furthermore, any product which has been, or is suspected of being damaged as a result of negligence, misuse, incorrect handling, servicing or maintenance; or has been damaged as a result of excessive current/voltage or temperature; or has had its serial number(s), any other markings, or parts thereof altered, defaced, or removed will also be excluded from this warranty.

A customer who has not excluded his eligibility for this warranty may, in the event of any claim, return the product at the earliest possible convenience, together with a copy of the original proof of purchase, a full description of the application it is used on, and a description of the defect; to the original place of purchase. Pack the product in such a way as to ensure safe transportation (we recommend the original packing materials), whereby **PEP** undertakes to repair or replace any part, assembly or sub-assembly at our discretion; or, to refund the original cost of purchase, if appropriate.

In the event of repair, refund, or replacement of any part, the ownership of the removed or replaced parts reverts to **PEP *Modular Computers***, and the remaining part of the original guarantee, or any new guarantee to cover the repaired or replaced items, will be transferred to cover the new or repaired items. Any extensions to the original guarantee are considered gestures of goodwill, and will be defined in the "Repair Report" returned from **PEP** with the repaired or replaced item.

Other than the repair, replacement, or refund specified above, **PEP *Modular Computers*** will not accept any liability for any further claims which result directly or indirectly from any warranty claim. We specifically exclude any claim for damage to any system or process in which the product was employed, or any loss incurred as a result of the product not functioning at any given time. The extent of **PEP *Modular Computers*** liability to the customer shall not be greater than the original purchase price of the item for which any claim exists.

**PEP *Modular Computers*** makes no warranty or representation, either express or implied, with respect to its products, reliability, fitness, quality, marketability or ability to fulfill any particular application or purpose. As a result, the products are sold "as is," and the responsibility to ensure their suitability for any given task remains the purchaser's.

In no event will **PEP** be liable for direct, indirect, or consequential damages resulting from the use of our hardware or software products, or documentation; even if we were advised of the possibility of such claims prior to the purchase of, or during any period since the purchase of the product.

Please remember that no **PEP *Modular Computers*** employee, dealer, or agent are authorized to make any modification or addition to the above terms, either verbally or in any other form written or electronically transmitted, without consent.


© 1996 PEP Modular Computer

## TABLE OF CONTENTS

Page

**1. Introduction . . . . . 1-1**
1.1 Product Overview . . . . . 1
1.2 Ordering Information . . . . . 1
1.3 Specifications . . . . . 2
1.4 Board Overview . . . . . 3
1.5 Features . . . . . 4
1.6 Related Publications . . . . . 4

**2. Functional Description . . . . . 2-1**
*Figure 2.0.0.1: PB-ADC3 Block Diagram . . . . . 1*
2.1 Analog to Digital Conversion . . . . . 2
*Figure 2.1.0.1: PB-ADC3 Input Circuitry . . . . . 2*
2.2 EEPROM Data Calibration . . . . . 4
*Figure 2.2.0.1: PB-ADC3 Characteristics (Bipolar) . . . . . 4*
2.3 Logic Interface . . . . . 5

**3. Configuration . . . . . 3-1**
*Figure 3.0.0.1: PB-ADC3 Jumper Layout (Solder Side) . . . . . 1*
*Table 3.0.0.2: PB-ADC3 Default Jumper Settings . . . . . 1*
3.1 Jumpers BP7-BP0, BM7-BM0: Input Selection . . . . . 2
3.2 Jumper B51: Offset Voltage Selection . . . . . 3
3.3 VMOD-2/IMOD Jumper Configuration . . . . . 3

**4. Programming . . . . . 4-1**
4.1 PB-ADC3 Address Map . . . . . 1
4.1.1 ADC Communications Register ($00) . . . . . 1
*Figure 4.1.1.1: Read Out of One Channel . . . . . 2*
4.1.2 EEPROM Communications Register ($10) . . . . . 3
4.1.3 EEPROM Program Register ($20) . . . . . 5
4.1.4 Status Register ($31) . . . . . 6
4.1.5 ID Register ($7F) . . . . . 6
4.2 Analog Input Correction . . . . . 7
*Table 4.2.0.1: 5V Range Parameters . . . . . 7*
*Table 4.2.0.2: 10V Range Parameters . . . . . 7*
*Table 4.2.0.3: 20mA Range Parameters . . . . . 7*
4.3 EEPROM Data Structure . . . . . 8

**5. Pinouts . . . . . 5-1**
5.1 Main Board . . . . . 1
*Figure 5.1.0.1: Board Connector Overview . . . . . 1*
5.1.1 ST100 Connector . . . . . 1
5.1.2 ST101 Connector . . . . . 2
5.1.3 ST102 Connector . . . . . 2
5.2 VMOD-2/IMOD Front Panel . . . . . 3
*Figure 5.2.0.1: VMOD-2/IMOD Front Panel . . . . . 3*
5.3 VMOD-2D Front Panel . . . . . 5
*Figure 5.3.0.1: VMOD-2D Front Panel . . . . . 5*
5.4 VMOD-2 / VMOD-2D Pinout Relationship . . . . . 6

**6. Installation . . . . . 6-1**
6.1 VMEbus Connection . . . . . 1
*Figure 6.1.0.1: The VMEbus Backplane . . . . . 1*
6.2 Installing the PB-ADC3 . . . . . 2
*Figure 6.2.0.1: PB-ADC3 Installation Overview . . . . . 2*
6.3 General Notes for Using the System . . . . . 3

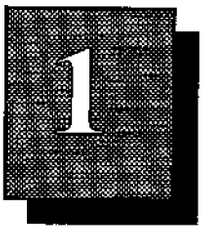
 © 1996 PEP Modular Computers 

**Appendix A. Complex Examples** . . . . . **A-1**
A.1 Read Channels . . . . . 1
A.1.1 C-Program Listing . . . . . 1
A.1.2 Assembler Program . . . . . 4
A.2 Converted Data Correction . . . . . 5


© 1996 PEP Modular Computer

*This page has been intentionally left blank*

 © 1996 PEP Modular Computers 



# 1. INTRODUCTION

## 1.1 Product Overview

The PB-ADC3 is an opto-isolated 8 channel A/D differential input piggyback for the VMOD-2 and IMOD. The input channels are differential with 12-bit resolution and galvanically isolated from the system's 500V DC supply. Either unipolar or bipolar conversion can be programmed, as can the end of conversion interrupt. The range of the inputs is from 0V to 5V, ±5V and 0V to 10V, ±10V. The input range for the current version is from 0-20mA.

## 1.2 Ordering Information

| Name | Description | Order Number | |
|---|---|---|---|
| PB-ADC3 | 8 channel, 12-bit resolution, ±5V, ±10V voltage input, serial EEPROM with calibration data | 5230-36 | *Old* |
| | | 3128 | *New* |
| PB-ADC3 | 8 channel, 12-bit resolution, 0-20mA current input, serial EEPROM with calibration data | 5230-36/1 | *Old* |
| | | 3129 | *New* |


© 1996 PEP Modular Computers

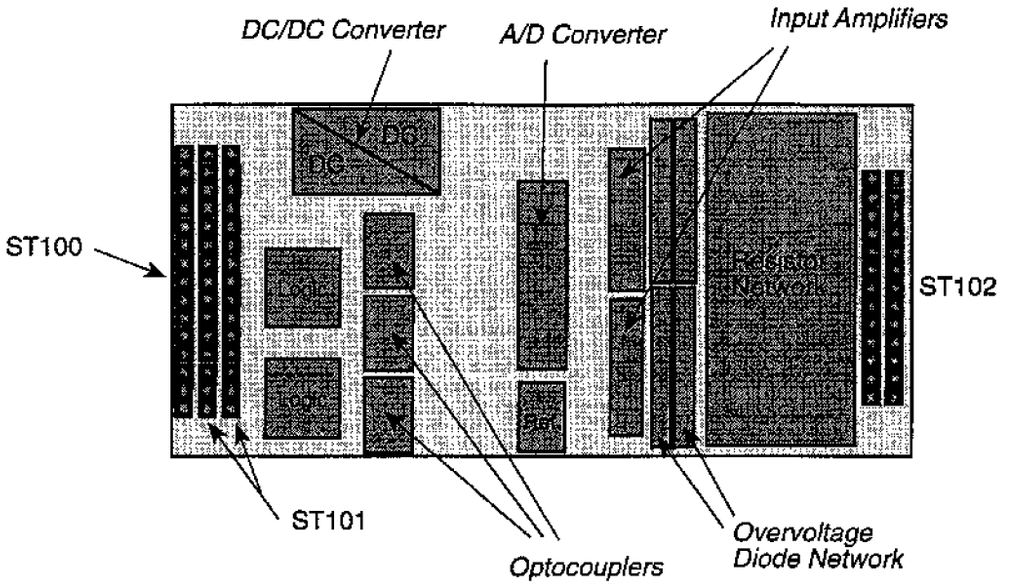
## 1.3 Specifications

| | |
|---|---|
| A/D converter chip | LTC1290DCN |
| Number of channels | 8 differential |
| Resolution | 12-bit |
| Conversion time | 43µs |
| Throughput rate | 20KHz |
| Linearity error | ± 3/4 LSB |
| Differential linearity | Monotonic over temperature |
| Voltage ranges:<br>Unipolar<br>Bipolar | <br>0-5V, 0-10V<br>±5V, ±10V |
| Current range | 0-20mA |
| Overvoltage protection:<br>Voltage version<br>Current version | <br>±35V continuous<br>±8V continuous, equivalent to 32mA |
| Input impedance:<br>5 Volt range<br>10 Volt range<br>20mA current | <br>20KΩ<br>40 KΩ<br>246Ω |
| EEPROM | 93C46, 128 byte calibration data |
| Galvanic isolation from the system | 500V DC system/process |
| ID Byte | $EB |
| Temperature range<br>Standard<br>Extended | <br>0...+70°C<br>-40...+85°C |
| Power requirement | @5V : 235mA Typical |
| Temperature drift | Typ. 1LSB at 10V, bipolar over temperature range |

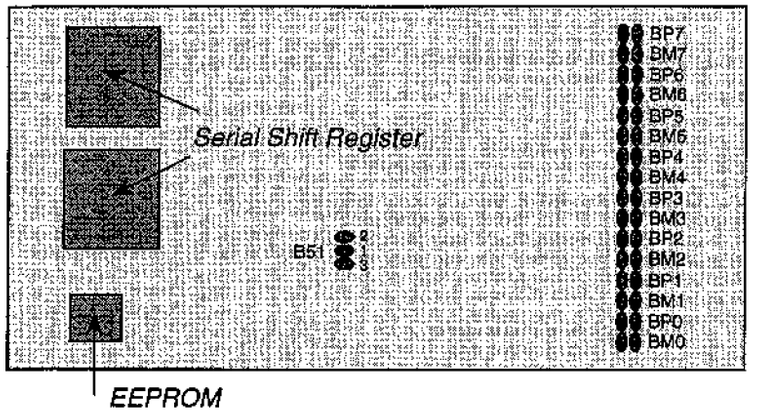

© 1996 PEP Modular Computers

## 1.4 Board Overview

*Component Side*




*Solder Side*




 © 1994 PEP Modular Computers 

## 1.5 Features

- *Eight analog channels*
- *12-bit resolution*
- *Unipolar or bipolar conversion programmable*
- *43µs conversion time*
- *500 volt DC galvanic isolation*
- *20 KHz throughput rate*
- *0-10/±10V, 0-5/±5V, 0-20 mA*
- *Optoisolated*

## 1.6 Related Publications

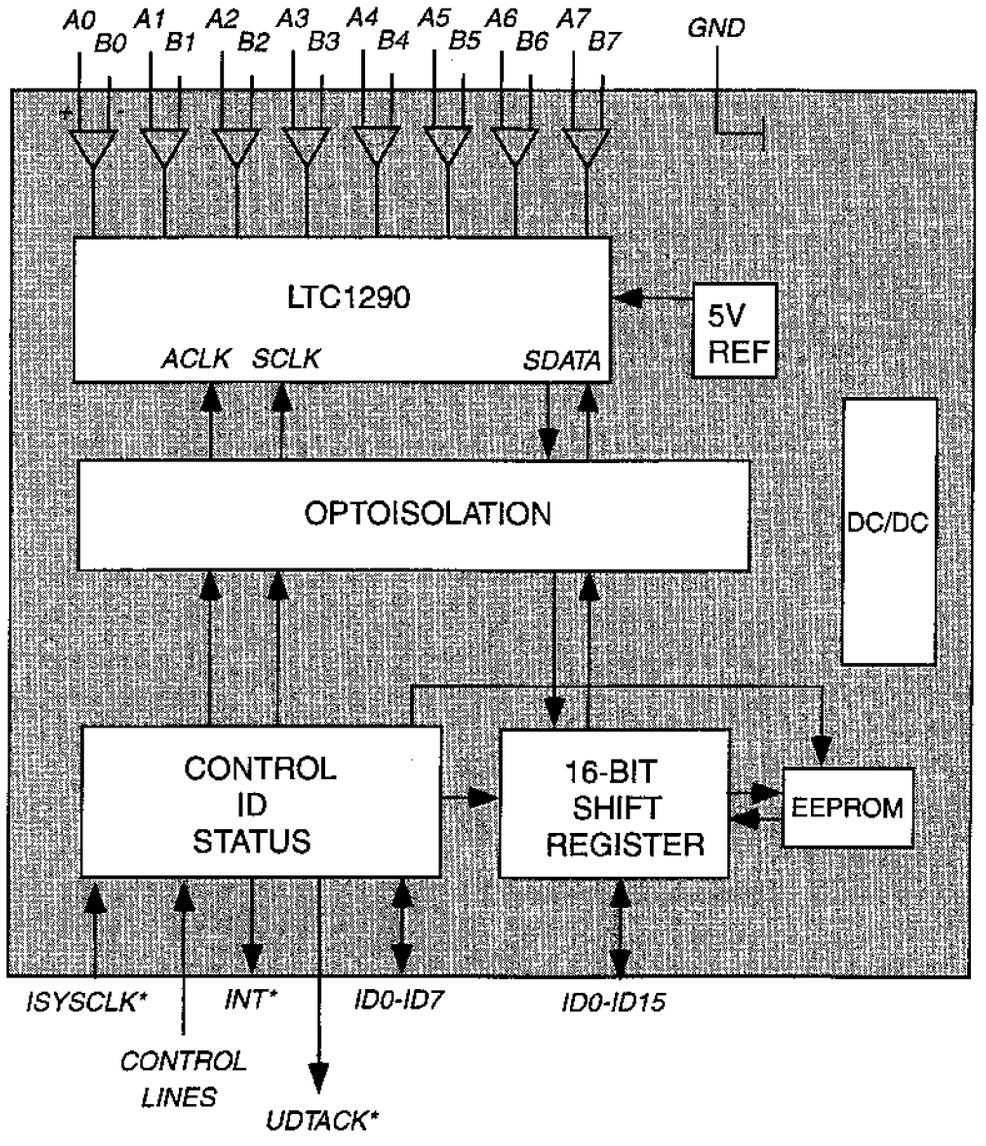
VMEbus Specifications Revision C1
Data Sheet for the LTC290DCN from Linear Technology
Data Sheet for the 93C46 EEPROM from Microchip/National Semiconductors

 © 1994 PEP Modular Computers 

# 2. FUNCTIONAL DESCRIPTION

This chapter describes the functionality of the main blocks of the PB-ADC3 piggyback.

**Figure 2.0.0.1: PB-ADC3 Block Diagram**

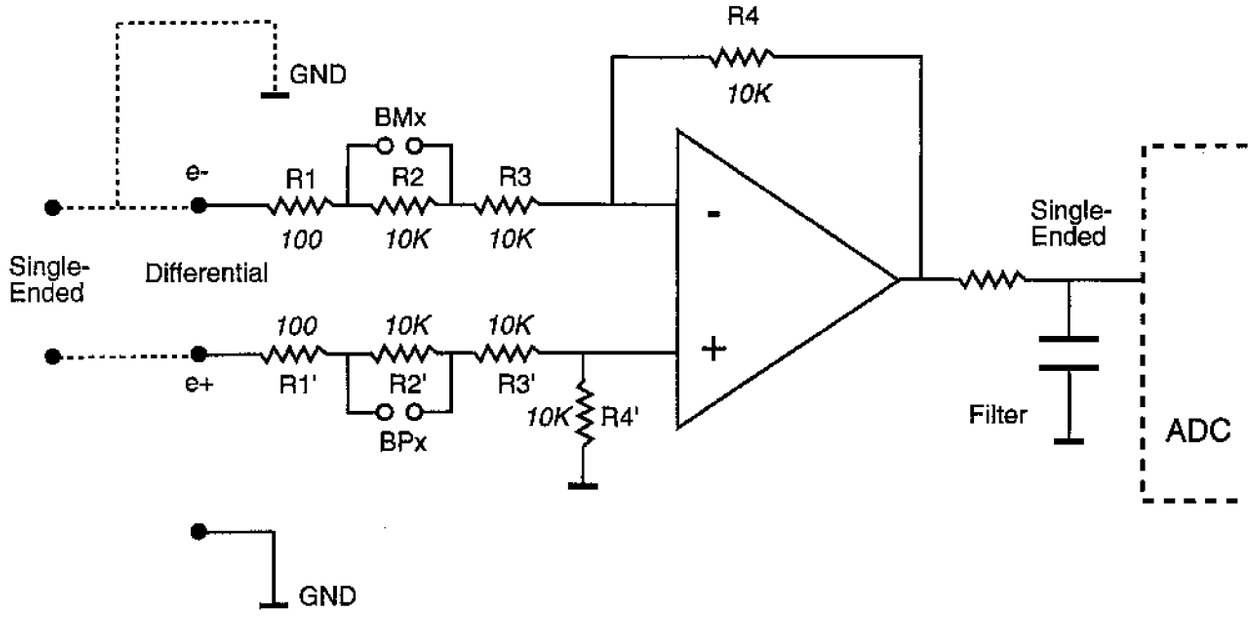



 © 1994 PEP Modulär Computers 

## 2.1 Analog to Digital Conversion

The eight multiplexed analog inputs of the LTC1290 are configured as single-ended inputs. These inputs perform either a 12-bit unipolar or an 11-bit plus sign bit bipolar conversion. To increase the input range of ±5V to ±10V with differential functionality the input circuitry in Figure 2.1.0.1 is used.

**Figure 2.1.0.1: PB-ADC3 Input Circuitry**




$$G = \frac{R4}{R1 + R2 + R3} \qquad \text{where } Rn = Rn';\quad n = 1\text{-}4;$$

*With BPx and BMx open, G=0.4975*

*With BPx and BMx set, G=0.9901*

The additional input resistors $R_1$ and $R_1$' help the input gain to be always less than 1 or less than 0.5. This compensates any critical gain error. The real gain error for each channel is stored in the EEPROM.

The data exchange across the optoisolation barrier is controlled by the serial interface. Each time a 16-bit data word is sent to the LTC1290, the last 16-bit conversion result is simultaneously returned. For the 1MHz SCLK, a 16-bit exchange takes 16 µsec.

The first four incoming bits to the LTC1290 contain the channel number for the multiplexing unit, together with the conversion mode, unipolar or bipolar. The remaining 12-bit shift cycles are used for the 12-bit output data. During this time, the analog input channel is sampled. In the next step the conversion is started (*see the LTC1290 Data Sheet*).

The data conversion concept of the LTC1290 uses a successive approximation method. For each of the 12 approximation steps, 4 ACLK cycles are necessary. Therefore the pure approximation time for the 2MHz ACLK is 24µsec.

 © 1994 PEP Modular Computers 

As well as the serial transmission time and the approximation time, the logic requires 3µsec in order to synchronize. In total, therefore, one conversion needs a maximum of 43µsec to be completed. After this time the next conversion cycle can be started.

**Note**

Due to the data exchange mechanism, one more conversion has always to be started than is actually needed.

An optional feature of the PB-ADC3 is the initial offset. All the input channels can be offsetted by approx. 7.5mV using the B51 jumper. This is especially useful for unipolar measurement, as a negative offset cannot be measured with unipolar conversion.

The readout values of the analog inputs depend on the reference voltage, Vref, which is 5V. The resultant values of the analog inputs are 16-bit values. For a bipolar conversion, a negative result is returned as a negative value in which the leading bits are already extended to 1's.

All the above points are also relevant for the current input version of the PB-ADC3. The only difference is that an additional input resistor is present that transforms the 0-20mA input current into a voltage.

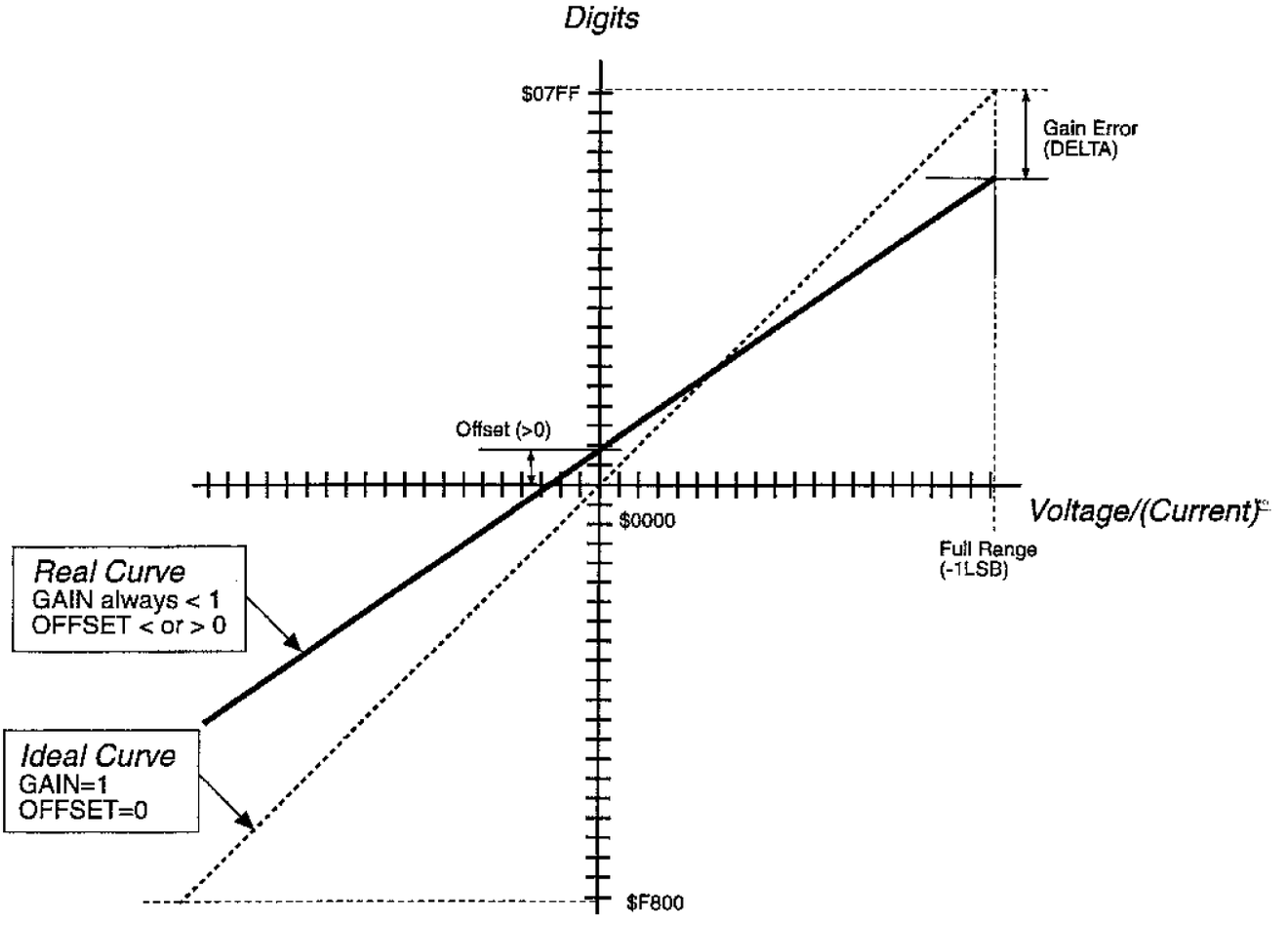
*For more information on PB-ADC3 jumper settings, please see the Configuration chapter in this manual.*


© 1994 PEP Modular Computers

## 2.2 EEPROM Data Calibration

The Figure below shows the ideal and real characteristics of the PB-ADC3. Both the slope and offset parameters are programmed into the hardware, being measured and stored in the EEPROM during the board production phase. As well as the standard offset, the adjustable offset is also stored in the EEPROM.

**Figure 2.2.0.1: PB-ADC3 Characteristics (Bipolar)**




The slope of the curve is measured in the form of the *Gain Error* for both the 5 volt and 10 volt range and stored in the EEPROM.

In order to calculate the corrected value, the offset is first subtracted from the actual value and then the slope is multiplied with the correction factor, as shown below.

*Corrected Value = (Actual Value - Offset Value) * (1 + ε) * 1LSB*

where:

*ε = Gain Error/ (Full Range - Gain Error)*

| | | | |
|---|---|---|---|
| 1LSB = 1.22mV | (*5V Unipolar*) | 1LSB = 2.44mV | (*5V Bipolar*) |
| 1LSB = 2.44mV | (*10V Unipolar*) | 1LSB = 4.88mV | (*10V Bipolar*) |
| 1LSB = 4.88µA | (*20mA Unipolar*) | 1LSB = 9.76µA | (*20mA Bipolar*) |

 © 1994 PEP Modular Computers 

## 2.3 Logic Interface

The logic interface, together with both shift registers, form the serial interface to the LTC1290 and the EEPROM. The shift register frequency is set to 1MHz for both registers. The data transfer is made up of 16 shifts for the LTC1290 and 25 shifts for the EEPROM.

As well as serial interface tasks, the status register is also realized using the PAL logic. Its tasks include the EOC (End Of Conversion) timing and interrupt handling.

Another important task performed by the logic interface is the generation of an ID byte. This built-in test feature allows an interrogation of the VMOD-2/IMOD, supplying an ID for each of the fitted piggybacks. If this is integrated into the application software, it can be used to check that any given tasks are valid for the fitted piggyback. Offsets $7F (for location A) and $FF (for location B), allow a software check of which piggybacks are fitted.

The PB-ADC3 has a "$EB" Byte.

Other ID Bytes are:

| | | |
|---|---|---|
| *$EE* | *PB-BIT* | *BITBUS communications controller* |
| *$EF* | *PB-DIO4* | *Digital I/O piggyback (high voltage)* |
| *$F0* | *PB-CNT* | *Counter piggyback* |
| *$F1* | *PB-DAC/2* | *D to A converter piggyback* |
| *$EA* | *PB-DAC3* | *D to A output piggyback* |
| *$F2* | *PB-DIO* | *Digital I/O piggyback* |
| *$F3* | *PB-DIN/3* | *Digital input piggyback* |
| *$F4* | *PB-ADC/2* | *A to D converter piggyback* |
| *$EB* | *PB-ADC3* | *A to D input piggyback* |
| *$F5* | *PB-CIO/2* | *Counter/I/O piggyback* |
| *$F7* | *PB-SIO4* | *Quad serial piggyback RS232* |
| *$E7* | *PB-SIO4A* | *Quad serial piggyback RS422/RS485* |
| *$F8* | *PB-DOUT* | *12 channel high voltage digital output* |
| *$ED* | *PB-DOUT2* | *16 channel high voltage digital output* |
| *$F9* | *PB-DIN2* | *Digital input piggyback* |
| *$FB* | *PB-DIO-2* | *Digital I/O piggyback* |
| *$FC* | *PB-REL* | *Relay piggyback* |
| *$FD* | *PB-DIO-3* | *Digital I/O piggyback* |
| *$FE* | *PB-STP* | *Stepper motor controller piggyback* |

 © 1994 PEP Modular Computers 

*This page has been intentionally left blank*

 © 1994 PEP Modular Computers 

# 3. CONFIGURATION

The PB-ADC3 has 8 pairs of solder jumpers on the solder side of the board, one pair for each channel. These allow the voltage distributor to be set for full or half operation. The 3-pad solder jumper, B51 sets the offset voltage for all inputs (to be used only in Unipolar Mode).

**Figure 3.0.0.1: PB-ADC3 Jumper Layout (Solder Side)**

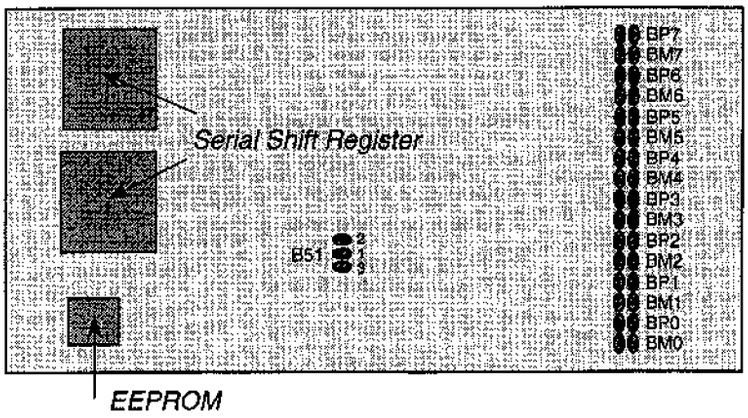



**Table 3.0.0.2: PB-ADC3 Default Jumper Settings**

*Voltage Version:*

| Channel | Jumper | Default Setting | Function |
|---|---|---|---|
| 0 | BP0, BM0 | open | 0-10V and ±10V input |
| 1 | BP1, BM1 | open | 0-10V and ±10V input |
| 2 | BP2, BM2 | open | 0-10V and ±10V input |
| 3 | BP3, BM3 | open | 0-10V and ±10V input |
| 4 | BP4, BM4 | open | 0-10V and ±10V input |
| 5 | BP5, BM5 | open | 0-10V and ±10V input |
| 6 | BP6, BM6 | open | 0-10V and ±10V input |
| 7 | BP7, BM7 | open | 0-10V and ±10V input |
| | B51 | 1-3 | No additional offset on all inputs |

*Current Version:*

| Channel | Jumper | Default Setting | Function |
|---|---|---|---|
| 0 | BP0, BM0 | set | 0-20mA input |
| 1 | BP1, BM1 | set | 0-20mA input |
| 2 | BP2, BM2 | set | 0-20mA input |
| 3 | BP3, BM3 | set | 0-20mA input |
| 4 | BP4, BM4 | set | 0-20mA input |
| 5 | BP5, BM5 | set | 0-20mA input |
| 6 | BP6, BM6 | set | 0-20mA input |
| 7 | BP7, BM7 | set | 0-20mA input |
| | B51 | 1-3 | No additional offset on all inputs |


© 1994 PEP Modular Computers

Default settings are shown in *italic* in the following sections.

## 3.1 Jumpers BP7-BP0, BM7-BM0: Input Selection

*Voltage Version:*

| Channel | Jumper | Setting | Function |
|---|---|---|---|
| 0 | BP0, BM0 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 1 | BP1, BM1 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 2 | BP2, BM2 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 3 | BP3, BM3 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 4 | BP4, BM4 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 5 | BP5, BM5 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 6 | BP6, BM6 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |
| 7 | BP7, BM7 | *open* | *0-10V and ±10V input* |
| | | set | 0-5V and ±5V |

*Current Version:*

Only one jumper setting is available for the current input version of the PB-ADC3.

| Channel | Jumper | Setting | Function |
|---|---|---|---|
| 0 | BP0, BM0 | *set* | *0-20mA input* |
| 1 | BP1, BM1 | *set* | *0-20mA input* |
| 2 | BP2, BM2 | *set* | *0-20mA input* |
| 3 | BP3, BM3 | *set* | *0-20mA input* |
| 4 | BP4, BM4 | *set* | *0-20mA input* |
| 5 | BP5, BM5 | *set* | *0-20mA input* |
| 6 | BP6, BM6 | *set* | *0-20mA input* |
| 7 | BP7, BM7 | *set* | *0-20mA input* |


©1994 PEP Modular Computers

## 3.2 Jumper B51: Offset Voltage Selection

This jumper can be configured to set a voltage offset, approx 7.5mV, on all channels. This jumper is used for both voltage and current versions of the PB-ADC3.

*Voltage and Current Versions:*

| Jumper | Setting | Function |
|---|---|---|
| B51 | *1-3* | *Offset 0V on all inputs* |
| | 1-2 | Offset approx. 7.5mV on all inputs |

The real measured value of the offset is stored in the EEPROM.

*For more details on offset voltage selection, please refer to section 2.1 in this manual.*

## 3.3 VMOD-2/IMOD Jumper Configuration

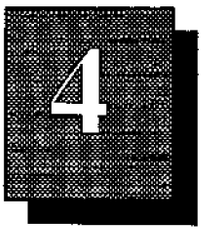
In order for the PB-ADC3 to be put into use on either the VMOD-2 or IMOD boards, the IRQ and Base Address jumper settings on these boards must be correctly configured.

*For more details on these jumper configurations, please refer to the VMOD-2 or IMOD user manuals.*


© 1994 PEP Modular Computers

*This page has been intentionally left blank*

 ©1994 PEP Modular Computers 



# 4. PROGRAMMING

The base address of the PB-ADC3 in the upper piggyback position (location A) is the same as that set for the VMOD-2/IMOD used (default: $87FE2400/$F70000), whereas the base address in the lower piggyback position (location B) is +$80 of that set for the VMOD-2/IMOD (default: $87FE2480/$F70080).

## 4.1 PB-ADC3 Address Map

| Address | Byte/Word | Read/Write | Function |
|---|---|---|---|
| BASE+$00 | Word | Read/Write | ADC communications register |
| BASE+$10 | Word | Read/Write | EEPROM communications register |
| BASE+$20 | Word | Write | EEPROM programming register |
| BASE+$31 | Byte | Read/Write | Status register |
| BASE+$7F | Byte | Read | ID register |

### 4.1.1 ADC Communications Register ($00)

This register is the serial interface register for data transfer to and from the LTC converter chip. The next conversion of the selected channel is started with a word write access to this register. The channel number, the bipolar or unipolar selection and the data output format of the converted value all have to be set.

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| BASE+$00 | *not used* | | | | | | | | *data format* | | | *mode* | *channel* | | | |

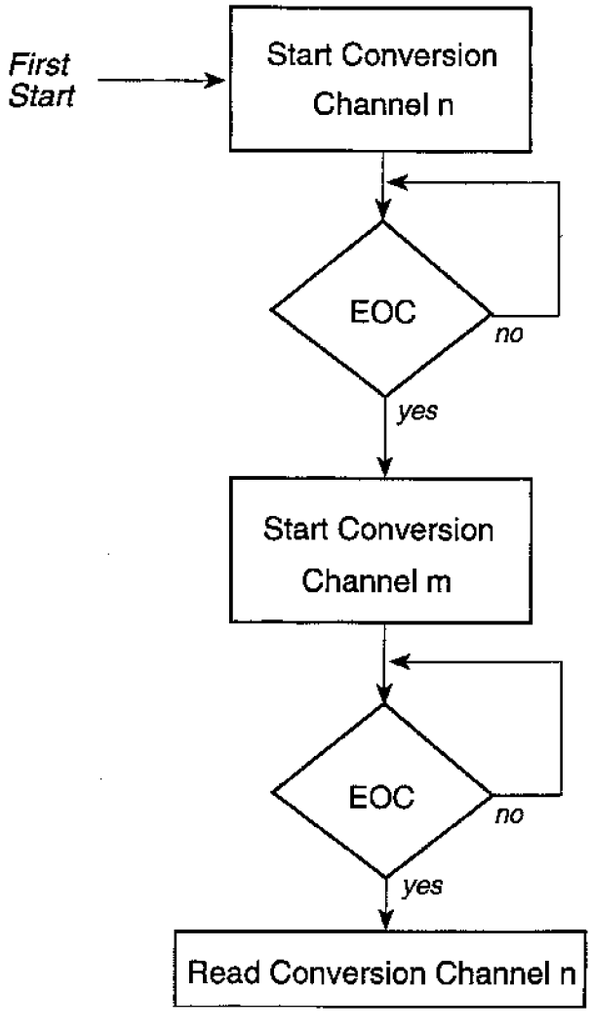
**Register Description**

| Name | Value | Description |
|---|---|---|
| **data format**<br>*bits 7-5* | %110 | 16-bit, LSB first. Other mode of the LTC1290 must not be used on the PB-ADC3. |
| **mode**<br>*bit 4* | 0<br>1 | Bipolar<br>Unipolar |
| **channel**<br>*bits 3-0* | %1111<br>%1101<br>%0111<br>%0101<br>%1011<br>%1001<br>%0011<br>%0001<br>%Rest | Channel 0<br>Channel 1<br>Channel 2<br>Channel 3<br>Channel 4<br>Channel 5<br>Channel 6<br>Channel 7<br>Not allowed |
| **not used**<br>*bits 15-8* | 1 or 0 | |


© 1994 PEP Modular Computers

With each 16 bit data transmission to the LTC converter a 16 bit data word is returned from the previous conversion. This means that a new conversion has to be started in order to get out the actual result.

**Figure 4.1.1.1: Read Out of One Channel**




**Example**

```
                .
                .
                .
wait:           btst.b          #2,BASE+$31         EOC?
                bne.s           wait
                move.w          #$CF,BASE+$00       convert channel 0, bipolar
wait2:          btst.b          #2,BASE+$31         EOC?
                bne.s           wait2
                move.w          #$9,BASE+$00        convert channel 5, bipolar
wait3:          bfst.b          #2,BASE+$31         EOC?
                bne.s           wait3
                move.w          BASE+$00,d0         get channel 0 conversion result
                .
                .
                .
```

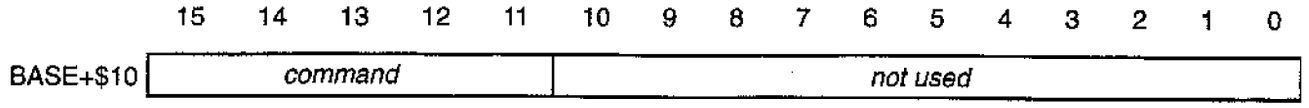
 © 1994 PEP Modular Computers 

### 4.1.2 EEPROM Communications Register ($10)

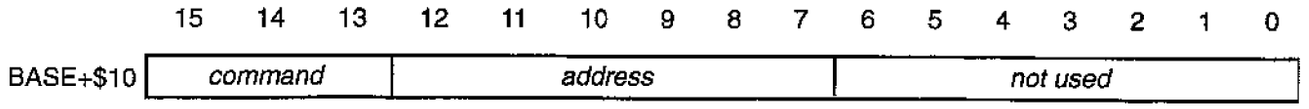
This register handles the data exchange with the EEPROM. This requires a 'WORD' write and read access. Before reading a particular address in the EEPROM, a *read* command has to be written to the required address on this register.

**Write** (*Two Cases*):

*1) Write for EN/DIS protection command*




*2) Write for readout command*

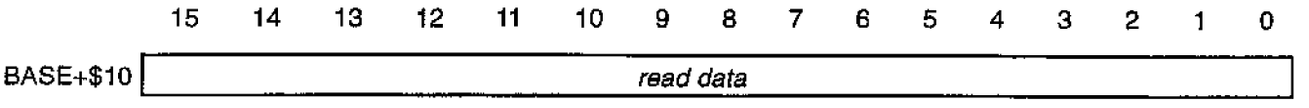



**Register Description**

| Name | Value | Description |
|---|---|---|
| **command**<br>*bits 15-11*<br>*or*<br>*bits 15-13* | %10011<br>%10000<br><br>%110 | Disable program protection<br>Enable program protection<br><br>Read stored data |
| **address**<br>*bits 12-7* | <br><br>0-63 | These bits can only be interpreted with a read command<br>Address 0-63 (dec) |
| **not used**<br>*bits 10-0*<br>*or*<br>*bits 6-0* | Free | |

**Read:**

After a *read* command, it can be determined using the status bit EOC, when the expected data can be read (WORD).





© 1994 PEP Modular Computers

**Example 1**

```
                .
                .
                .
wait:       btst.b    #2,BASE+$31        EOC?
            bne.s     wait
            move.w    #$C000,BASE+$10    read command for EPROM address offset 0
wait2:      btst.b    #2,BASE+$31        EOC?
            bne.s     wait2
            move.w    BASE+$10,d0        get EEPROM read result
                .
                .
                .
```

*Address offset 1 -> read command = $C080*
*Address offset 63 -> read command = $DF80*

**Example 2**

```
                .
                .
                .
wait:       btst.b    #2,BASE+$31        EOC?
            bne.s     wait
            move.w    #$9800,BASE+$10    set EEPROM in programming mode
                .
                .

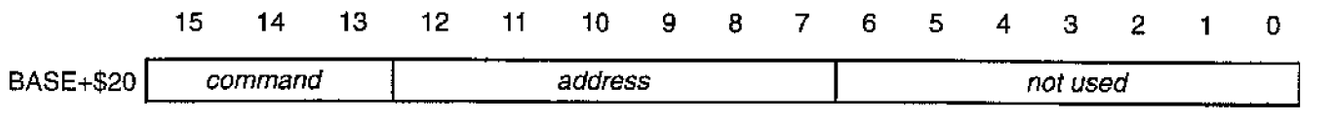
wait2:      btst.b    #2,BASE+$31        EOC?
            bne.s     wait2
            move.w    #$8000,BASE+$10    set EEPROM in protected mode
                .
                .
                .
```


© 1994 PEP Modular Computers

### 4.1.3 EEPROM Programming Register ($20)

This register is used for data exchange in the programming mode. This requires two consecutive word write accesses to the register. The first access sends the *write* command and the address that is to be overwritten.

**First value write:**

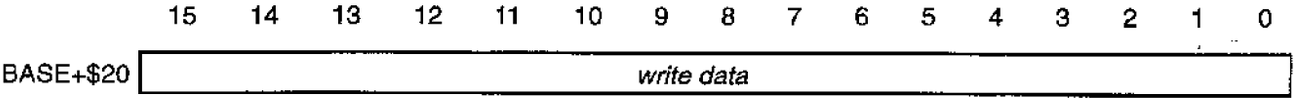



**Register Description**

| Name | Value | Description |
|---|---|---|
| **command** *bits 15-13* | %101 | Write the following data |
| **address** *bits 12-7* | 0-63 | Address 0 to 63 (dec) |
| **not used** *bits 6-0* | Free | |

**Second value write:**

After a *write* command, it must be determined using the status bit EOS, when the next write access can take place.




After the data has been transferred, the start of the program can be determined with the status register's EOC flag. The PB-ADC3/EEPROM can only be accessed again after the EOC flag has reverted to 0 **and** a programming time of at least 5ms has passed.

**Example**

```
                .
                .
wait:     btst.b    #2,BASE+$31        EOS?
          bne.s     wait
          move.w    #$B000,BASE+$20    write command to first user location
wait2:    btst.b    #2,BASE+$31        EOS?
          bne.s     wait2
          move.w    #$1234,BASE+$20    program $1234
wait3:    btst.b    #4,BASE+$31        EOS?
          bne.s     wait3
then      wait at least 5ms
```

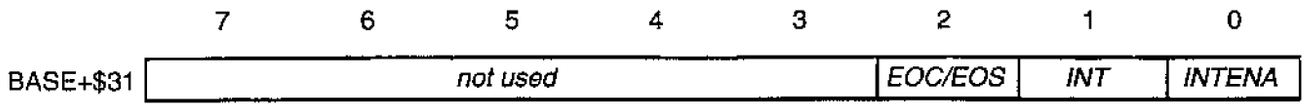

© 1994 PEP Modular Computers

**Note**

Due to the fact that the EEPROM contains calibration data programmed by PEP, it is not recommended that data be written to the EEPROM, even in free locations.

### 4.1.4 Status Register ($31)

The status register can be read and written to in byte access. This returns four pieces of information; EOC/EOS (*End Of Conversion/End Of Shift*), INT and INTENA.




**Register Description**

| Name | Value | Description | | Access |
|---|---|---|---|---|
| **not used** *bits 7-3* | 1 | Basic setting | | |
| **EOC/EOS** *bit 2* | 0<br>1 | End of conversion (default)<br>Conversion active | End of shift (default)<br>Shift busy | *Read only |
| **INT** *bit 1* | 0<br>1 | PB-ADC3 IRQ line active<br>No PB-ADC3 IRQ | | *Read only |
| **INTENA** *bit 0* | 0<br>1 | IRQ disabled<br>IRQ enabled | | Read/Write |

* a write access to this bit has no influence

**Interrupt Handling**

The interrupt is activated as soon as "1" is set on the *INTENA* bit or, in other words, if the *EOC* flag is "0" the IRQ routine starts immediately.

If a conversion was started immediately before setting an *INTENA* bit, the *EOC* flag will already be "1" and the first IRQ will occur after the end of the *EOC*.

Taking away an *INT* signal at the end of an IRQ routine is possible by starting the next conversion or *INTENA* bit clear.

**Note**

If using interrupts, the IRQ vector is set on the VMOD-2 or IMOD.

### 4.1.5 ID Register ($7F)

This register contains the ID byte, allowing automatic recognition of the piggyback fitted to the VMOD-2 or IMOD. This feature is particulary useful in a system with several piggybacks fitted. The ID byte is read using a byte read access.

PB-ADC3 ID byte = $EB

*For more information on the ID byte, please see the ID Byte section in the Functional Description chapter.*

 © 1994 PEP Modular Computers 

## 4.2 Analog Input Correction

In order to calculate the accurate result from the actual value, the following formula is used (see also the *Functional Description* chapter in this manual):

*Corrected Value = (Actual Value - Offset Value) * (1 + ε) * 1LSB*

where:

*ε = Gain Error/ (Full Range - Gain Error)*

The values of the above parameters are extracted from the EEPROM (described in the next Section) and are shown in the following Tables for the available voltage and current ranges.

**Table 4.2.0.1: 5V Range Parameters**

| | Conversion | |
|---|---|---|
| **Parameter** | **Unipolar** | **Bipolar** |
| Actual Value | 0 - $FFF | $F800 - 0 - $07FF |
| Offset Value | Offset EEPROM | Offset EEPROM / 2 |
| LSB | 5V / 4096 = 1.22mV | 10V / 4096 = 2.44mV |
| ε | 5V range gain error / ($FFF - 5V range gain error) | (5V range gain error / ($FFF - 5V range gain error))/2 |

**Table 4.2.0.2: 10V Range Parameters**

| | Conversion | |
|---|---|---|
| **Parameter** | **Unipolar** | **Bipolar** |
| Actual Value | 0 - $FFF | $F800 - 0 - $07FF |
| Offset Value | Offset EEPROM | Offset EEPROM / 2 |
| LSB | 10V / 4096 = 2.44mV | 20V / 4096 = 4.88mV |
| ε | 10V range gain error / ($FFF - 10V range gain error) | (10V range gain error / ($FFF - 10V range gain error))/2 |

**Table 4.2.0.3: 20mA Range Parameters**

| | Conversion | |
|---|---|---|
| **Parameter** | **Unipolar** | **Bipolar** |
| Actual Value | 0 - $FFF | $F800 - 0 - $07FF |
| Offset Value | Offset EEPROM | Offset EEPROM / 2 |
| LSB | 20mA / 4096 = 4.88µA | 40mA / 4096 = 9.76 µA |
| ε | 20mA range gain error / ($FFF - 20mA range gain error) | (20mA range gain error / ($FFF - 20mA range gain error))/2 |

**Note**

The above Tables assume that no offset is used on the analog inputs (Jumper B51 default setting).

*See Appendix A.2 for a programming example using optimized INTEGER ARITHMETIC.*

 © 1994 PEP Modular Computers 

## 4.3 EEPROM Data Structure

The EEPROM 93C46 provides a 64-word address, word access, permanently programmable memory. The address assignments to the individual analog outputs are shown below. Note that the address counter of the Table is **word** orientated.

| Address* | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $0 | *additional offset for unipolar mode* | | | | | | | | *channel* | | | | *offset* | | | | channel 0 |
| $1 | *10V range gain error (reserved)* | | | | | | | | *5V range (20mA range) gain error* | | | | | | | | |
| $2 | *additional offset for unipolar mode* | | | | | | | | *channel* | | | | *offset* | | | | channel 1 |
| $3 | *10V range gain error (reserved)* | | | | | | | | *5V range (20mA range) gain error* | | | | | | | | |
| | . | | | | | | | | | | | | . | | | | |
| | . | | | | | | | | | | | | . | | | | |
| | . | | | | | | | | | | | | . | | | | |
| | . | | | | | | | | | | | | . | | | | |
| $E | *additional offset for unipolar mode* | | | | | | | | *channel* | | | | *offset* | | | | channel 7 |
| $F | *10V range gain error (reserved)* | | | | | | | | *5V range (20mA range) gain error* | | | | | | | | |
| $10 | *user* | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| . | | | | | | | | | | | | | | | | | |
| $3F | | | | | | | | | | | | | | | | | |

*Read command for accessing an EEPROM address:

*EEPROM Read Command = $C000 + (Address <<7);*

**Note**

The parameters in brackets above refer to those specific to the 20mA version of the PB-ADC3.


© 1994 PEP Modular Computers

**EEPROM Description**

| Name | Value | Description |
|---|---|---|
| **additional offset for unipolar mode** *bits 15-8* | Typ. 6 | A byte value of the offset when offset voltage is selected on jumper B51 (position 1-2); unipolar measured in 5V range. **Example** $06 = 6LSB; (1 LSB = 1.22mV) |
| **channel** *bits 7-4* | 0<br>1<br>2<br>3<br>4<br>5<br>6<br>7 | Channel 0<br>Channel 1<br>Channel 2<br>Channel 3<br>Channel 4<br>Channel 5<br>Channel 6<br>Channel 7 |
| **offset** *bits 3-0* | Typ. 0 | A signed nibble value of the offset when the additional offset voltage is not selected on jumper B51 (position 1-3); unipolar measured in 5V range. **Example** $0 = 0LSB; (1 LSB = 1.22mV) |
| **10V range gain error** *bits 15-8* | Typ. $10 | A byte value for the gain difference of the full voltage value; unipolar measured. Not significant for the 20mA input version. **Example** Contents = $11 -> At $FFF - $11 = $FEE full voltage for unipolar voltage measurement 10V - 1LSB |
| **5V range gain error** *bits 7-0* | Typ. $20 | A byte value for the gain difference of the full voltage, or current, value; unipolar measured. **Example** Contents = $23 -> At $FFF - $23 = $FDC full voltage for unipolar voltage measurement 5V - 1LSB (for unipolar measurement of the current 20mA - 1LSB) |
| **user** *bits 15-0* | | 48 words free for user-specific data. |

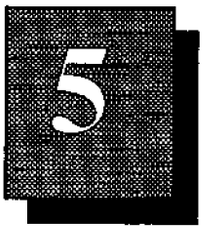
**Note**

Care must be taken when editing the EEPROM. PEP accepts no responsibility for the erasure of calibration data.


© 1994 PEP Modular Computers

*This page has been intentionally left blank*

 © 1994 PEP Modular Computers 

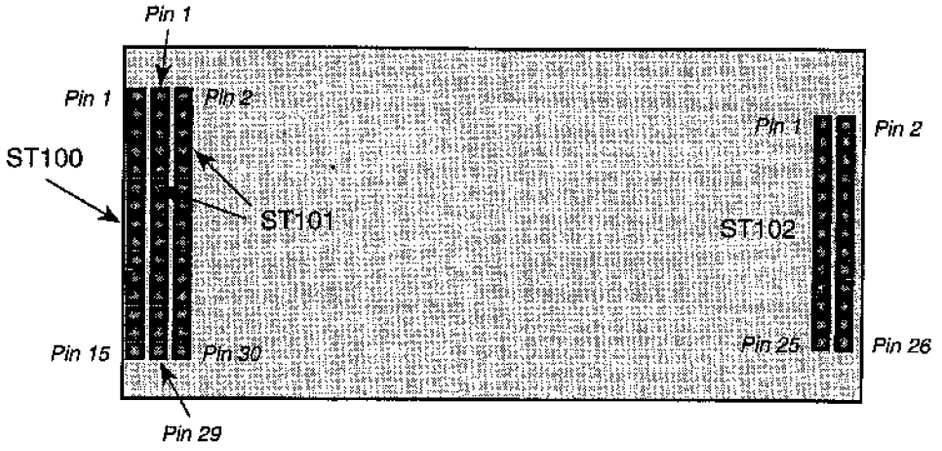


# 5. PINOUTS

The PB-ADC3 has three sets of connectors. ST100 is the rearmost row of 15 pins directly next to ST101 with 30 pins. At the front end of the piggyback there is ST102 with 26 pins.

## 5.1 Main Board

**Figure 5.1.0.1: Board Connector Overview**




### 5.1.1 ST100 Connector

The ST100 connector is fitted into the socket row BU0A or BU0B on the IMOD and VMOD-2, depending if there is one or two piggybacks to be fitted.

| Pin # | Signal |
|---|---|
| 1 | GND |
| 2 | N/C |
| 3 | N/C |
| 4 | N/C |
| 5 | N/C |
| 6 | IDS1* |
| 7 | ID15 |
| 8 | ID14 |
| 9 | ID13 |
| 10 | ID12 |
| 11 | ID11 |
| 12 | ID10 |
| 13 | ID09 |
| 14 | ID08 |
| 15 | GND |

* *Active Signal Low*


© 1994 PEP Modular Computers

### 5.1.2 ST101 Connector

The ST101 connector is fitted into the socket row BU1A or BU1B on the IMOD and VMOD-2 (depending if there is one or two piggybacks to be fitted) to the VMEbus interface logic.

| Pin# | Signal | Pin# | Signal |
|---|---|---|---|
| 1 | GND | 2 | VCC |
| 3 | N/C | 4 | N/C |
| 5 | IR/W* | 6 | CLK |
| 7 | RESET* | 8 | UDTACK* |
| 9 | INTA* | 10 | CS* |
| 11 | INT* | 12 | N/C |
| 13 | ID7 | 14 | IDS0 |
| 15 | ID6 | 16 | N/C |
| 17 | ID5 | 18 | IA6 |
| 19 | ID4 | 20 | IA5 |
| 21 | ID3 | 22 | IA4 |
| 23 | ID2 | 24 | N/C |
| 25 | ID1 | 26 | N/C |
| 27 | ID0 | 28 | N/C |
| 29 | GND | 30 | VCC |

* *Active Signal Low*

### 5.1.3 ST102 Connector

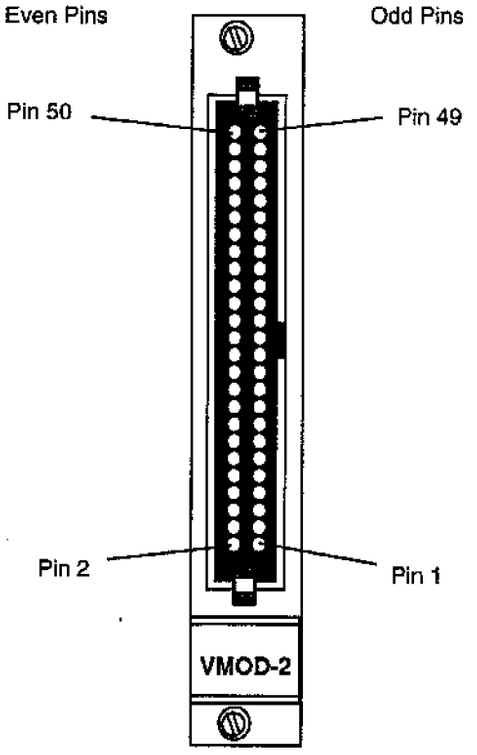
The ST102 connector is fitted into the socket row BU2A or BU2B on the IMOD and VMOD-2 (depending if there is one or two piggybacks to be fitted), directly to half of the 50-way VMOD-2 front panel connector, and ultimately to the user's external interfaces.

| Pin# | Signal | Pin# | Signal |
|---|---|---|---|
| 1 | A0(+) | 2 | A0(+) |
| 3 | GND | 4 | GND |
| 5 | A1(+) | 6 | B1(-) |
| 7 | A2(+) | 8 | B2(-) |
| 9 | GND | 10 | GND |
| 11 | A3(+) | 12 | B3(-) |
| 13 | A4(+) | 14 | B4(-) |
| 15 | GND | 16 | GND |
| 17 | A5(+) | 18 | B5(-) |
| 19 | A6(+) | 20 | B6(-) |
| 21 | GND | 22 | GND |
| 23 | A7(+) | 24 | B7(-) |
| 25 | B0(-) | 26 | B0(-) |


© 1994 PEP Modular Computers

## 5.2 VMOD-2/IMOD Front Panel

**Figure 5.2.0.1: VMOD-2/IMOD Front Panel**




*Piggyback A*

| Pin # | Signal | Pin # | Signal |
|---|---|---|---|
| 50 | A0(+) | 49 | B0(-) |
| 48 | GND | 47 | GND |
| 46 | A1(+) | 45 | B1(-) |
| 44 | A2(+) | 43 | B2(-) |
| 42 | GND | 41 | GND |
| 40 | A3(+) | 39 | B3(-) |
| 38 | A4(+) | 37 | B4(-) |
| 36 | GND | 35 | GND |
| 34 | A5(+) | 33 | B5(-) |
| 32 | A6(+) | 31 | B6(-) |
| 30 | GND | 29 | GND |
| 28 | A7(+) | 27 | B7(-) |
| 26 | Ext. Reset GND | 25 | Ext. Reset Vcc |

*Piggyback B*

| Pin # | Signal | Pin # | Signal |
|---|---|---|---|
| 24 | A0(+) | 23 | B0(-) |
| 22 | GND | 21 | GND |
| 20 | A1(+) | 19 | B1(-) |
| 18 | A2(+) | 17 | B2(-) |
| 16 | GND | 15 | GND |
| 14 | A3(+) | 13 | B3(-) |
| 12 | A4(+) | 11 | B4(-) |
| 10 | GND | 9 | GND |
| 8 | A5(+) | 7 | B5(-) |
| 6 | A6(+) | 5 | B6(-) |
| 4 | GND | 3 | GND |
| 2 | A7(+) | 1 | B7(-) |


© 1994 PEP Modular Computers

The relevant half of the VMOD-2 and IMOD front panel 50-way connector (pins 1..24 for lower position and pins 27..50 for upper) assumes the relationship of PB-ADC3 signals and its ST102.

An optional 50-way header behind the front panel connector has an identical pin-out to the front panel connector. It is provided for applications where the flat band cable is to be routed internally, or where an alternative front panel is to be fitted and used. In some cases, cables can be routed through the systems interior i.e. to the back panel (from this optional connector) and some from the external connector on the front panel. In doing so take care not to exceed the fan out ability of the piggyback's driver circuits.

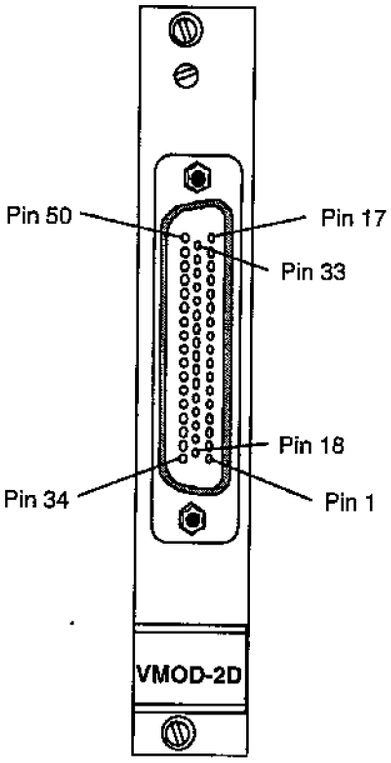
**Note**

With systems that have more than one of this type of connector , or which use several VMOD-2 or IMODs with various piggybacks, it is advisable to put a drop of paint on the back of the mating connector and on the front panel of the VMOD-2 or IMOD, for correct connection. The connector splits virtually in half (pins 1-24 and 27-50) for connection to the rear piggyback location.


© 1994 PEP Modular Computers

## 5.3 VMOD-2D Front Panel

**Figure 5.3.0.1: VMOD-2D Front Panel**




*Piggyback A*

| Pin # | Signal | Pin # | Signal | Pin # | Signal |
|---|---|---|---|---|---|
| 50 | A0(+) | 33 | GND | 17 | B0(-) |
| 49 | GND | 32 | B1(-) | 16 | A1(+) |
| 48 | A2(+) | 31 | GND | 15 | B2(-) |
| 47 | GND | 30 | B3(-) | 14 | A3(+) |
| 46 | A4(+) | 29 | GND | 13 | B4(-) |
| 45 | GND | 28 | B5(-) | 12 | A5(+) |
| 44 | A6(+) | 27 | GND | 11 | B6(-) |
| 43 | GND | 26 | B7(-) | 10 | A7(+) |
| 42 | Ext. Reset GND | | | 9 | Ext. Reset Vcc |

*Piggyback B*

| Pin # | Signal | Pin # | Signal | Pin # | Signal |
|---|---|---|---|---|---|
| | | 25 | A0(+) | | |
| 41 | B0(-) | 24 | GND | 8 | GND |
| 40 | A1(+) | 23 | A2(+) | 7 | B1(-) |
| 39 | B2(-) | 22 | GND | 6 | GND |
| 38 | A3(+) | 21 | A4(+) | 5 | B3(-) |
| 37 | B4(-) | 20 | GND | 4 | GND |
| 36 | A5(+) | 19 | A6(+) | 3 | B5(-) |
| 35 | B6(-) | 18 | GND | 2 | GND |
| 34 | A7(+) | | | 1 | B7(-) |


© 1994 PEP Modular Computers

## 5.4 VMOD-2 / VMOD-2D Pinout Relationship

*Piggyback A*

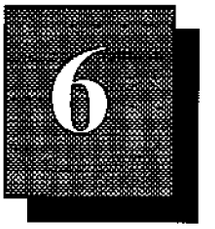
| Signal | VMOD-2 Pin # | VMOD-2D Pin # | Signal | VMOD-2 Pin # | VMOD-2D Pin # |
|---|---|---|---|---|---|
| A0(+) | 50 | 50 | B0(-) | 49 | 17 |
| GND | 48 | 33 | GND | 47 | 49 |
| A1(+) | 46 | 16 | B1(-) | 45 | 32 |
| A2(+) | 44 | 48 | B2(-) | 43 | 15 |
| GND | 42 | 31 | GND | 41 | 47 |
| A3(+) | 40 | 14 | B3(-) | 39 | 30 |
| A4(+) | 38 | 46 | B4(-) | 37 | 13 |
| GND | 36 | 29 | GND | 35 | 45 |
| A5(+) | 34 | 12 | B5(-) | 33 | 28 |
| A6(+) | 32 | 44 | B6(-) | 31 | 11 |
| GND | 30 | 27 | GND | 29 | 43 |
| A7(+) | 28 | 10 | B7(-) | 27 | 26 |

| Signal | VMOD-2 Pin # | VMOD-2D Pin # | Signal | VMOD-2 Pin # | VMOD-2D Pin # |
|---|---|---|---|---|---|
| Ext. Reset GND | 26 | 42 | Ext Reset Vcc | 25 | 9 |

*Piggyback B*

| Signal | VMOD-2 Pin # | VMOD-2D Pin # | Signal | VMOD-2 Pin # | VMOD-2D Pin # |
|---|---|---|---|---|---|
| A0(+) | 24 | 25 | B0(-) | 23 | 41 |
| GND | 22 | 8 | GND | 21 | 24 |
| A1(+) | 20 | 40 | B1(-) | 19 | 7 |
| A2(+) | 18 | 23 | B2(-) | 17 | 39 |
| GND | 16 | 6 | GND | 15 | 22 |
| A3(+) | 14 | 38 | B3(-) | 13 | 5 |
| A4(+) | 12 | 21 | B4(-) | 11 | 37 |
| GND | 10 | 4 | GND | 9 | 20 |
| A5(+) | 8 | 36 | B5(-) | 7 | 3 |
| A6(+) | 6 | 19 | B6(-) | 5 | 35 |
| GND | 4 | 2 | GND | 3 | 18 |
| A7(+) | 2 | 34 | B7(-) | 1 | 1 |


© 1994 PEP Modular Computers



# 6. INSTALLATION

## 6.1 VMEbus Connection

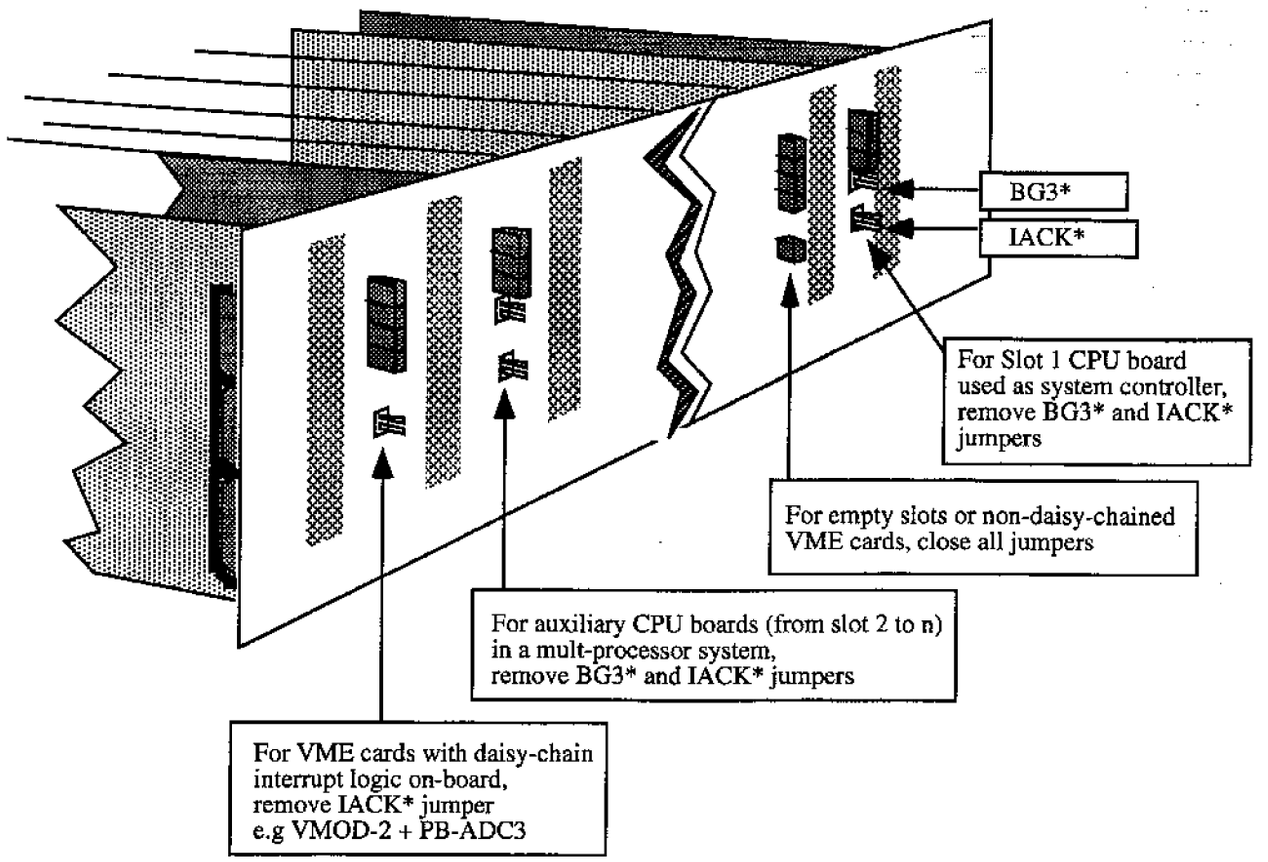
**Caution!**

Before installing or removing any VMEbus boards, always turn off the power to the bus and any external peripherals.

Inserting or removing modules, while the power is on, could result in damage to the VME module or peripherals interface.

Please refer to VMOD-2/IMOD user's manual for details on installing or removing.

**Figure 6.1.0.1: The VMEbus Backplane**




 © 1994 PEP Modular Computers 

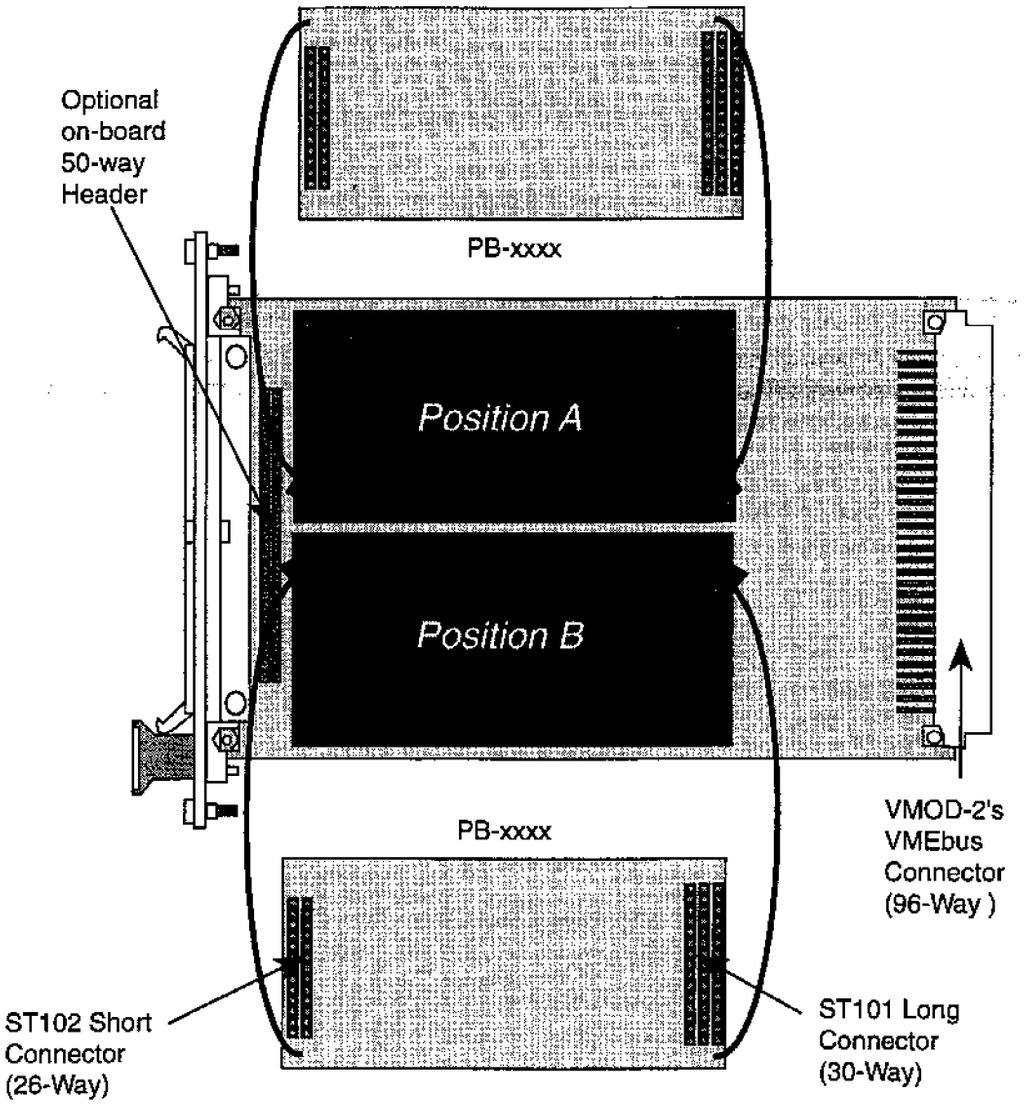
## 6.2 Installing the PB-ADC3

The PB-ADC3 may be plugged into any free piggyback position (A or B) on the VMOD-2 or IMOD. Please ensure the correct location before fitting.

**Note**

1. One connector on the PB-ADC3 has fewer pins than the other.
2. ST101 has two-rows which are to fit the front two-rows of the VMOD-2 amd IMOD three-row interface socket. Take care to ensure that the piggyback is in its correct position.

**Figure 6.2.0.1: PB-ADC3 Installation Overview**




Mechanical fastening and support is provided by the two interface connectors. In addition, the piggyback can be attached to the motherboard by screws and stand-off pillars at the front end of the piggyback, and at the corresponding location on the VMOD-2 and IMOD at the two holes provided for this purpose.

 © 1994 PEP Modular Computers 

## 6.3 General Notes for Using the System

Having designed a system, it is necessary to keep it in good working order. The three biggest risks to the system occur when:

- Connecting peripherals, disk-drives, printers, terminals and external power sources;
- Adding or changing modules, address settings and locations, etc;
- Becoming complacent and not referring to the manuals when altering or adding modules.

These risks can be reduced by:

- Checking the electrical compatibility of all devices to be connected;
- Ensuring that they are powered from the same mains supply branch (phase) and grounded to the same reference point;
- Shutting down all power before making or breaking any connections to modules or attachments to the system, including power to the peripherals;
- Observing sensible static protection procedures before handling any modules, piggybacks or memory IC's;
- Keeping all manuals available by the system and refer to them when required.

Some tips are:-

PEPCards are not over sensitive to static, but it is generally advisable to observe normal antistatic procedures.

When configuring the module, it should not be taken out of the original packing unless necessary. The clear packs can be opened and the jumpers set, piggybacks added, etc. without removing the card. This prevents inadvertent shorting of any on-board devices.

When inserting modules into a system, the power should be turned off and the mains lead not removed! The ground wire prevents the rack floating with dangerous static voltages, which could destroy circuits on the module being inserted.

The front panel of the module and the shell of the connector should be touched to any part of the rack before fitting. This discharges any static from the user.

Modules should not be pulled straight out of a rack and the back of the front panel checked to see if there are cables to unplug (such as the VSBC-1's 40-pin parallel on-board headers). Any leads should be disconnected from a module before unscrewing the front panel and removing from the rack. It should be ensured that, when fitted, these cables have enough play to allow the modules to be removed far enough to detach these cables. Modules should be put into the rack before connecting any front-panel connectors.

The "pulled" jumpers should be parked on to one of the pins they would normally bridge, so they can be quickly replaced.

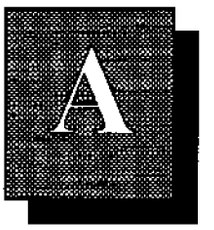
It should be remembered to check the mains input voltage selector switch before installing or using any PSU!

A record should be kept of settings and a copy forwarded with any board returned to PEP for failure analysis.


© 1994 PEP Modular Computers

*This page has been intentionally left blank*

 © 1994 PEP Modular Computers 




# APPENDIX A. COMPLEX EXAMPLES

## A.1 Read Channels

The following program is for a continuous read of all 8 channels.

### A.1.1 C-Program Listing

```
#include <errno.h>
#include <modes.h>
#include <sgstat.h>
#include <stdio.h>

/*****************************
*          Address Offset Table            *
 *****************************/

#define LTC           ( 0x00 >>1 )          /* word read/write */
#define EEPROM        ( 0x10 >>1 )          /* word read/write */
#define EEP_PROG      ( 0x20 >>1 )          /* word write */
#define STATUS          0x31                /* byte read/write */
#define ID              0x7f                /* byte read */

/*****************
*       definitions        *
*****************/

#define READY         0xfa        /* STATUS register READY check  */
#define PBADC3_ID     0xEB        /* PB-ADC3 identification byte */
#define UNI           0x10        /* to add to code-word for unipolar conversion
*/
#define BIP           0x00        /* to add to code-word for bipolar conversion */

#define LSBUni        0.00122
#define LSBBip        0.00244

#define LSB10VBip     LSBBip*2
#define LSB5VBip      LSBBip
#define LSB10VUni     LSBUni*2
#define LSB5VUni      LSBUni


static unsigned char *babase=0x87fe2400;   /* default base address of VMOD-2 */
static unsigned short *wabase=0x87fe2400;
```

 © 1994 PEP Modular Computers 

```
short Channel[ ]={
                    0xcf + Bip,       /* channel 0 */
                    0xcd + Bip,       /* channel 1 */
                    0xc7 + Bip,       /* channel 2 */
                    0xc5 + Bip,       /* channel 3 */
                    0xcb + Bip,       /* channel 4 */
                    0xc9 + Bip,       /* channel 5 */
                    0xc3 + Bip,       /* channel 6 */
                    0xc1 + Bip,       /* channel 7 */
                    0xcf + Bip        /* again channel 0 for the read out loop */
                    };

float chan_LSB[ ]={
                    LSB10VBip,        /* channel 0 LSB definition */
                    LSB10VBip,        /* channel 1 LSB definition */
                    LSB10VBip,        /* channel 2 LSB definition */
                    LSB10VBip,        /* channel 3 LSB definition */
                    LSB10VBip,        /* channel 4 LSB definition */
                    LSB10VBip,        /* channel 5 LSB definition */
                    LSB10VBip,        /* channel 6 LSB definition */
                    LSB10VBip,        /* channel 7 LSB definition */
                    LSB10VBip         /* channel 0 LSB definition */
                    };


/************************
 *   PB-ADC3 Functions     *
 ************************/

ADC_check()
{
          for(count=0;count<100;count++)
          {
                    status=*(babase+STATUS);/* check if EOC or shift is ready */
                    if ((status&0xff)==READY)
                    break;
          }
}


ADC_convert(channel,mode)
short channel;                /* Channel number 0-7 */
char mode;                    /* Unipolar = 'U'; Bipolar = '`B'; */
{
          short adcconv;
          adcconv=code[channel];
          if (mode=='U') adcconv=adcconv|0x10;
          ADC_check();
          *(adc_wabase+LTC)=adcconv;
}
```


© 1994 PEP Modular Computers

```
/***************
* Main Program  *
***************/
main()
{
          short value,bipvalue,univalue;


          if (*(babase+ID)!=PBADC3_ID)      /* PB-ADC3 ID Check */
          {
               printf("\nError: Wrong PB-ADC3 ID found! Correct one is $EB!\n");
               printf(" Read: $%x\n\n",*(babase+ID));
               exit(0);
          }


          printf("Select HEX/Voltage? (H/V):   "); fflush(stdout);
          scanf("%c",&chr);
          chr = toupper(chr);

          while(1)
          {
               for (n=0;n<8;n++)
               {
               ADC_convert(n+1,'B');             /* initiate next conversion */
               ADC_check();                      /* wait for end of conversion */
               bipvalue=*(wabase+LTC);           /* read out conversion result */
               voltage=(float)bipvalue*chan_LSB[n];     /* calculate voltage */

               switch (chr)            /* switch output mode */
               {
                         case 'H':    printf("%d=$%4x",n,bipvalue&0xffff);
                                      if (n<7) printf(" | ");
                                      break;

                         case 'V':    if (voltage>=0)printf("+%2.5fV",voltage);
                                      else printf("%2.5f%V",voltage);
                                      if (n<7) printf(" ");
                                      break;

                         default:     printf("\nGood bye! Please try again!\n\n");
                                      break;
               }
               printf("\n");
          }
}

exit(0);
}


/* program end  */
```


© 1994 PEP Modular Computers

### A.1.2 Assembler Program

```
* endless loop example for read out channel 0

BASE      equ         $fe2400             base address VMOD2 default
LTC       equ         $00                 LTC communication register
STATUS    equ         $31                 status register offset
CHAN0     equ         $cf                 channel 0 bipolar conversion code


start:    cmpi.b      #$fa,BASE+STATUS    shift register ready ?
          bne.s       start               no, wait

          move.w      #CHAN0,BASE+LTC     start first conversion channel 0
                      .
eoc:      cmpi.b      #$fa,BASE+STATUS    end of conversion ?
          bne.s       eoc                 no, wait

loop:     move.w      #CHAN0,BASE+LTC     start next conversion channel 0

data:     cmpi.b      #$fa,BASE+STATUS    end of conversion and previous data
                                          ready?
          bne.s       data                no, wait

          move.w      BASE+LTC,d0         get previous conversion result

          bra.s       loop                loop back to next conversion


* end of file
```


© 1994 PEP Modular Computers

## A.2 Converted Data Correction

```
#define EOC_READY      0xFA     /* STATUS register ready check */
#define NOT_EOC        0x04     /* STATUS register ready check */
#define EE_WRITE_CMD   0xA000   /* Command to write a value to an address */
#define EE_READ_CMD    0xC000   /* Command to read a value from an address */
#define EE_ENA_CMD     0x9800   /* Command to read a value from an address */
#define EE_DISA_CMD    0x8000   /* Command to read a value from an address */
#define STATUS_EOC     0x4      /* EOC bit  0=finished 1=conversion */
#define STATUS_INT     0x2      /* Pending Interrupt bit 0=active 1=inactive */
#define STATUS_INTENA  0x1      /* Interrupt enabled bit 0=disabled 1=enabled */
#define INT_DISABLE    0xFE     /* Disable Interrupts */

#define C_SHIFT     19       /* Shift to get higher accuraccy */

#define UNIPOLAR         0x10
#define BIPOLAR          0x00
#define SINGLE_ENDED     0xC0

/* special register of piggy back ... */
#define PB_ID(base)        ((UINT8*)((int)(base)+0x7f))  /* piggy back id adrs */

/* ... and its possible correct contains */
#define ID_ADC3      0xEB

/* register definitions */
#define ADC3_COMM(base)     ((UINT16*)((base)+0x00)) /* communication reg */
#define ADC3_EE_COMM(base)  ((UINT16*)((base)+(0x10>>1))) /* EEPROM comm reg */
#define ADC3_EE_PROG(base)  ((UINT16*)((base)+(0x20>>1))) /* EEPROM program reg */
#define ADC3_STATUS(base)   ((UINT8*) ((base)+0x31)) /* Status Register */


typedef struct          /* Calibration Data Structure */
    {
    UINT8  extOffset;
    UINT8  channel;
    UINT8  stdOffset;
    UINT8  calib10;
    UINT8  calib5;
    } CALIB_DATA;

typedef struct          /* ADC3_DEV */
    {
    DEV_HDR ioDev;
    int     chan;           /* channel number */
    BOOL    created;        /* true if this device has really been created */
    UINT16  *pb_base;       /* base address of the piggy back */
    SEM_ID  login;          /* login flag */
    BOOL    calibration;    /* correct converted data with calibration
                               (default true) */
    BOOL    offsetType;  /* correct converted data with offset (default false) */
    BOOL    voltage10;      /* What voltage is set (default 10 V) */
    CALIB_DATA calibData[8];/* calibration data copy */
    INT32   calArrBi10V[8];   /* calibration data for channels */
    INT32   calArrUni10V[8];  /* calibration data for channels */
    INT32   calArrBi5V[8];    /* calibration data for channels */
    INT32   calArrUni5V[8];   /* calibration data for channels */
    INT16   offArrBi10V[8];   /* calibration data for channels */
```

 © 1994 PEP Modular Computers 

```
    INT16   offArrUni10V[8];  /* calibration data for channels */
    INT16   offArrBi5V[8];    /* calibration data for channels */
    INT16   offArrUni5V[8];   /* calibration data for channels */
    } ADC3_DEV;

LOCAL ADC3_DEV adc3Dv [] =  /* device descriptors */
    {
    /* VMOD_0 PB_A */
    {{{NULL}}, 0, FALSE, (UINT16 *)(ADR_VMOD_0+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_0 PB_B */
    {{{NULL}}, 1, FALSE, (UINT16 *)(ADR_VMOD_0+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_1 PB_A */
    {{{NULL}}, 2, FALSE, (UINT16 *)(ADR_VMOD_1+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_1 PB_B */
    {{{NULL}}, 3, FALSE, (UINT16 *)(ADR_VMOD_1+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_2 PB_A */
    {{{NULL}}, 4, FALSE, (UINT16 *)(ADR_VMOD_2+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_2 PB_B */
    {{{NULL}}, 5, FALSE, (UINT16 *)(ADR_VMOD_2+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_3 PB_A */
    {{{NULL}}, 6, FALSE, (UINT16 *)(ADR_VMOD_3+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_3 PB_B */
    {{{NULL}}, 7, FALSE, (UINT16 *)(ADR_VMOD_3+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_4 PB_A */
    {{{NULL}}, 8, FALSE, (UINT16 *)(ADR_VMOD_4+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_4 PB_B */
    {{{NULL}}, 9, FALSE, (UINT16 *)(ADR_VMOD_4+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_5 PB_A */
    {{{NULL}}, 10, FALSE,(UINT16 *)(ADR_VMOD_5+ADR_PB_A), FALSE, TRUE, TRUE,
          TRUE},

    /* VMOD_5 PB_B */
    {{{NULL}}, 11, FALSE,(UINT16 *)(ADR_VMOD_5+ADR_PB_B), FALSE, TRUE, TRUE,
          TRUE}
    };
```

 © 1994 PEP Modular Computers 

```
/*****************************************************************************
*
* eePromRead - read data from EEPROM at specified address
*/

LOCAL STATUS eePromRead(base, address, value)
UINT16 *base;
UINT16 address;
UINT16 *value;
    {
    STATUS status;

    *(ADC3_EE_COMM (base)) = ((UINT16) EE_READ_CMD)|(address<<7);

    adc3Check(base);
    *value = *(ADC3_EE_COMM (base));

    status = adc3Check(base);
    wait();
    return (status);
    }

/*****************************************************************************
*
* getEEdata - copy default EEPROM data into RAM
*
*          15   12 11    8 7     4 3     0
*         ---------------------------------
* WORD 0 | ext. Offset   |channel|std.Off|
*         ---------------------------------
* WORD 1 | cal. Data 10V | cal. Data 5V  |
*         ---------------------------------
*/

LOCAL STATUS getEEdata (base, data)
UINT16 *base;
CALIB_DATA data[];

    {
    INT16 chan;
    STATUS status;
    UINT16 value;

    for (chan=0; chan<8; chan++)
        {
        if ((status = eePromRead (base, chan*2, &value)) != OK)
            return (status);
        data[chan].extOffset = value >> 8;
        data[chan].stdOffset = value & 0x0F;
        data[chan].channel = (value & 0xF0) >> 4;

        if ((status = eePromRead (base, chan*2+1, &value)) != OK)
            return (status);
        data[chan].calib10 = value >> 8;
        data[chan].calib5  = value & 0xFF;
        }

    return (OK);
    }
```


© 1994 PEP Modular Computers

```
/*****************************************************************************
*
* correct - corrects converted data with calibration data
*/

LOCAL INT16 correct(pAdc3Dv, value, polarity, channel)
ADC3_DEV *pAdc3Dv;
INT16 value;
UINT8 polarity;
UINT8 channel;

     {
     if (pAdc3Dv->calibration)
          if (pAdc3Dv->voltage10)
               {
               if (polarity == UNIPOLAR)
                    return ((INT16) ( (INT32) ( ( (INT32) value << C_SHIFT) +
                         value * pAdc3Dv->calArrUni10V[channel]) >> C_SHIFT) -
                         pAdc3Dv->offArrUni10V[channel]);
               else
                    return ((INT16) ( (INT32) ( ( (INT32) value << C_SHIFT) +
                         value * pAdc3Dv->calArrBi10V[channel]) >> C_SHIFT) -
                         pAdc3Dv->offArrBi10V[channel]);
               }
          else
               if (polarity == UNIPOLAR)
                    return ((INT16) ( (INT32) ( ( (INT32) value << C_SHIFT) +
                         value * pAdc3Dv->calArrUni5V[channel]) >> C_SHIFT) -
                         pAdc3Dv->offArrUni5V[channel]);
               else
                    return ((INT16) ( (INT32) ( ( (INT32) value << C_SHIFT) +
                         value * pAdc3Dv->calArrBi5V[channel]) >> C_SHIFT) -
                         pAdc3Dv->offArrBi5V[channel]);
     else
          return (value);
     }

/*****************************************************************************
*
* setArrays - set calibration arrays from calibration data
*/

LOCAL STATUS setArrays (pAdc3Dv)
     ADC3_DEV *pAdc3Dv;

     {
     INT16 chan;
    INT32 value;
     INT8 offset;
```

 © 1994 PEP Modular Computers 

```
    for (chan=0; chan<8; chan++)
        {
        value = (INT32) pAdc3Dv->calibData[chan].calib10;
        pAdc3Dv->calArrUni10V [chan] =
            (INT32)((value << C_SHIFT) / ((INT16) 0x0FFF - (INT16)value));
        pAdc3Dv->calArrBi10V [chan] =
            pAdc3Dv->calArrUni10V [chan] + pAdc3Dv->calArrUni10V [chan];
        value = (INT32) pAdc3Dv->calibData[chan].calib5;
        pAdc3Dv->calArrUni5V [chan] =
            (INT32) ((value << C_SHIFT) / ((INT16) 0x0FFF - (INT16)value));
        pAdc3Dv->calArrBi5V [chan] =
            pAdc3Dv->calArrUni5V [chan] + pAdc3Dv->calArrUni5V [chan];

        if (pAdc3Dv->offsetType)
            offset = pAdc3Dv->calibData[chan].stdOffset;
        else
            offset = pAdc3Dv->calibData[chan].extOffset;
        pAdc3Dv->offArrUni5V [chan] = offset;
        pAdc3Dv->offArrBi5V [chan] = (offset + 1) >> 1;      /* round (x/2) */
        pAdc3Dv->offArrUni10V [chan] = (offset + 1) >> 1;    /* round (x/2) */
        pAdc3Dv->offArrBi10V [chan] = (offset + 2) >> 2;     /* round (x/4) */
        }
    return (OK);
    }


main ()
    {
    ADC3_DEV *pAdc3Dv;
    int i;
    INT16 *buffer, buf[10], value, address;
    UINT8 chan, channel;
    UINT16 mode;
    UINT32 repeater;

    pAdc3Dv = &adc3Dv[0]
   /* copy default EEPROM data into RAM */

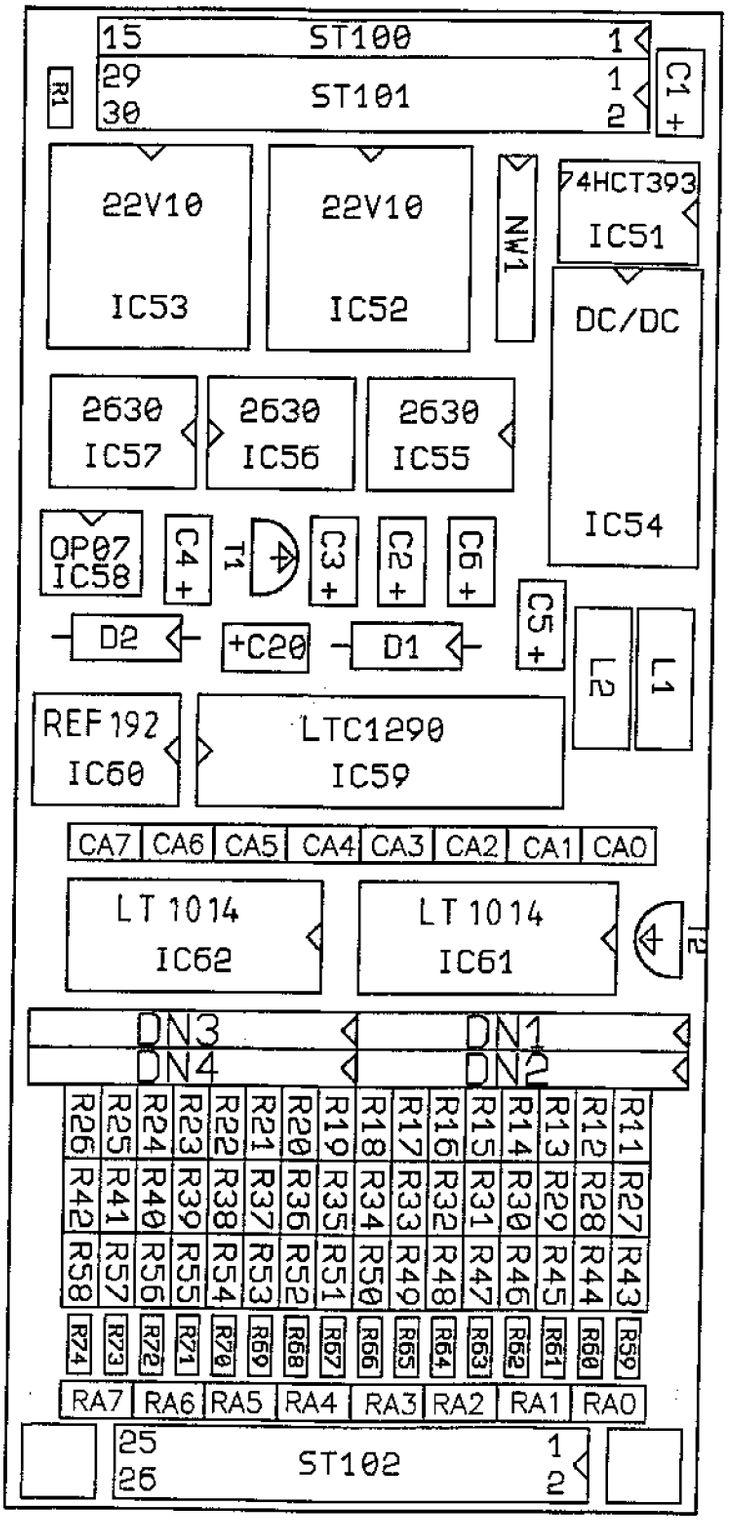
   if (getEEdata(adc3Dv[channel].pb_base, &adc3Dv[0].calibData) == ERROR)
       return (ERROR);

    buffer = buf;
    chan = 0;
    channel = channels [chan];
    mode = (UINT16) (UNIPOLAR | channel);
    repeater = 10;
```


© 1994 PEP Modular Computers

```
    for (i=0; i<=repeater; i++)                    /* repeat + 1 conversions */
{                                                  /* to get 10 values    */
        adc3Convert(pAdc3Dv->pb_base, mode);
        if (i != 0)
             {
             value = adc3Read(pAdc3Dv->pb_base);
             if (value == (INT16) 0xF000)
               {
                return (ERROR);
                }
             *buffer++ = correct (pAdc3Dv, value, UNIPOLAR, channel);
             }
         else
             status = adc3Check(pAdc3Dv->pb_base);
         }
     }
```


© 1994 PEP Modular Computers




B-Seite / B-side

ORIGINAL

Modular Computers

PB-ADC 3

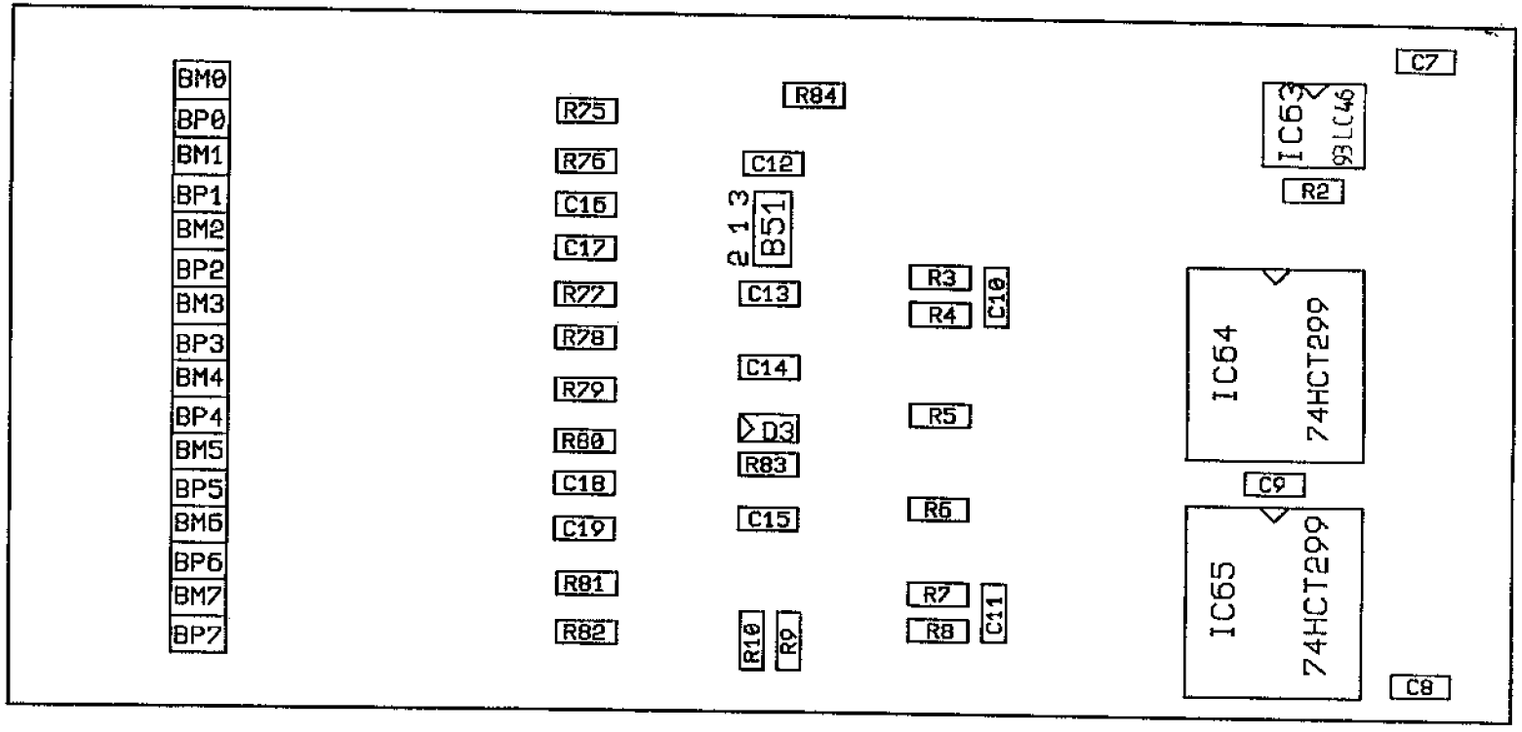
Date 02.07.1996

31 112-12301

Index 03

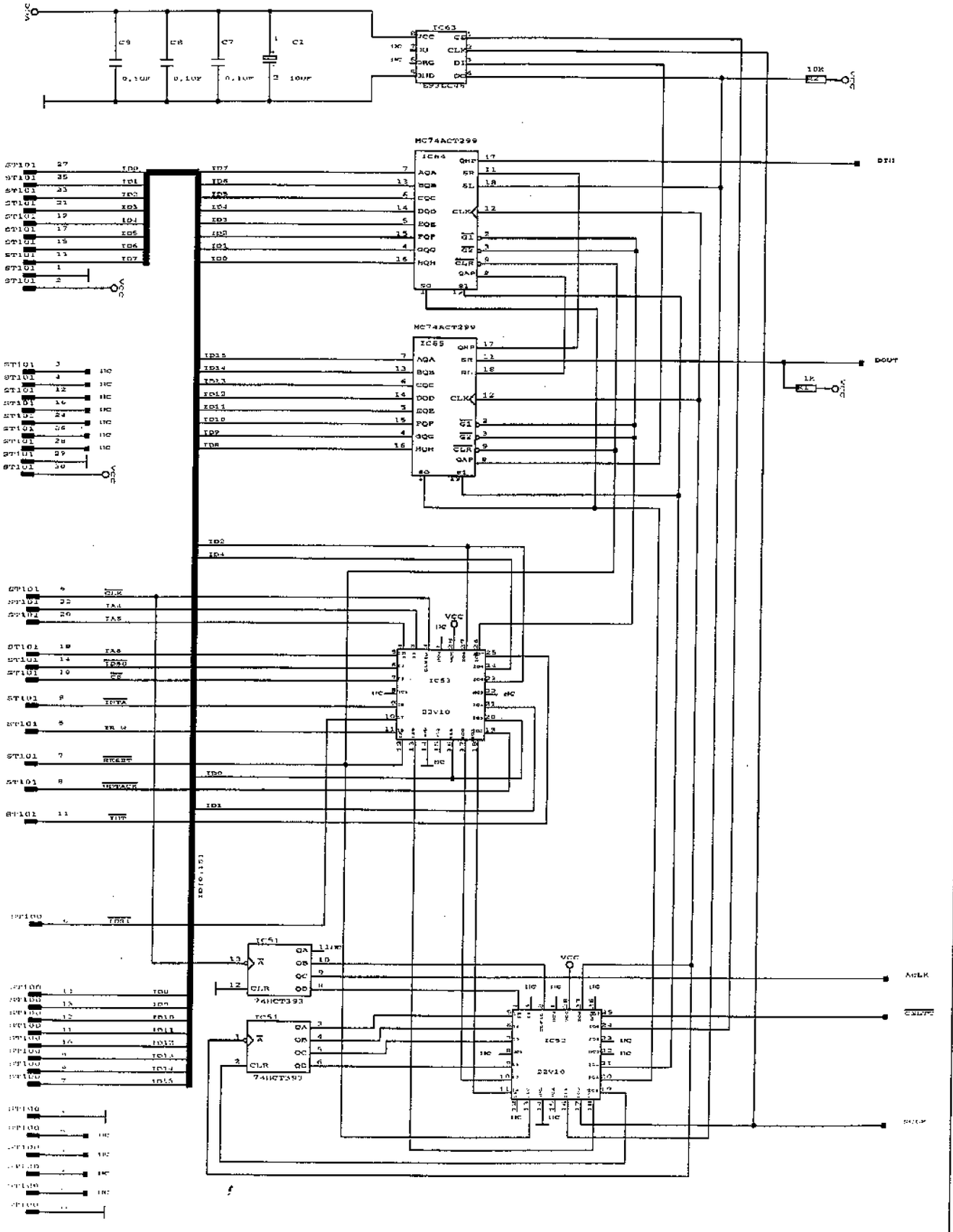
Page 1 of 2

L-Seite / L-side



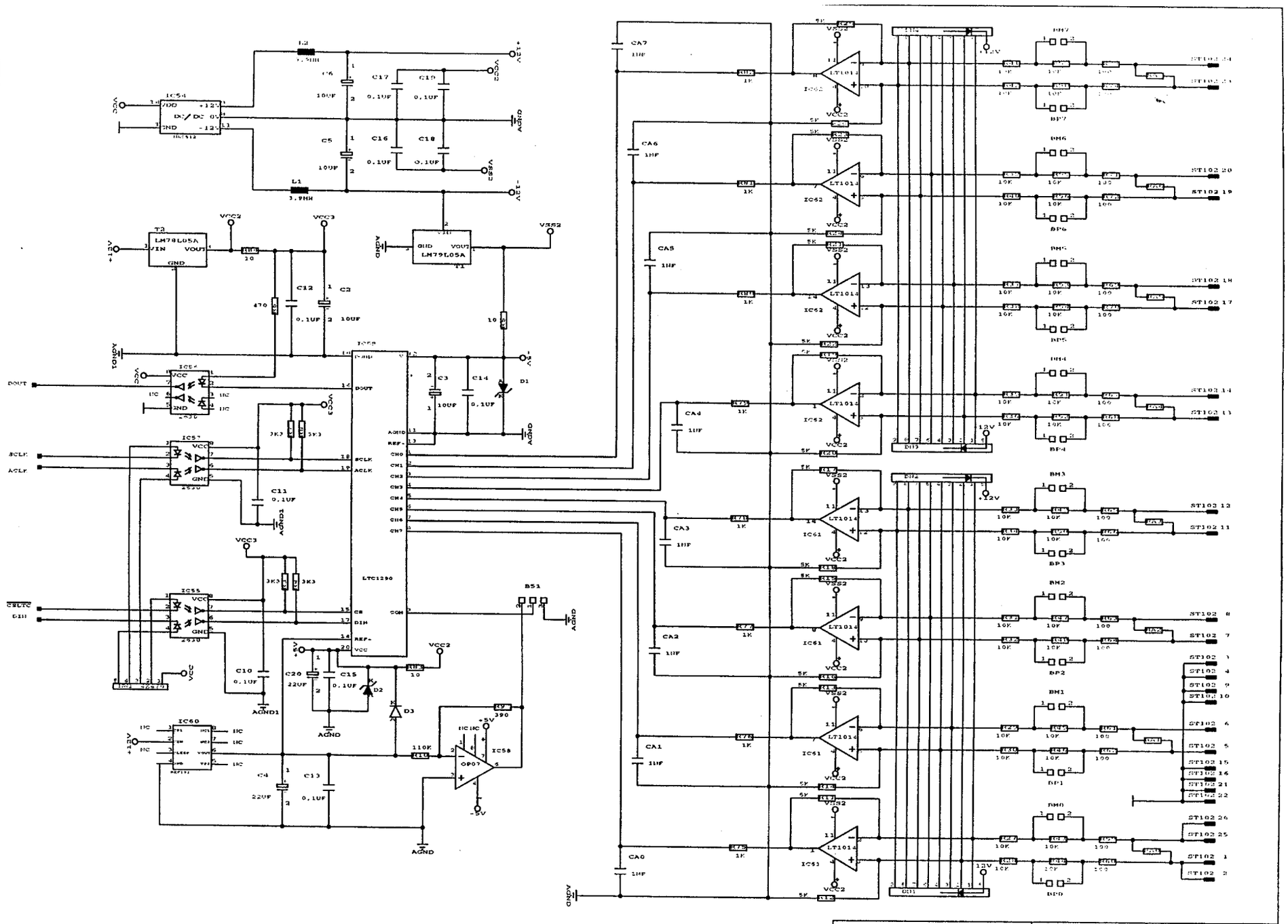

ORIGINAL

| | | |
|---|---|---|
| Modular Computers® | PB-ADC3 | 31112-12301 |
| | Date 02.07.1996 | Index 03 |
| | | Page 2 of 2 |




ORIGINAL

| | | |
|---|---|---|
| Modular Computers | PB-ADC3 | 31.112-1230.1 |
| | Date 02.07.96 | Index 03 |
| | | Page 1 of 2 |



ORIGINAL

Modular Computers

| | |
|---|---|
| PB AD14 | 11.112-1210.1 |
| Date 02.07.97 | Index 04 |
| | Page 2 of 2 |

**Artisan Technology Group** is an independent supplier of quality pre-owned equipment

## Gold-standard solutions

Extend the life of your critical industrial, commercial, and military systems with our superior service and support.

## We buy equipment

Planning to upgrade your current equipment? Have surplus equipment taking up shelf space? We'll give it a new home.

## Learn more!

Visit us at **artisantg.com** for more info on price quotes, drivers, technical specifications, manuals, and documentation.

Artisan Scientific Corporation dba Artisan Technology Group is not an affiliate, representative, or authorized distributor for any manufacturer listed herein.

We're here to make your life easier. How can we help you today?

(217) 352-9330 | sales@artisantg.com | artisantg.com

ARTISAN® TECHNOLOGY GROUP