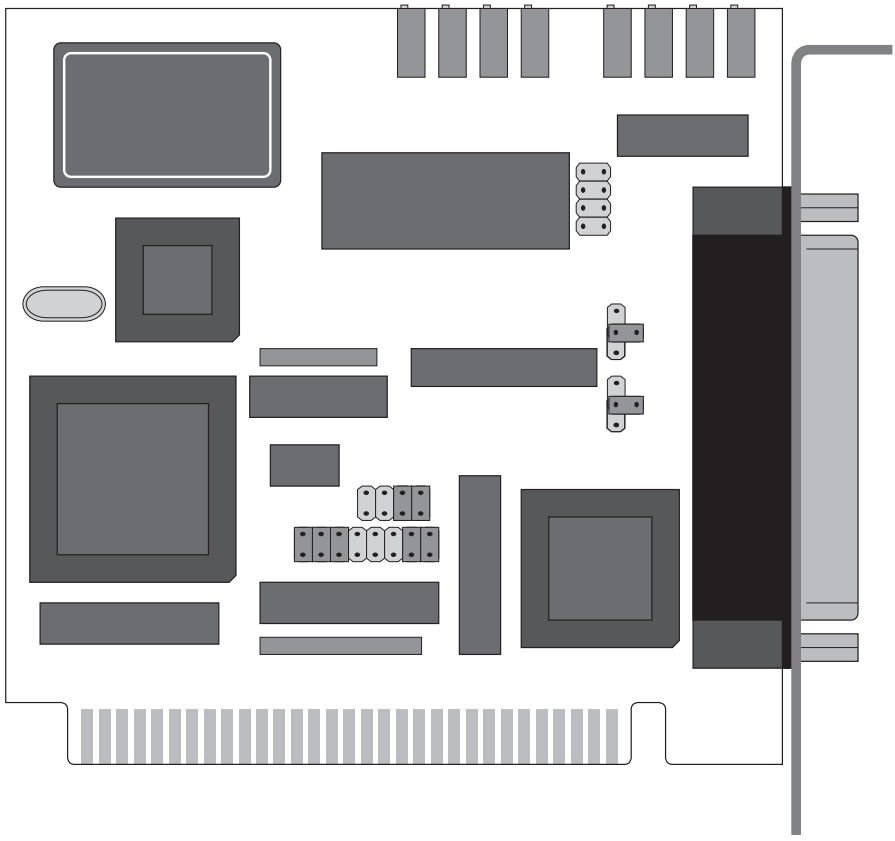
Meilhaus Electronic Manual

ME-270 2.0E



Multi-I/O Board with simultaneous Sampling for ISA Bus

Imprint

Manual ME-270

Revision 2.0E
Revised: 6.12.99

Meilhaus Electronic GmbH
Fischerstraße 2
D-82178 Puchheim/Munich
Germany
http://www.meilhaus.com

Important note:
The information contained in this manual has been reviewed with great care and is believed to be complete and accurate. Meilhaus Electronic assumes no responsibility for its use, any infringements of patents or other rights of third parties which may result from use of this manual or the product. Meilhaus Electronic assumes no responsibility for any problems or damage which may result from errors or omissions. Specifications and instructions are subject to change without notice.

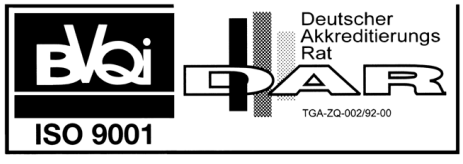


Table of Content

1 Introduction

Valued customer!

Thank you for purchasing the ME-270 data acquisition board. You have chosen an innovative high technology board that left our premises in a fully functional and new condition.

Please take the time to examine the contents of the package for any loss or damage that may have occurred during shipping. If there are any contents missing or if an item is damaged, contact Meilhaus Electronic immediately.

Before you install the board in your computer, read this manual carefully, especially the chapter describing board installation. Pay careful attention to the instructions on how to set the dip switches and jumpers on the board. This will save you having to open your PC case again.

1.1 Package Contents

We take great care to make sure that the package is complete in every way when it is shipped. We do ask, that you take the time to examine the contents of the box. Your ME-270 package should consist of:

- ME-270
- This manual
- ME-270 driver system for Windows 95/98/NT: 1 disk (3,5“ HD) „ME-270 Driver System“
- 25pin D-Sub female connector

1.2 Features

The ME 270 is a 12 bit simultaneous sampling multifunction board for the ISA bus. The four single ended simultaneously sampled analog input channels can be sampled with maximum frequency of 29 kHz. The board converts with a 12 bit accuracy and

offers input ranges of ±10 V and ±5 V. An interrupt can occur after the 4 channels have been sampled.

The board has two 12 bit analog output channels which can be set to output ranges of 0…10 V, 0…5 V, and ±5 V.

The digital I/O section of the board offers 3 ports, each 8 bits using the standard component 8255.

The software included allows quick integration of the board into user measurement and control applications running under Windows 95/98/NT. Drivers and demo programs are available for HP VEE (Hewlett-Packard) and LabVIEW™ (National Instruments) – both under Windows 95/98/NT. Drivers for Windows 3.1 and DOS are available upon request.

1.3 System Requirements

The ME-270 can be installed into any IBM AT/386/486, Pentium or compatible computer. One free ISA 8 bit bus slot is required.

1.4 Important Note for ISA Models

For computers with a PCI bus and a BIOS which supports Plug&Play make sure to reserve the interrupt lines for the ISA bus in the BIOS for any boards requiring the interrupt functions. The BIOS menu will vary depending on the manufacturer (consult the motherboard documentation on how to do this). **If the interrupt lines are not reserved in the BIOS, the interrupt function will not be guaranteed!!**

Also note that on some newer computers the frequency on the ISA bus will be more than the specified 8 MHz. Proper functioning of the boards is not guaranteed if this is the case. Check the BIOS setup of your computer to make sure.

To avoid excess electrical "noise" the board should be installed in an ISA slot as far away from the video board as possible.

1.5 Available Software

Windows 95/98/NT 4.0	ME-270 system driver
Windows 3.x	on request
MS-DOS	on request

High level languages 32 bit (included)

Visual C++ V 4.0 or higher
Delphi V 2.0 or higher
Visual Basic V 4.0 or higher

Graphical programming environments (optional)

Meilhaus HP VEE Driver System for HP VEE V3.2 or higher

ME-270 Driver System for LabView™ V4.0 or higher

Test and demo software (included)

For the newest versions and latest software releases, please consult the README file on the driver disk(s) supplied.

2 Installation

2.1 Hardware Installation

2.1.1 Locations of the Jumpers

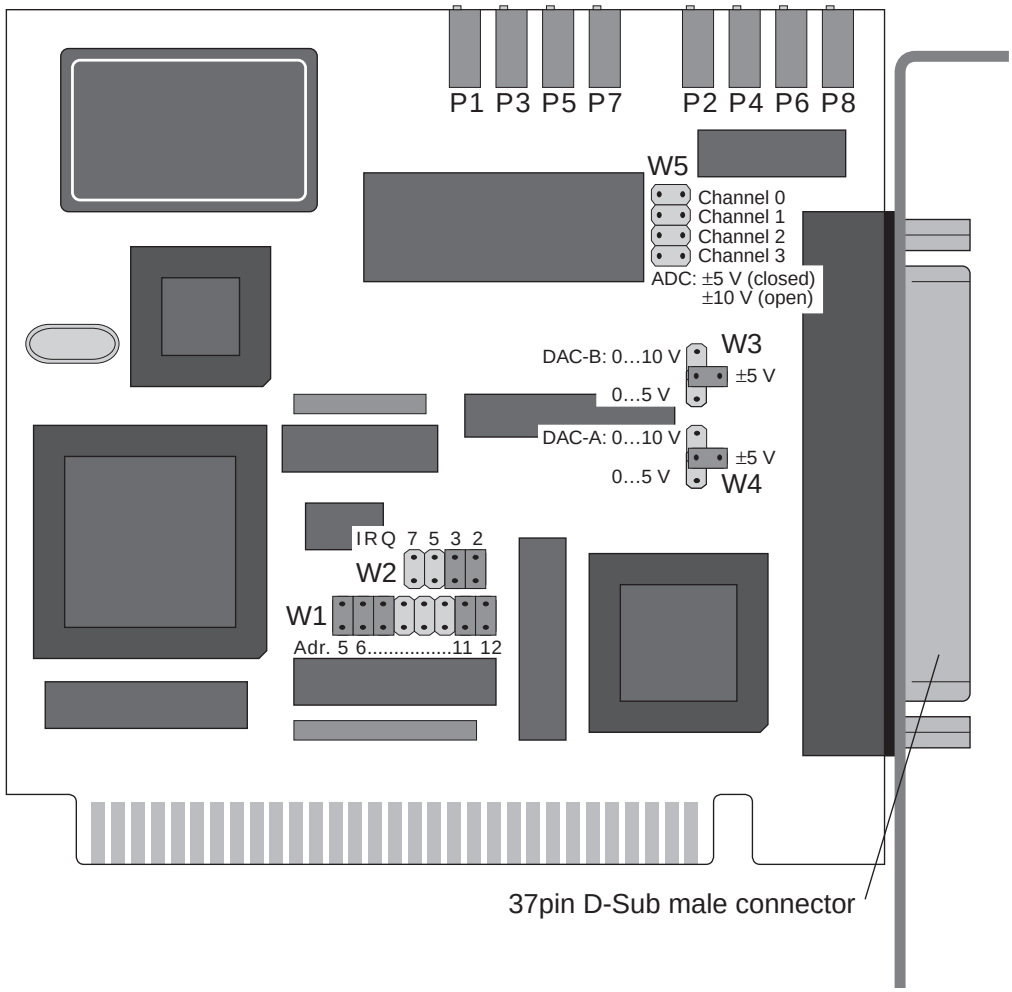


Diagram 1: Simplified illustration of the board

On the ME-270 the base address (W1), interrupt channel (W2), output voltage range (W3, W4) and input voltage range (W5) are selected by jumper settings. The positions of these jumpers can be seen in Diagram 1: "Simplified illustration of the board". Detailed description of the jumper settings are given in the next section.

2.1.2 Jumpers Settings

2.1.2.1 Base Address

The base address (BA) is set by the jumper W1 in the range of 0…1FE0Hex in 32 byte increments. Starting with the base address the ME-270 occupies 16 byte of the I/O address space. Make sure that there are no conflicts with other boards in the system before installing the board in the computer!

A jumper that is "on" sets a logic "0" on the address line and a jumper that is "off" sets a logic "1" on the address line. The base address is determined by summing the jumpers which are "off". The following example shows the default setting on the board (700Hex).

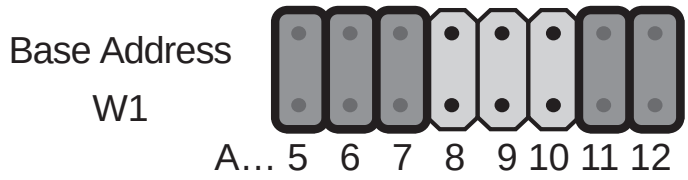


Diagram 2: Setting the base address, e. g. standard setting 700Hex

2.1.2.2 Interrupts

The desired interrupt line is selected by jumper W2. The IRQ lines 2, 3, 5 oder 7 can be chosen. Make sure that the selected interrupt is not in use with other boards in the system!

The following diagram shows, how the jumper has to be set to choose e. g. IRQ line 5::

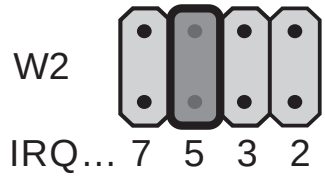


Diagram 3: Setting the interrupt line, e. g. standard setting IRQ 5

2.1.2.3 Input Range

The jumper W5 is used to set the input range for the A/D channels 0…3. Each channel can be separately set to either ±10 V or ±5 V.

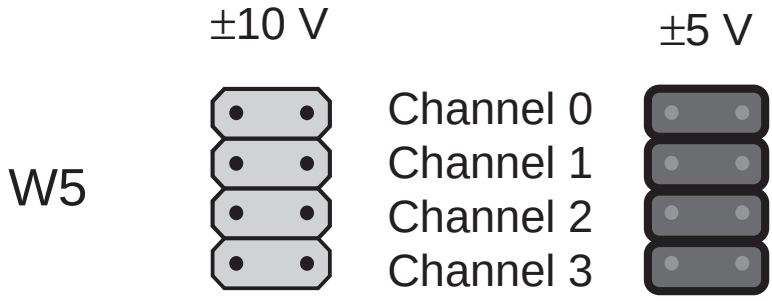


Diagram 4: Setting of input range

2.1.2.4 Output Range

The jumpers W3 (DAC-A) and W4 (DAC-B) are used to set the output ranges for the D/A converters. Ranges of 0…10 V, 0…5 V, and ±5 V are available:

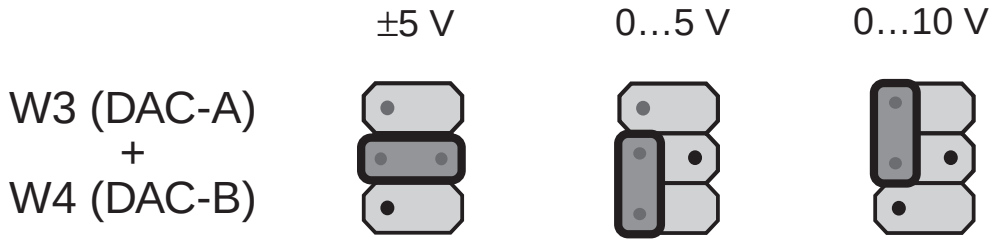


Diagram 5: Setting of output range

2.1.2.5 Default Settings

Function	Jumper/Switches	Settings
Base address	W1	700Hex
Interrupt	W2	IRQ 5
Output range	W3, W4	±5 V
Input range	W5	all channels ±10 V

Table 1: Default settings of the ME-270

2.2 Driver Installation under Windows 95/98/NT

2.2.1 Initial Installation under Windows95/98/NT

To install the driver under Windows 95/98 and Windows NT for the first time, follow this procedure

☛ Insert disk "ME-270 Driver System".

☛ Choose **Run...** from the Windows 95/98/NT start menu. Under the option **Open:** enter the path for the file `SETUP.EXE` on the installation disk, or click **Browse...** Confirm the selection and follow the instructions of the setup program:

⇒ The installation programm will start.

☛ In the window "Install Options", choose "Install a new board" and click OK.

☛ Note the dialogs and keep the settings of base address and interrupt channel ("2.1.2 Jumpers Settings" on page 10) ready for input. **Pay attention to the fact, that the input values have to match the jumper settings** on the board!

⇒ The following files will be installed:

- ❑ Windows95/98 only: Kernel driver `ME270_32.VXD` into path <*Windows-Verzeichnis*>`\SYSTEM`
- ❑ WindowsNT only: Kernel driver `ME270_32.SYS` into path <*Windows-Verzeichnis*>`\SYSTEM32\DRIVERS`
- ❑ API-DLL `ME270_32.DLL` under Windows95/98 into path <*Windows-Verzeichnis*>`\SYSTEM`; under WindowsNT into path <*Windows-Verzeichnis*>`\SYSTEM32`
- ❑ Dialog DLL `MEDLG32.DLL` under Windows95/98 into path <*Windows-Verzeichnis*>`\SYSTEM`; under WindowsNT into path <*Windows-Verzeichnis*>`\SYSTEM32`

⇒ Several files for high level language programming as well as example- and test programs into the directory <*Meilhaus working directory*>`\ME-270` (see also README file on driver disk).

⇒ Registry entries will be made.

☛ Reboot your computer.

⇒ The system driver will be loaded automatically.

⇒ All the boards which were properly installed can be found in the Windows NT diagnostics under **Resources**. The important entries for the ME boards can be found under "IRQ" and "I/O port".

Note:

The ME-270 does not have Plug&Play functionality. Therefore they can not be found in the Windows 95/98 device manager!

2.2.2 Updating the Board Driver

To update the board driver, the same procedure can be followed for Windows 95/98 and Windows NT 4.0:

☛ Insert disk "ME-270 Driver System".

☛ Choose **Run...** from the Windows start menu. Under the option **Open:** enter path and file name for the file `SETUP.EXE` on the installation disk, or click **Browse...** Confirm the selection and follow the instructions of the installation program:

⇒ The installation program will start.

☛ In the windows "Install Options" choose "Update driver and language libraries" and click OK.

⇒ System drivers, API-DLL as well as libraries for programming languages, demo programs and test programs will all be updated.

☛ Reboot your computer.

2.2.3 Changing Board Settings

Use the following procedure to change the settings of base address and interrupt in the Windows registry. Note that the jumper settings on your board have to match the settings in the registry. The same procedure can be followed for Windows 95/98 and Windows NT 4.0:

☛ Insert disk "ME-270 Driver System".

☛ Choose **Run...** from the Windows start menu**.** Under the option **Open:** enter path and file name for the file SETUP.EXE on the installation disk, or click **Browse...** Confirm the selection and follow the instructions of the installation program:

⇒ The installation program will start.

☛ In the window "Install Options" choose "Update settings of an installed board" and click OK.

☛ Note the dialogs and keep the settings of base address and interrupt channel ("2.1.2 Jumpers Settings" on page 10) ready for input. **Pay attention to the fact, that the input values have to match the jumper settings** on the board!

☛ Reboot your computer

Under Windows NT alternativly choose **Settings ➜ System Control ➜ Devices** from Start menu to unload and reload the driver.

2.3 Uninstall

2.3.1 Uninstall a Single Board

The install and uninstall program on your "ME-270 Driver System" disk can be used to remove individual boards of type ME-270 from the Windows registry. The individual software components, e. g. system driver, API-DLL and the libraries for programming languages, will not be removed in this case. The same procedure can be followed for Windows 95/98 and Windows NT 4.0:

☛ Insert disk "ME-270 Driver System".

☛ Choose **Run...** from the Windows start menu**.** Under the option **Open:** enter path and file name for the file SETUP.EXE on the installation disk, or click **Browse...** Confirm the selection and follow the instructions of the installation program:

⇒ The installation program will start.

- ☛ In the window "Install Options" choose "Uninstall a single board" and click OK.
- ☛ Choose the board you would like to remove from the Windows registry.
- ☛ Reboot your computer.

  ⇒ The board will be removed from the registry!

2.3.2 Uninstall the DriverSystem

Please note, that this procedure removes the entire ME-270 driver system from your computer. All files will be removed, including the system driver, the API-DLL (for **all** ME-270 boards installed), the libraries for programming languages, the demo programs and the test programs, which are installed in the "ME-270" subdirectory of "C:\MEILHAUS" (if the default install options were used). The same procedure can be followed for Windows 95/98 and Windows NT 4.0:

- ☛ From the Windows start menu under the path **Settings ➜ System Control ➜ Software** in the property page **Install/Uninstall** choose the "ME-270 Driver Uninstall" option and click "OK".

  ⇒ The entire ME-270 driver system will be removed from your computer!

3 Hardware

3.1 Block Diagram

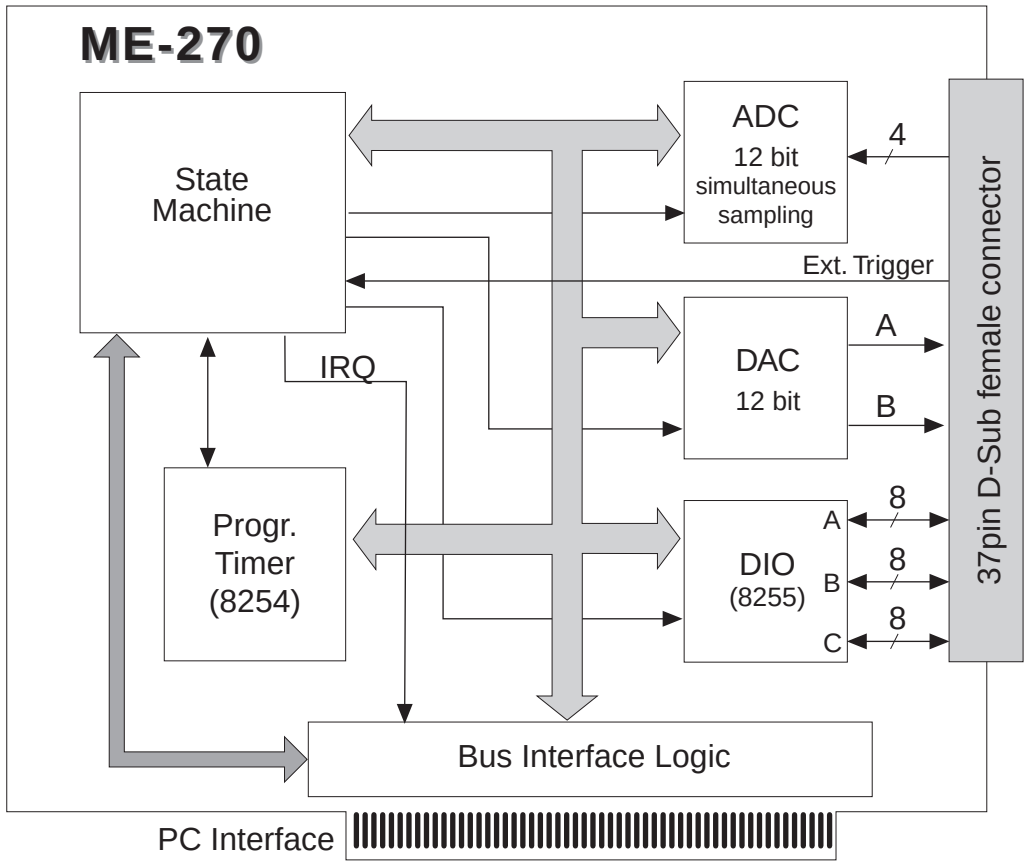


Diagram 6: Block diagram of ME-270

3.2 General Notes

Important Note: The external connections to the board should only be made or removed in a powered down state.

3.3 Operation Modes

3.3.1 Analog Input

The ME-270 has 4 simultaneously sampled analog input channels (single ended). The sample & hold step is integrated into the A/D converter and holds the voltage constant for each channel.

The A/D converter of type AD7874 converts the measured voltage level, depending on the input range (±10 V or ±5 V) into a 12 bit digital value.

Voltage Shape of the A/D Section

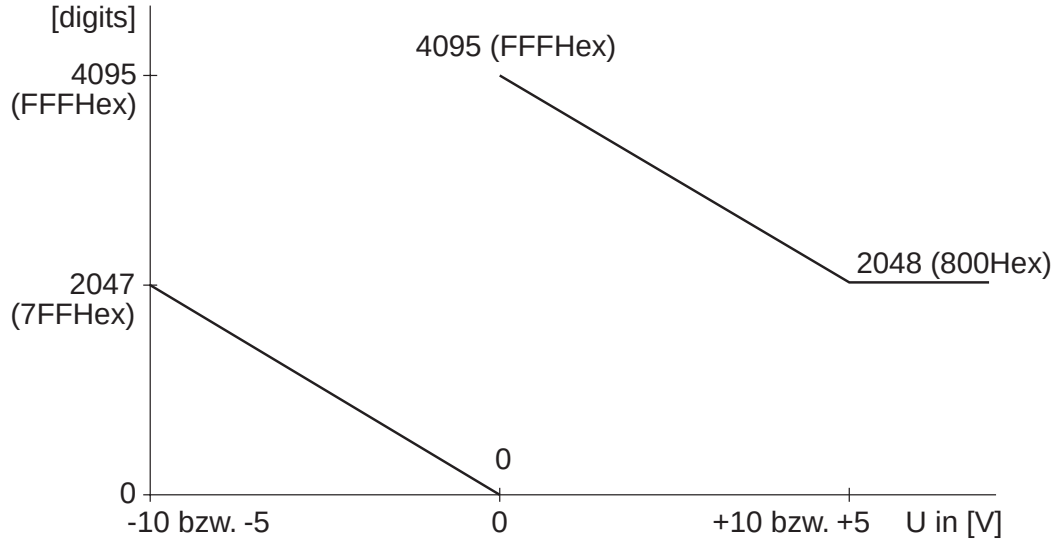


Diagram 7: Digits over input voltage

The A/D converters output inverse non-linearized 12 bit values. In this case the values must be linearized before the values are processed.

Linearization syntax in Delphi: Digits = Digits XOR $7FF

Linearization syntax in C: Digits ^= 0X7FF

The A/D conversion can be triggered by software, an external trigger signal, or by using the on board timer component. After an A/D conversion, the control logic allows an interrupt to be generated on the PC.

3.3.2 Timer

A programmable timer of type 8253 is used to generate a wide range of sampling rates by dividing down the 3 MHz base frequency on the board. This allows synchronised sampling of measured values.

The component has 3 independent 16 bit down counters whose functions are as follows:

- Counters 0, 1 and 2 are cascaded together. The clock input of counter 2 has an input frequency of 3 MHz. The output of counter 2 is connected to the input of counter 1 whose output is connected to the input of counter 0, in effect making a 48 bit divider to allow a wide range of sampling times to be set.

  The total sampling rate f_a is calculated as follows:

  f_a = 3 MHz / (entry to counter 0) x (entry to counter 1) x (entry to counter 2)

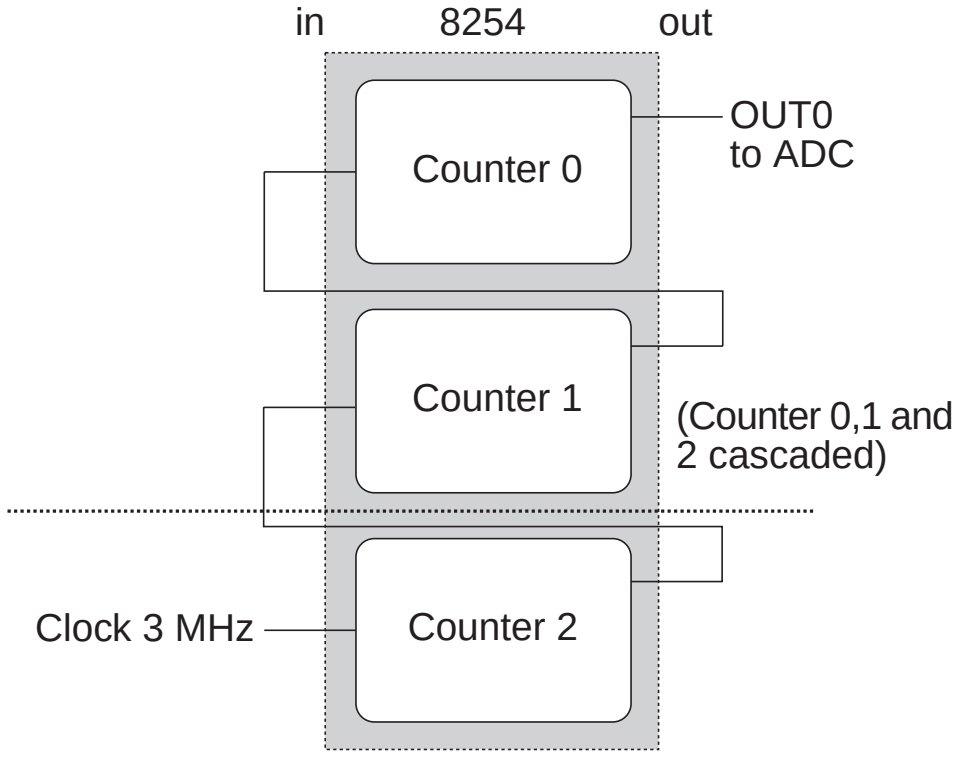


Diagram 8: Timer ME-270

3.3.3 External Trigger

An external trigger signal (TTL level) can be used (pin 10 on the D-Sub connector) to start the A/D conversion. It can be set by software to a rising or falling edge (see function *_me270AISetTrigger*).

3.3.4 Analog Output

The board offers 2 analog outputs each 12 bits at output ranges of 0…5 V, 0…10 V, and ±5 V.

The D/A converter of type AD7237 works linear.

Voltage Shape of the D/A Section

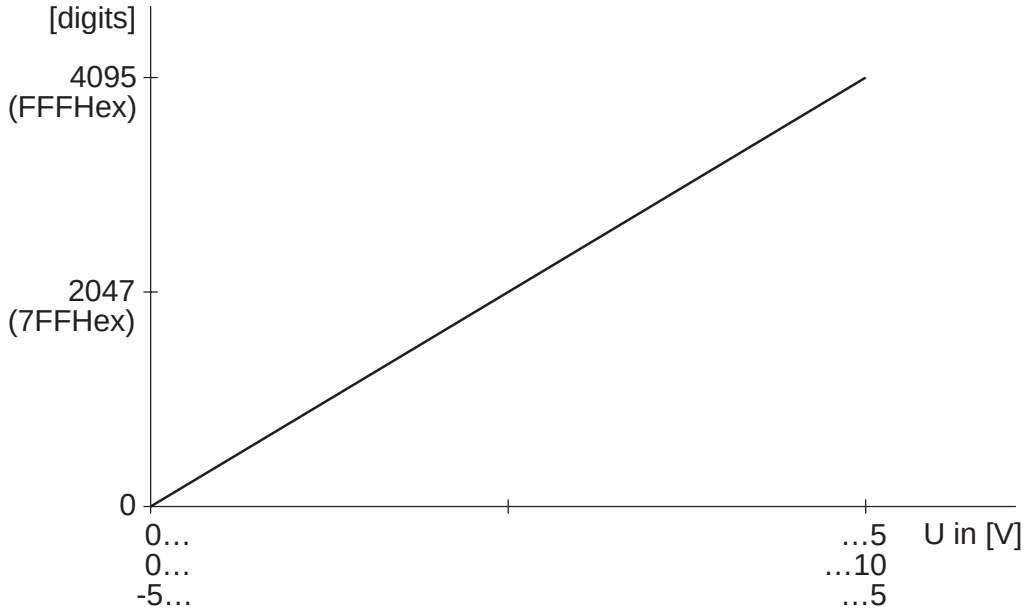


Diagram 9: Output voltage over digits

Attention:

Depending on the selected output ranges (set by jumpers W3 and W4) the outputs will have a voltage corresponding to the value 00Hex. In other words, 0…5 V range will have 0 V on the output, 0…10 V will have 0 V, and ±5 V will have +5 V on the output.

3.4 Digital Input/Output

The digital I/O section is realised using an 8255 (or compatible) PIO component. It offers 3 ports, each 8 bits wide designed for TTL signal levels.

The PIO component 8255, operates in mode 0, „Simple Input/ Output“. The configuration is set by the user with software. The driver software for Windows 95/98/NT delivered with the board can be used to do this as necessary. See section 5.3.4 on page 51.

3.5 Switching

The pinout of the the 37pin female D-Sub connector is shown in Diagram 13: on page 59

Important Note: The external connections to the board should only be made or removed in a powered down state.

3.5.1 A/D Channel Switching

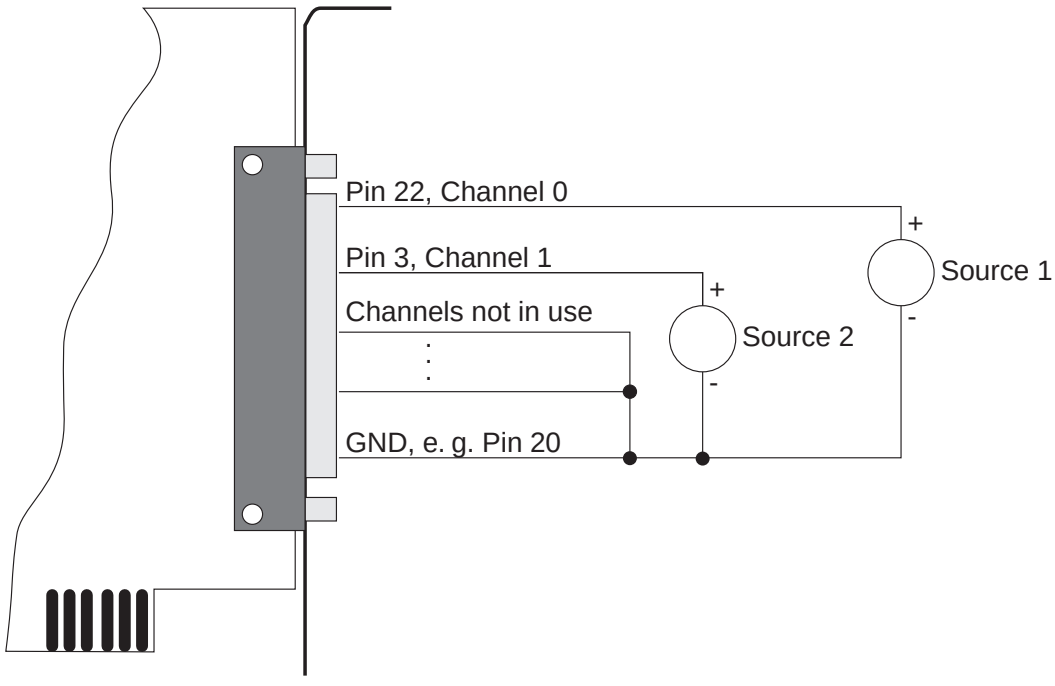


Diagram 10: A/D Channel Switching

The signals are all single ended and therefore use the input signal ground as a reference. All negative lines must therefore be con-

nected to ground (GND: pins 5, 19, and 20). Make sure that no potential differences exist between the ground lines to avoid short circuits on the input lines.

All unused input lines should be connected to ground to avoid cross talk between channels.

3.5.2 D/A Channel Switching

The D/A channels on the ME-270 have a band width limitation caused by the operational amplifiers. This means that in theory, the maximum frequency range of the converter is available. The user however, should limit the bandwidth, depending on the application, to compensate for the distortion of the amplifiers and HF frequency noise which may be caused by the digital I/O section.

The easiest way to implement this is to use an R/C low-pass filter circuit with a 3 dB cutoff frequency which is approximately 10 times the maximum required signal frequency.

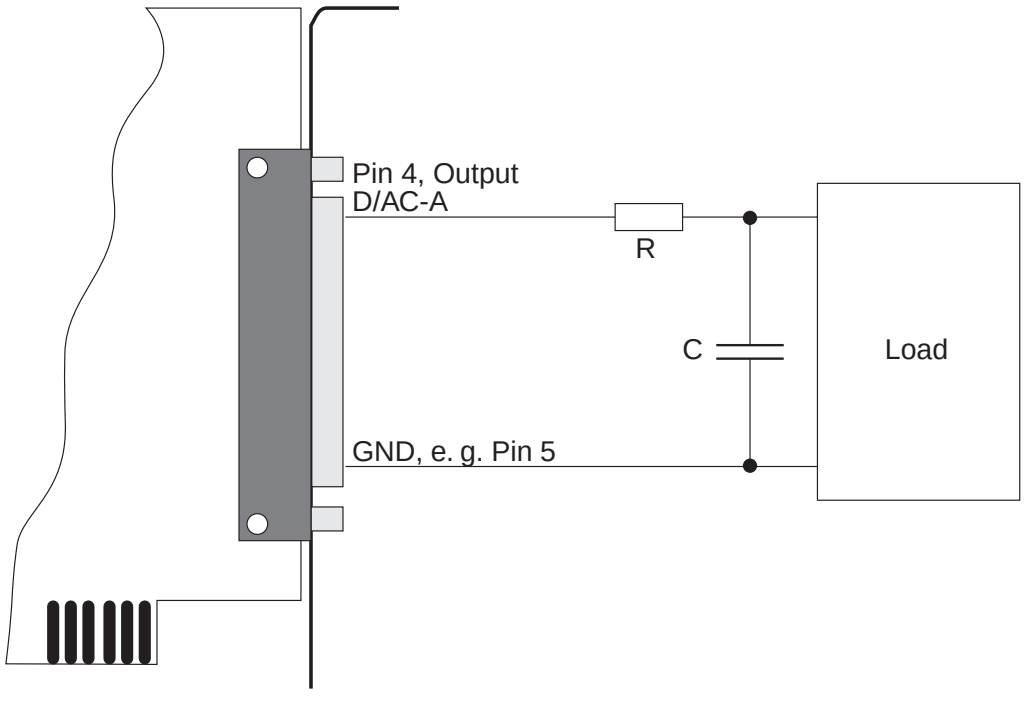


Diagram 11: Switching of analog outputs

All input voltages are with respect to ground (single ended). This means that all negative lines of the measured voltage source must

be connected with ground (pins 1, 16 or 17). The positive lines of the measured voltage source are connected to the desired input channel.

3.5.3 Switching of the Digital-I/Os

All the signal for the digital input/output lines and external trigger signals must be within the TTL signal level specifications, and a connection to ground (e. g. pin 19) must exist.

3.6 Register Description

The ME-270 can be programmed at the register level (e. g. in DOS). The registers are briefly described in this section. We recommend use of the driver software for Windows 95/98 and Windows NT shipped with the board!.

The ME-270 requires 16 consecutive bytes of I/O address space starting at the base address „BA“. The registers are described in the following tables (R = read, W = write)

Offset	8 Bit Registers	
BA+00H	**R**	**M27_FIDREG** contains Infos/ID of the board *b7* . . . . . . 0 must be „0“ *b6* . . . . . . 1 must be „1“ *b5* . . . . . . Functional group „Analog Input“ (AI) *b4* . . . . . . Functional group „Analog Output“ (AO) *b3* . . . . . . Functional group „Digital-I/O“ (DIO) *b2...b0* . . . PROM version the following Function ID results in PROM version 1: 01001001 = 49Hex

Table 2: Address space ME-270

Offset		8 Bit Registers
BA+01H	**W**	**M27_CONTROL** Control register *b7...b5. . .* reserved *b4 . . . . . RST_8255* 1 Reset signal for 8255 (all ports configured for input) *b3 . . . . . TRIGPOL_BIT* 0 Ext. trigger falling edge 1 Ext. trigger rising edge *b2 . . . . . EXIN_BIT* 0 Timer controlled trigger 1 External trigger (pin 10) *b1 . . . . . TRIG_BIT* 0 no trigger used 1 Trigger by counter or ext. trigger *b0 . . . . . ENINT_BIT* 0 Interrupt disabled 1 Interrupt enabled
BA+01H	**R**	**M27_STATUS** Status register *b7...b1. . .* reserved *b0 . . . . . INTR_BIT* 0 during reading the A/D values 1 set after A/D conversion
BA+02H	**W**	**M27_AD_CONV** Start of A/D conversion by writing a dummy value
BA+02H	**R**	**M27_AD_LOW** *b7...b0. . . AD7...AD0* Acquired value low byte

Table 2: Address space ME-270

Offset	8 Bit Registers	
BA+03H	**R**	**M27_AD_HIGH** *b7…b4* . . . reserved *b3…b0* . . . *AD11…AD8* Acquired value high nibble
BA+04H	**R**	**M27_DA_UPDATE** By dummy reading from this register the D/A channels are updated *b7* . . . . . *IENTR* 0 Interrupt line disabled 1 Interrupt line enabled
BA+04H	**W**	**M27_DA_A_L** *b7…b0* . . . Low byte of output value for DAC-A
BA+05H	**W**	**M27_DA_A_H** *b7…b0* . . . High byte of output value for DAC-A
BA+06H	**W**	**M27_DA_B_L** *b7…b0* . . . Low byte of output value for DAC-B
BA+07H	**W**	**M27_DA_B_H** *b7…b0* . . . High byte of output value for DAC-B
BA+08H	**R/W**	**M27_TIMER0** Data register timer 0 *b7…b0* . . . Write low byte first then high byte (always two write cycles necessary!); see 3.6.2

Table 2: Address space ME-270

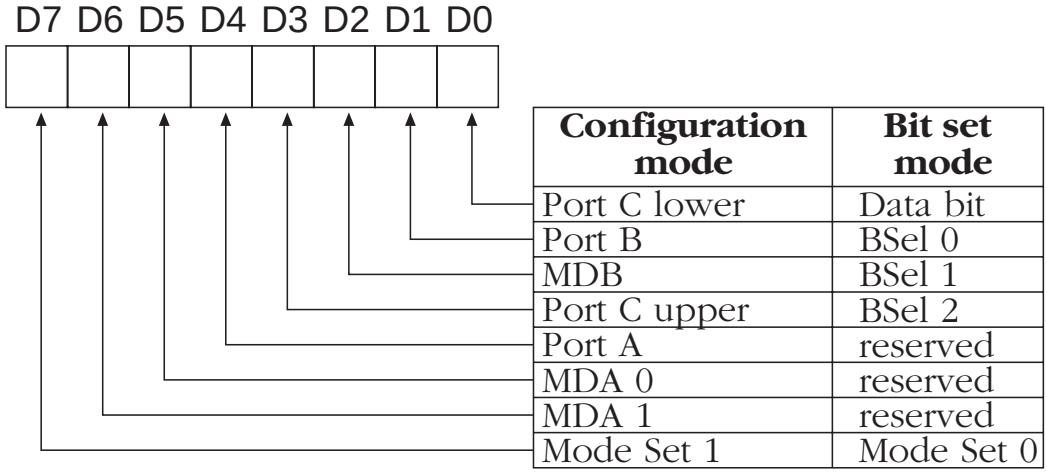
Offset		8 Bit Registers
BA+09H	**R/W**	**M27_TIMER1** Data register timer 1 *b7…b0. . .* Write low byte first then high byte (always two write cycles necessary!); see 3.6.2
BA+0AH	**R/W**	**M27_TIMER2** Data register timer 2 *b7…b0. . .* Write low byte first then high byte (always two write cycles necessary!); see 3.6.2
BA+0BH	**R/W**	**M27_TIMER_CTRL** *b7…b0. . .* Control word for timer component 8254, see 3.6.2
BA+0CH	**R/W**	**M27_DIO_A** *b7…b0. . .* Data register for port A of 8255, see 3.6.1
BA+0DH	**R/W**	**M27_DIO_B** *b7…b0. . .* Data register for port B of 8255, see 3.6.1
BA+0EH	**R/W**	**M27_DIO_C** *b7…b0. . .* Data register for port C of 8255, see 3.6.1
BA+0FH	**R/W**	**M27_DIO_CTRL** *b7…b0. . .* Control word for PIO component 8255, see 3.6.1

Table 2: Address space ME-270

3.6.1 Registers of 8255

The registers of the PIO component can be accessed on register level, e. g. under DOS. For all other cases we recommend the use of the software shipped with the board for Windows 95/98 and NT 4.0, which guarantees full functionallity. (For programming see also chap. 4.4.4 on page 39)

There are 4 user accessible registers on the 8255 chip. Data is exchanged using 3 registers, 8 bits wide (BA+0CH … BA+0EH. The mode of operation is set by an 8 bit control register (BA+0FH) as shown here:



Configuration mode	Bit set mode
Port C lower	Data bit
Port B	BSel 0
MDB	BSel 1
Port C upper	BSel 2
Port A	reserved
MDA 0	reserved
MDA 1	reserved
Mode Set 1	Mode Set 0

Diagram 12: Control word of the 8255

Basically there are 3 modes of operation; however only mode 0 is relevant for the ME-270. For more detailed information about the 8255 consult the manufacturers data sheets.

3.6.1.1 Mode 0 - Simple Input/Output

Mode 0 operation allows simple input and output on all 3 ports. The data is read from or written to the selected port. No handshaking is required. In mode 0 there are three 8 bit ports available. The ports can be independently configured as input (not latched) or output (latched).

In mode 0, 8 different input/output configurations are possible for the ME-270 (the bits 7, 6, 5 and 2 don´t change):

D7	D6	D5	D4	D3	D2	D1	D0
1	0	0	x	x	0	x	x

Table 3: Control word of the 8255

Control word D7...D0	Hex	Port A	Port B	Port C
1 0 0 0 0 0 0 0	$80	Output	Output	Output
1 0 0 0 1 0 0 1	$89	Output	Output	Input
1 0 0 0 0 0 1 0	$82	Output	Input	Output
1 0 0 0 1 0 1 1	$8B	Output	Input	Input
1 0 0 1 0 0 0 0	$90	Input	Output	Output
1 0 0 1 1 0 0 1	$99	Input	Output	Input
1 0 0 1 0 0 1 0	$92	Input	Input	Output
1 0 0 1 1 0 1 1	$9B	Input	Input	Input

Table 4: Port configuration

3.6.2 Registers of the 8254

The registers of the timer component can be accessed on register level, e. g. under DOS. For all other cases we recommend the use of the software shipped with the board for Windows 95/98 and NT 4.0, which guarantees full functionallity. (For programming see also chap. 4.4.5 on page 39)

The control word (BA+0BH) sets the mode of operation and the counting system (BCD or binary) and controls the loading of the count register.

D7	D6	D5	D4	D3	D2	D1	D0
SC1	SC0	RL1	RL0	M2	M1	M0	BCD

Table 5: Control word of the 8254

The counter number is selected with bits SC1/0:

SC1	SC0	Function
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	not allowed

Table 6: Selection of the counter

The Read/Write mode is selected with bits RL1/0:

RL1	RL0	Function
0	0	Counter latching operation
1	0	only MSB
0	1	only LSB
1	1	first LSB, second MSB (2 x 8 bit)

Table 7: Selection of the read/write mode

The mode of operation for the individual counters is selected with bits M0 .. M2:

M2	M1	M0	Function
0	0	0	Mode 0
0	0	1	Mode 1
X	1	0	Mode 2
X	1	1	Mode 3
1	0	0	Mode 4
1	0	1	Mode 5

Table 8: Selection of the operation mode

The counting system for the individual counters is selected with the BCD bit:

BCD	Function
0	16 bit binary counter
1	BCD counter

Table 9: Selection of the counter mode

First, the counter is initialised by writing the control word to the appropriate register (BA+0BH). This selects the counter, sets the read/write mode, counter system and mode of operation. The start value is then loaded into the appropriate count register (BA+08H…BA+0AH). The counter is decremented by 1 on every falling edge of Clk-signal. The ME-270 is only used in mode 2 which is briefly described next.

Mode 2: Asymmetric divider

In this mode, the counter functions as a frequency divider. The counter output (Out 0…2) is set high after initialisation. When the counter is enabled with a high level on the Gate input, the counter begins counting downwards and the output remains high. When the count value reaches 0001Hex, the output goes low for one clock cycle. This process repeats as long as the Gate signal is high. If the Gate signal is not kept high, the output will immediately go to the high state.

If the counter is reloaded between two output pulses, the count at that moment is not effected. The new value is used on the following period.

3.7 Test Program

For test issues a test program is provided. The appropriate test program can be found in a corresponding subdirectory of `C:\MEILHAUS\` (Default) and can be run by double clicking on the file. (Condition: system driver correctly installed).

3.8 Balancing

The board is delivered in a fully functional and balanced condition. If it should become necessary to balance the board, please send the board to the Meilhaus Electronic Service Department, see „Service address” on page 61.

4 Programming

4.1 High Level Language Programming

The following high level languages are supported by standard:

- Visual C++ (version 4.0 or higher). Please read the notes in the respective README files
- Delphi (version 2.0 or higher). Please read the notes in the respective README files.
- Visual Basic (version 4.0 or later). Please read the notes in the respective README files.
- for further high level languages see the respective README files on your driver disk(s).

Note: The compilers and linkers require the correct paths to be set to the corresponding files in the high level languages.

By linking the high level language specific definition files into your project you can pass many macros and parameters in the form of predefined constants. As an alternative, you can pass the matching Hex value at any time.

4.1.1 Example Programs

We have provided simple demo programs and small projects with source code to help understanding of the functions and how to incorporate them into your project. These demo programs are installed automatically to appropriate subdirectories of `C:Meilhaus\` (Default). Please read the notes in the appropriate README files.

4.2 HP VEE Programming

HP VEE components for the ME-270 are delivered on (a) seperate disk(s) included with HP VEE from Meilhaus Electronic. The ME-270 is supported by HP VEE full versions 3.2 or higher under Windows 95/98/NT. For installation of HP VEE components and for further infos please note the documentation on the disks delivered with the HP VEE driver system. For basics of HP VEE programming please use your HP VEE documentation and the HP VEE online help index.

4.2.1 User Objects

For convenient use of the driver, predefined „User Objects“ have been developed which internally call API functions. They can be called by the additional menu item „ME Board“ and be included in the HP VEE development environment. They can be placed and „wired“ in your application the same as standard HP VEE objects.

The User Objects are self descriptive and based on the API functions documented in the chapter „Function Reference“. Additionally there are some „Expanded User Objects“ for making programming as easy as possible for you. A short description of every UO is also available under the item „Description“, if you move the cursor over the UO and push the right mouse button.

The UOs can be changed any time for user requirements and can be saved as a user specific object.

4.2.2 HP VEE Example Programs

For demonstration purposes and for easier understanding, demo programs using the important UOs have been written. They can be called by the menu item „ME Board – Demos“.

The HP VEE demo programs contain partial additions to the „normal“ UOs and for differentiation from the „normal“ UOs the prefix "x…" in their file name is used.

4.2.3 The "ME Board"-Menu

The installation program automatically expands the HP VEE menu by the „ME Board“ entry. It enables a convenient use of all driver functions available in HP VEE. By the „ME Board“ menu you can call the driver and demo User Objects sorted by board families.

Note:
The UOs installed, depend on the selected board family at the beginning of your HP VEE driver installation. If you call UOs under the „ME Board“ menu which are not installed, an error message occures:

File ´*filename*´ was not found. Error number: 700

If necessary, you can install the additional HP VEE components any time. To do this use the disk „Meilhaus HP VEE Driver System“.

4.3 LabVIEW™-Programming

LabVIEW™ components for the ME-270 are delivered optional on (a) seperate disk(s) from Meilhaus Electronic. The ME-270 is supported by LabVIEW™ full versions 4.x or higher under Windows 95/98/NT. For installation of LabVIEW™ driver components and for further infos please note the documentation on the disks deliverd with the ME-270 LabVIEW™ driver. For basics of LabVIEW™ programming please use your LabVIEW™ documentation and the LabVIEW™ online help index.

4.3.1 Virtual Instruments

For convenient use of the driver, predefined „Virtual Instruments“ have been developed. They can be called by the additional menu item „File - Open“ and be included in the LabVIEW™ development environment. They can be placed and „wired“ in your application the same as standard LabVIEW™ objects.

The source VIs are self descriptive and based on the API functions documented in the chapter „Function Reference“ on

page 41. Additionally there are some „Expanded Virtual Instruments“ for making programming as easy as possible for you.

A short description of every VI is also available in the VI „ME-270 Function Tree“. This VI is only for documentation purposes and can be opened by the menu „File - Open“. Under „Description“ you find a short description of every Virtual Instrument.

The VIs can be changed any time for user requirements and can be saved as a user specific VI.

4.3.2 LabVIEW™ Example Programs

For demonstration purposes and for easier understanding, demo programs using the important virtual instruments (VIs) have been written. They can be called by the menu item „File - Open“.

4.4 Register Programming

Note: The ME-270 can also be programmed at the register level (e. g. in DOS). This can be done with high level language input and output port commands, see „Register Description” on page 24. Consult the manuals for the high level language of your choice for the proper commands and syntax. We recommend however, the use of the driver software for Windows 95/98 and Windows NT, supplied with the board!

4.4.1 Initialisation

- Read the ID-register (M27_FIDREG). The higher 5 bits of the lower byte (b7…b3) have to contain the value '01111' while the lower 3 bits (b2…b0) will depend on the PROM version. For example, for the PROM version 1 (001), the value in the ID register will be:

  01111001Bin = 79Hex

  If this is not the case, either the base address is not correct or the board is defective.

- To stop all functions on the board: set the control register (M27_CONTROL) to 10Hex

4.4.2 A/D Conversion

A/D Conversion Start

The A/D conversion process can be started in 3 different ways depending on the setting of bits 2 and 1 (EXIN_BIT and TRIG_BIT) in the control register BA + 01Hex as shown in the table below:

EXIN_BIT	TRIG_BIT	Start mode
0	0	Start by software
0	1	Start by internal trigger (timer)
1	1	Start by external trigger

Table 10: Bits 1 and 2 of control register

Simple A/D Conversion

(conversion is started by software/polling)

- Initialise the board (see 4.4.1)
- Set the following bits in the control register M27_CONTROL:
  - Bit 0 (ENINT_BIT): '0' (no interrupt)
  - Bit 1 (TRIG_BIT): '0' (no trigger)
  - Bit 2 (EXIN_BIT): '0' (no ext. trigger)
  - Bit 3 (TRIG_POL_BIT): '0'
  - Bit 4 (RST_8255_BIT): '1' (Reset the 8255)
  - Bit 5…7: reserved
- Start the conversion by writing a dummy value to the register M27_AD_CONV.
- Wait until the A/D conversion is completed: Check bit 0 of the M27_STATUS register until it is set to '1'.
- Read in the result of the A/D conversions on all 4 channels. The low byte is read in first (M27_AD_LOW), then the high byte (M27_AD_HIGH). The first value is for channel 0, the second for channel 1 etc.

Attention:
The bit INTR_BIT in the status register M27_STATUS is only reset when all 4 channels have been read in. No new A/D conversion can take place until this is done.

Interrupt Controlled A/D Conversion

- Initialise the board (see 4.4.1)
- For interrupt operation, set the following bits in the control register M27_CONTROL.
  - Bit 0 (ENINT_BIT): '1' (enable interrupt)
  - Bit 1 (TRIG_BIT): '1' (use a trigger)
  - Bit 2 (EXIN_BIT): '0' (timer controlled start of conversion)
  - Bit 3 (TRIG_POL_BIT): '0' or '1' (for a rising or falling edge)
  - Bit 4 (RST_8255_BIT): '1' (Reset the 8255)
  - Bit 5…7: reserved
- Load the timer (see 4.4.5): The output of timer 0 starts the A/D conversion (see also „3.6.2 Registers of the 8254“ on page 29).
- Initialise the interrupt: The interrupt routine writes the converted values into a data field.
- Read the data field (all 4 values) thus resetting bit 0 of the status register.
- Enable the interrupt by setting bit 0 (ENINT_BIT) of the control register M27_CONTROL to a '1'

4.4.3 D/A Conversion

(Example is for channel A)

- Set jumpers W3 and W4 to the desired range before building the board into the PC (0…5 V, 0…10 V, ±5 V, see „Analog Output” on page 20)
- Initialise the board (see 4.4.1)

- Load the D/A converter:
  - Write data for channel A low (M27_DA_A_L)
  - Write data for channel A high (M27_DA_A_H)
- Set the output by reading the register M27_DA_UPDATE

4.4.4 Digital Input/Output

- Initialise the board (see 4.4.1)
- Set the direction of the port A, B, and C of the 8255: Write the control word into the M27_DIO_CTRL register (see „Port configuration" on page 29)
- Read or write to/from the ports using the M27_DIO_A, M27_DIO_B, and M27_DIO_C registers.

4.4.5 Timer Configuration

The control registers of counter 0, 1 and 2 must be initialized and the data registers loaded. Note the following order of operation:

- Write control word for counter 0 to control register M27_TIMER_CTRL
- Write the low byte to data register of counter 0 (M27_TIMER0)
- Write the high byte to data register of counter 0 (M27_TIMER0)
- Write control word for counter 1 to control register M27_TIMER_CTRL
- Write the low byte to data register of counter 1 (M27_TIMER1)
- Write the high byte to data register of counter 1 (M27_TIMER1)
- Write control word for counter 2 to control register M27_TIMER_CTRL
- Write the low byte to data register of counter 2 (M27_TIMER2)

- Write the high byte to data register of counter 2 (M27_TIMER2)

5 Function Reference

5.1 Functional Overview of the 32 Bit Driver

The 32 bit driver for the ME-270 is written for the Windows 95/98 and Windows NT operating system. It consists of the following files:

- VxD driver (`ME270_32.VXD`) for Windows 95/98, will be loaded dynamically.
- Kernel driver (`ME270_32.SYS`) for Windows NT will be loaded automatically on power up.
- API-DLL (`ME270_32.DLL`) with the API functions for the ME-270.
- Dialogue DLL (`MEDLG32.DLL`) with dialogue functions.

The installation program for the ME-270 automatically registers every new board of the ME-270 board family with the operating system. The driver supports up to 4 boards of one board family (iBoardNumber 0…3) and up to 12 boards as a whole.

Upon every start, the driver searches for registered boards but does not check the physical existence (the ME-270 is not Plug&Play compatible). After the driver is successfully loaded, the board can be accessed via the API functions.

The API functions allow convenient access to the hardware. Every function that accesses the ME-270 requires an integer value for identification of the board. This integer value will be indicated by `<iBoardNumber>` in the following API function descriptions.

5.2 Naming Conventions

The API functions were written specially for the ME-270 board family. For Visual C and Delphi (Pascal) every API function starts with an underscore „_“ (not so in Borland C and BASIC).

The function names were selected to be as descriptive as possible. Each function name consists of a board type specific prefix

and several elements which stand for the corresponding sections (e. g. "AI" for "Analog In" i. e. the A/D section).

_me270… Function valid for board type ME-270

For the description of the functions, the following standards will be used:

function name	will be italic in body text e. g. *_me270AIScan*
<parameters>	will be in brackets as shown and in font Courier
<*variables*>	will indicate a predefined constant and will be written in italic text and in brackets as shown
[square brackets]	variables in square brackets have to be used depending on board type.
FILE NAMES	or PATHS will be capitalized in font Courier
me270…()	parts of programs will be in Courier type

To identify data types, the following letters will be used:

i… or dw…	32 bit integer value
s… or w…	16 bit short value
c… or b…	8 bit character value
p…	pointer of data type (i, s, l or c)

5.3 Description of the API Functions

The functions will be described by functional groups as listed below. Within each functional group, the individual functions will be described in alphabetical order:

Function	Short Description	Page
General Functions		
_me270GetDLLVersion	Determine DLL version number	44
_me270PROMVersion	Determine PROM-ID	44
Analog Input		
_me270AIScan	Timer controlled conversion	45
_me270AISetTimer	Loading the timer	47
_me270SetTrigger	Configuring trigger source and trigger edge	48
_me270AISingle	Acquiring single value	49
Analog Output		
_me270AOSingle	Simultaneous analog output	50
Digital Input/Output		
_me270DIGetBit	Reading a single bit	51
_me270DIGetByte	Reading a byte	52
_me270DIOSetPortDirection	Configures a port as an input or output	53
_me270DOSetBit	Writing a single bit	54
_me270DOSetByte	Writing a byte	55
Error Handling		
_me270GetDrvErrMess	Error string corresponding to error code	56

Table 11: Overview of library functions

5.3.1 General Functions

_me270GetDLLVersion

✎ Description

Function is used for: ME-270.

Determines the version number of the driver DLL.

● Definitions

C: int _me270GetDLLVersion();

Delphi: Function _me270GetDLLVersion: integer;

Basic: Declare Function me270GetDLLVersion Lib "me270_32" Alias "_VBme270GetDLLVersion@0" () As Long

→ Parameters none

‹ Return value

The version number is returned as a 32 bit value. The upper 16 bits contain the main version number and the lower 16 bits contain the sub version number. E. g.: 00010003Hex indicates the version number 1.03

_me270PROMVersion

✎ Description

Function is used for: ME-270.

Returns the board identification and the PROM-ID.

● Definitions

C: int _me270PROMVersion (int iBoardNumber, int *piVersion;)

Delphi: Function _me270PROMVersion (iBoardNumber: integer; Var iVersion: integer): integer;

Basic: Declare Function me270PROMVersion Lib "me270_32" Alias "_VBme270PROMVersion@8" (ByVal iBoardNumber As Long, ByRef iVersion As Long) As Long

➔ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<Version>` Pointer to an integer value with the encoded PROM version. The value is interpreted as a hexadecimal value. Only the lower 8 bits are significant. The meaning of the individual bits is described in the register description (FID register, BA+00Hex).

If an error occurs, the function is still executed but an invalid version number (FFHex) is returned.

❮ Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

5.3.2 Analog Input

_me270AIScan

✎ Description

Function is used for: ME-270

This function starts the scanning process and continues until the defined number of scans have been completed. When the entire process is completed, the values are transferred to the application.

☞ Important Note!

This function may only be called after the functions *_me270AISetTrigger* and *_me270AISetTimer* have been successfully run at least once!

Example in C:

```
...
if(_me270AISetTrigger(...) &&
   _me270AISetTimer(...))
_me270AIScan(...);
...
```

● Definitions

C: int _me270AIScan (int iBoardNumber, int iFromChannel, int iToChannel, int *piNumberOfScans, short *psArray);

Delphi: Function _me270AIScan (iBoardNumber, iFromChannel, intToChannel: integer, Var iNumberOfScans: integer; Var sArray: smallint): integer;

Basic: Declare Function me270AIScan Lib "me270_32" Alias "_VBme270AIScan@20" (ByVal iBoardNumber As Long, ByVal iFromChannel As Long, ByVal iToChannel As Long, iNumberOfScans As Long, sArray as Integer) As Long

➔ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<NumberOfScans>` Number of scans; one scan is the single simultaneous conversion of the channels 0…3

`<Array>` Pointer to the array of 16 bit values. Only the lower 12 bits are significant and contain the linearized measured values. A decimal value of 0 represents the lowest value and a decimal value of 4095 represents the highest value. This will depend on the range selected.

Array[n] is a pointer to the value field for channel n with a size of `<NumberOfScans>`. The array has a total size of `<NumberOfScans>` multiplied by the number of channels being sampled.

‹ Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270AISetTimer

✎ Description

Function is used for: ME-270

Loads the timer values for the 8253 compatible timer on the board. Timers 0, 1 and 2 are cascaded together. The output of timer 0 is used to synchronise the start of A/D conversion.

● Definitions

C: int _me270AISetTimer (int iBoardNumber, int iTimer0, int iTimer1, int iTimer2);

Delphi: Function _me270AISetTimer (iBoardNumber, iTimer0, iTimer1, iTimer2: integer): integer;

Basic: Declare Function me270AISetTimer Lib "me270_32" Alias "_VBme270AISetTimer@16" (ByVal iBoardNumber As Long, ByVal iTimer0 As Long, ByVal iTimer1 As Long, ByVal iTimer2 As Long) As Long

→ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<Timer0…2>` Values for timers 0, 1, and 2. Possible values are: 2…65535 (02Hex…FFFFHex).

The resulting sampling frequency is determined by: frequency = 3 MHz /(timer 0 x timer 1 x timer 2)

‹ Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270AISetTrigger

✎ Description

Function is used for: ME-270

Configures the trigger source and trigger polarity.

● Definitions

C: int _me270AISetTrigger (int iBoardNumber, int iModePolarity);

Delphi: Function _me270AISetTrigger (iBoardNumber, iModePolarity: integer): integer;

Basic: Declare Function me270AISetTrigger Lib "me270_32" Alias "_VBme270AISetTrigger@8" (ByVal iBoardNumber As Long, ByVal iModePolarity As Long) As Long

➔ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<ModePolarity>` Mode and polarity of triggers.

<ModePolarity>	*Hex value*
AI270_TRIGGER_TIMER	(00Hex)
AI270_TRIGGER_EXT_LOW	(50Hex)
AI270_TRIGGER_EXT_HIGH	(10Hex)

‹ Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess.*

_me270AISingle

✎ Description

Function is used for: ME-270

Calling this function automatically starts a scan process i. e. a single simultaneous sampling of the channels 0…3 in polling mode without interrupt, and the result is then returned. There is no external trigger support for this function and no configuration is required.

● Definitions

C: int _me270AISingle (int iBoardNumber, short *psArray);

Delphi: Function _me270AISingle (iBoardNumber: integer; Var sArray: smallint): integer;

Basic: Declare Function me270AISingle Lib "me270_32" Alias "_VBme27AISingle@8" (ByVal iBoardNumber As Long, sArray As Integer) As Long

→ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<Array>` Pointer to an array of four 16 bit values. Only the lower 12 bits are significant and contain the linearized measured values. A decimal value of 0 represents the lowest value and a decimal value of 4095 represents the highest value of the input range selected. By using the pointer Array[chan], the individual measured values can be accessed, where „chan“ is the channel number 0…3.

‹ Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

5.3.3 Analog Output

_me270AOSingle

✎ Description

Function is used for: ME-270

This function outputs the voltage values to both D/A converters simultaneously.

● Definitions

C: int _me270AOSingle (int iBoardNumber, short iChan0Value, short iChan1Value);

Delphi: Function _me270AOSingle (iBoardNumber: integer; sChan0Value, sChan1Value: smallint): integer;

Basic: Declare Function me270AOSingle Lib "me270_32" Alias "_VBmeAOSingle@12" (ByVal iBoardNumber As Long, ByVal sChan0Value As Integer, ByVal sChan1Value As Integer) As Long

→ Parameters

<BoardNumber> Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

<Chan0Value> Value to be output for channel 0; possible values are: 0…4095 (0000Hex…0FFFHex) where the decimal value 0 is the lowest value and decimal 4095 the highest value of the chosen output range

<Chan1Value> Value to be output for channel 1; possible values are: 0…4095 (0000Hex…0FFFHex) where the decimal value 0 is the lowest value and decimal 4095 the highest value of the chosen output range

‹ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

5.3.4 Digital I/O

_me270DIGetBit

✎ Description

Function is used for: ME-270

This function determines the status of a single input line.

☞ Important note!

Before calling this function, the direction must be set by calling *_me270DIOSetPortDirection*.

● Definitions

C: int _me270DIGetBit (int iBoardNumber, int iPortNo, int iBitNo, int *piBitValue);

Delphi: Function _me270DIGetBit (iBoardNumber, iPortNo, iBitNo: integer; Var iBitValue: integer): integer;

Basic: Declare Function me270DIGetBit Lib "me270_32" Alias "_VBme270DIGetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByRef iBitValue As Long) As Long

➔ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<PortNo>` Port name, possible values:

<PortNo>	*Description*
PORTA (00Hex)	Port A
PORTB (01Hex)	Port B
PORTC (02Hex)	Port C

`<BitNo>` Number of input line to be read; possible values: 0…7 corresponding bits 0…7 of a port

`<BitValue>` Pointer on an integer value which corresponds with the line status; possible return values:

0: Line is set to '0'
1: Line is set to '1'

❮ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270DIGetByte

✎ Description

Function is used for: ME-270

This function reads a byte (8 bits) from a port defined as input.

☞ Important note!

Before calling this function, the direction must be set by calling *_me270DIOSetPortDirection*.

● Definitions

C: int _me270DIGetByte (int iBoardNumber, int iPortNo, int *piValue);

Delphi: Function _me270DIGetByte (iBoardNumber, iPortNo: integer; Var iValue: integer): integer;

Basic: Declare Function me270DIGetByte Lib "me270_32" Alias "_VBme270DIGetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, iValue As Long) As Long

→ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<PortNo>` Port name, possible values:

<PortNo>	*Description*
PORTA (00Hex)	Port A
PORTB (01Hex)	Port B
PORTC (02Hex)	Port C

`<Value>` Pointer on an integer value which contains the input byte; only the lower 8 bits are significant.

‹ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270DIOSetPortDirection

✎ Description

Function is used for: ME-270

This function sets the direction of a digital port as input or output.

☞ Important note!

This function must be called before any bit/byte read/write operation can be done. It must be called separately for each port.

● Definitions

C: int _me270DIOSetPortDirection (int iBoardNumber, int iPortNo, int iDir);

Delphi: Function _me270DIOSetPortDirection (iBoardNumber, iPortNo, iDir: integer): integer;

Basic: Declare Function me270DIOSetPortDirection Lib "me270_32" Alias "_VBme270DIOSetPortDirection@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iDir As Long) As Long

➔ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<PortNo>` Port name, possible values:

<PortNo>	*Description*
PORTA (00Hex)	Port A
PORTB (01Hex)	Port B
PORTC (02Hex)	Port C

`<Dir>` Port direction; possible values:

<Dir>	*Description*
MEINPUT (00Hex)	Input port
MEOUTPUT (01Hex)	Output port

❮ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270DOSetBit

✎ Description

Function is used for: ME-270

Sets a single digital output line to 0 or 1.

☞ Important note!

Before calling this function, the direction must be set by calling *_me270DIOSetPortDirection*.

● Definitions

C: int _me270DOSetBit (int iBoardNumber, int iPortNo, int iBitNo, int iBitValue);

Delphi: function _me270DOSetBit (iBoardNumber, iPortNo, iBitNo, iBitValue: integer): integer;

Basic: Declare Function me270DOSetBit Lib "me270_32" Alias "_VBme270DOSetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByVal iBitValue As Long) As Long

→ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<PortNo>` Port name, possible values:

<PortNo>	*Description*
PORTA (00Hex)	Port A
PORTB (01Hex)	Port B
PORTC (02Hex)	Port C

`<BitNo>` Number of output line to be set; possible values: 0…7 corresponding bits 0…7 of a port

`<BitValue>` Possible values:
= 0: Bit set to '0'
> 0: Bit set to '1'

‹ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

_me270DOSetByte

✎ Description

Function is used for: ME-270

Writes a byte to a digital output.

☞ Important note!

Before calling this function, the direction must be set by calling *_me270DIOSetPortDirection*.

● Definitions

C: int _me270DOSetByte (int iBoardNumber, int iPortNo, int iValue);

Delphi: function _me270DOSetByte (iBoardNumber, iPortNo, iValue: integer): integer;

Basic: Declare Function me270DOSetByte Lib "me270_32" Alias "_VBme270DOSetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iValue As Long) As Long

→ Parameters

`<BoardNumber>` Board number for 1., 2., 3. or 4. installed boards of type ME-270; possible values: 0…3

`<PortNo>` Port name, possible values:

<PortNo>	*Description*
PORTA (00Hex)	Port A
PORTB (01Hex)	Port B
PORTC (02Hex)	Port C

`<Value>` Output value; possible values: 00…FFHex (0…255).

‹ Return Value

If the Function is successfully executed, a '1' is returned. If an error occurs, a '0' Is returned. The cause of the error can be determined with the function *_me270GetDrvErrMess*.

5.3.5 Error Handling

_me270GetDrvErrMess

✎ Description

Function is used for: ME-270

If an error occurs during the processing of the previously called API function of the driver, this routine returns the matching error code and text.

☞ Important Note!

This function may only be called, if the previously called API function of the `ME270_32.DLL` returned an error code (i. e. error code 0)!

● Definitions

C: int _me270GetDrvErrMess (char *pcErrortext);

Delphi: Function _me270GetDrvErrMess (Var errortext: errorstring): integer;

Basic: Declare Function me270GetDrvErrMess Lib "me270_32" Alias "_VBme270GetDrvErrMess@4" (ByVal errortext As String) As Long

➔ Parameters

`<Errortext>` Pointer to a string; the return value is the error code.

‹ Return value

The function always returns the DLL global variable `<DLLErrorCode>`

Appendix

A Specifications

PC Interface

Type	ISA 8 bit
Base address	0000…1FE0Hex in 20Hex steps (jumper)
Address space	16 byte
Interrupt	2, 3, 5, 7 (jumper)

Analog Inputs

Number	4 single ended, simultaneous sampling
Type	AD7874
Resolution	12 bit
Sampling rate	29 kHz over all 4 channels
Input ranges	±5 V, ±10 V for all channels (selectable by jumper)
Input impedance	200 kΩ/20 pF (±10 V range), 100 kΩ/20 pF (±5 V range)
Rel. accuracy	±1 bit
Diff. non linearity	max. ±5 LSB
Offset/gain	separatly adjustable for each channel
Trigger modes	Timer controlled, by software or external: programmable for rising or falling edge
Ext. Trigger level	TTL level (see Digital-I/O Specs.)

Analog Output

Number	2 independent converters
Type	AD7237
Resolution	12 bit
Linearity error	max. ±6 LSB
Offset error (unipolar)	max. ±3 LSB
Zero error (bipolar)	max. ±6 LSB
Settling time	max. 8 µs
Output rate	max. 50 kHz (abhängig von Anwender-software und Betriebssystem)
Output current	2 mA
Output ranges	0…5 V, 0…10 V, ±5 V for each channel (by jumper)
Operation mode	Simultaneous output to both channels (when using the 32 bit driver)

Digital Input/Output

Type	82(C)55 in mode 0
Number	24, TTL level (three 8 bit ports)
Input voltage	Low: -0,5 V…+0,8 V (I_{ILmax} =±10 µA)
	High: +2,0 V…+5,5 V (I_{IHmax} =±10 µA)
Output voltage	Low: max. +0,45 V (I_{OL}=+2,5 mA)
	High: min. +2,4 V (I_{OH}=-2,5 mA)

Timer

Type	82(C)54 or compatible
Number	3 x 16 bit used as a frequency divider, for control of A/D section cascaded by hard-ware

Source frequency for divider 3 MHz connected to counter 2

General Information

Power consumption	typ. 0,7 A (ohne Last)
DC/DC converter	±15 V/2 W
Physical size	100 x 112 mm (without mounting bracket and connector)
Connectors	37pin D-Sub female
Operating temperature	0…70˚C
Storage temperature	0…50°C
Relative humidity	20…55% (not condensing)

CE Certification

EMC Directive	89/336/EMC
Emission	EN 55022
Noise immunity	EN 50082-2

B Pinout

B1 37pin D-Sub female connector

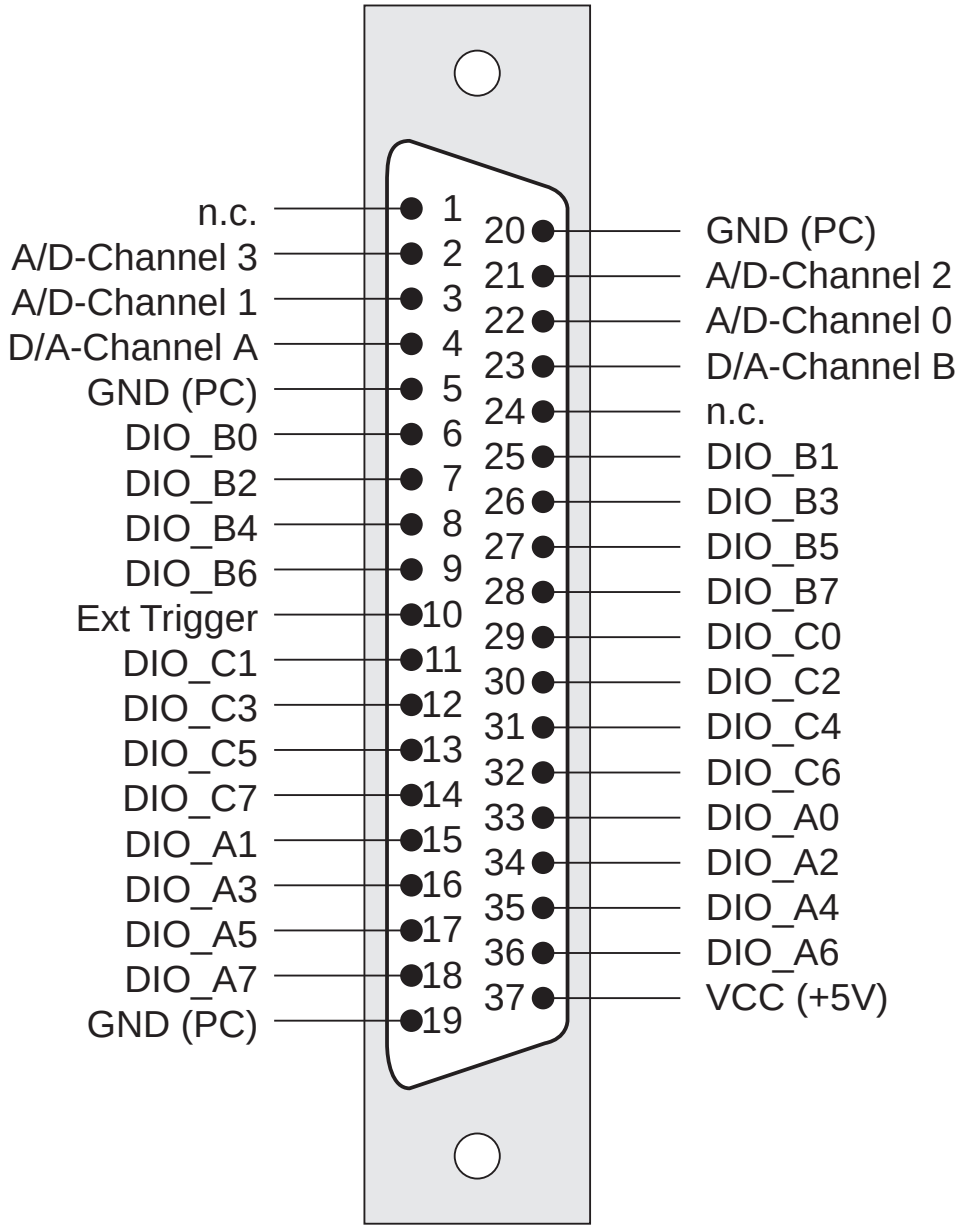


Diagram 13: 37pin D-Sub female connector

C Accessories

Optionally the following products are available:

ME-AB-D37 M

37pin D-Sub connector block (female)

ME-AK-D37

37pin D-Sub cable (male - female), 2 m

D Technical Questions

D1 Hotline

If you should have any technical questions or problems with the board hardware or the driver software, please send us a fax to the following number:

Fax hotline: ++ 49 (0) 89/89 01 66 28

or via e-mail:

Hardware support: support@meilhaus.de

Software support: software@meilhaus.de

Please give a full description of the problems and as much information as possible, including operating system information.

D2 Service address

We hope that your board will never need to be repaired. If this should become necessary please contact us at the following address:

Meilhaus Electronic GmbH

Service Department
Fischerstraße 2
D-82178 Puchheim/Germany

If you would like to send a board to Meilhaus Electronic for repair, please do not forget to add a full description of the problems and as much information as possible, including operating system information.

D3 Driver Update

The current driver versions are available around the clock on our ftp site. Access to the ftp site is password protected. To obtain access to the Meilhaus ftp site we ask that you register your company. This can be done on our homepage (http://www.meilhaus.com). We will fax or e-mail you the correct password to access the ftp site. For urgent requests, please give us a call directly at ++49 (0) 89/89 01 66-0.

E Index