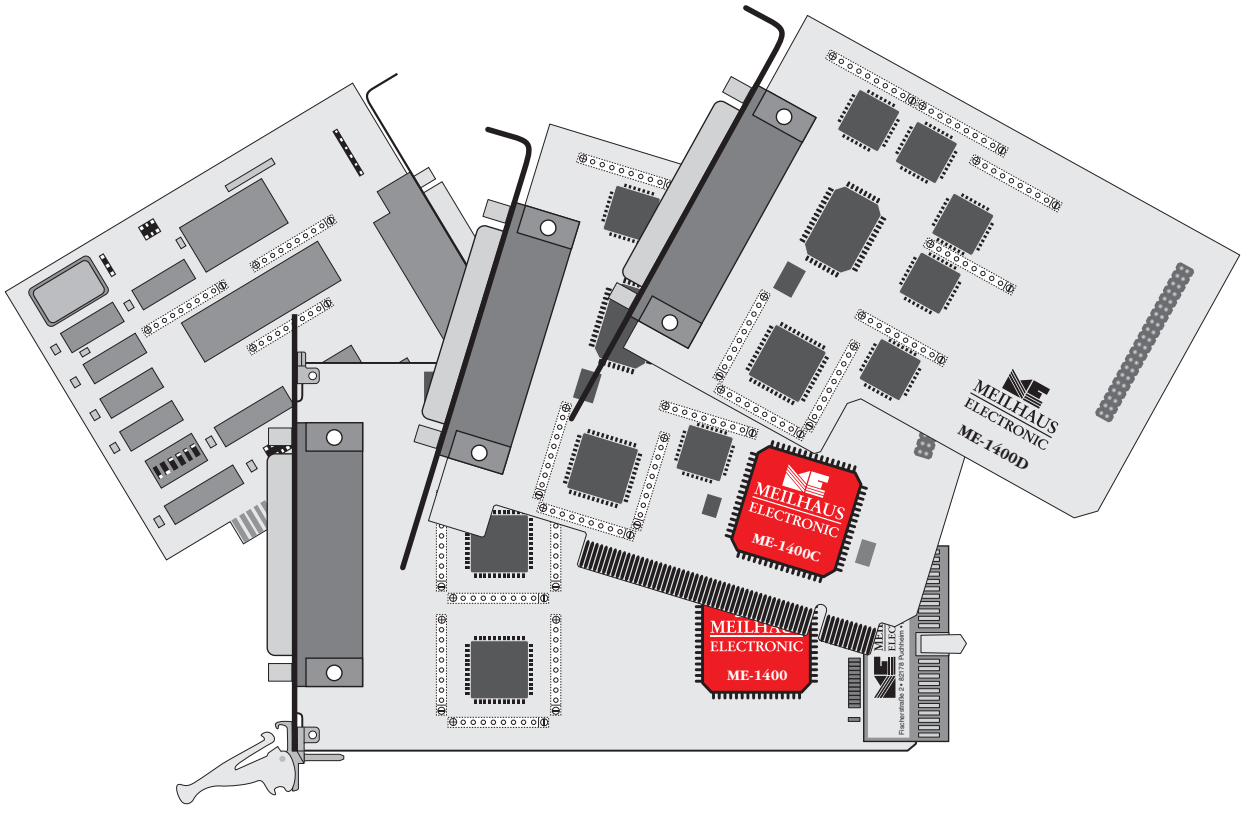
Meilhaus Electronic Manual

ME-14, ME-1400 1.91E

ISA, PCI- and CompactPCI Versions



TTL Digital-I/O- and Counter Boards

Imprint

Manual for the ME-14, ME-1400

Revision 1.91E
Revised: 17. January 2005

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.

ISO 9001
Deutscher Akkreditierungs Rat
TGA-ZQ-002/92-00

PCI INDUSTRIAL COMPUTERS PICMG EUROPE

Table of Contents

1 Introduction

Valued customer,

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

Take the time to carefully examine the contents of the package for any loss or damage that may have occurred during shipping. If there are any items 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.

On the ISA bus versions of the boards pay careful attention to the sections describing how to set the jumpers. This will save having to open the computer case again.

1.1 Package contents

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

- Digital-I/O and Counter board of the ME-14/1400 series
- Manual in PDF format on CD-ROM
  (optional as printed version)
- Driver software on CD-ROM.
- ME-14, ME-1400E/EA/EB: D-sub 37pin male connector
- ME-1400/A/B/C/D: D-sub 78pin male connector
- ME-14B, ME-1400EB: ribbon cable from IDC connector to 37pin D-Sub female connector mounted on additional mounting bracket.

1.2 Features

Model Overview

Model	Connector	TTL-IOs	Counter
ME-14A ISA	37pin D-sub	24	3 x 16 bit
ME-14B ISA	2 x 37pin D-sub	48	6 x 16 bit
ME-1400 PCI/cPCI	78pin D-sub	24	—
ME-1400A PCI/cPCI	78pin D-sub	24	3 x 16 bit
ME-1400B PCI/cPCI	78pin D-sub	48	6 x 16 bit
ME-1400C PCI	78pin D-sub	24	15 x 16 bit
ME-1400D EXP (Expansion board for ME-1400C)	78pin D-sub	24	15 x 16 bit
ME-1400E PCI (connector compatible with ME-14)	37pin D-sub	24	—
ME-1400EA PCI (connector compatible with ME-14A)	37pin D-sub	24	3 x 16 bit
ME-1400EB PCI (connector compatible with ME-14B)	37pin D-sub	48	6 x 16 bit

Table 1: Model overview ME-14/1400 family

The boards of the ME-14/1400 family are programmable digital-I/O and counter boards. Depending on the model, the boards provide 24 or 48 TTL-compatible digital-I/O lines (8255 compatible) and up to 30 independent programmable 16 bit counters (8254 compatible).

All models with counters provide a 10 MHz oscillator which is independent from the system clock of the PC. The frequency can be set to 1 MHz by a jumper on the ISA versions and by software on the PCI and cPCI versions. With the proper configuration, the oscillator frequency is also available on the D-sub connector (not on ME-1400C/D). The boards have an external interrupt line

available (except ME-1400/E) and the ISA versions have a “wait state logic” available.

The external connections to the board are realised with a 37pin D-Sub (ISA and ME-1400E versions) resp. a 78pin D-Sub connector (ME-1400/A/B/C/D). The ME-14B and ME-1400EB versions have an extra IDC connector on the board to enable access to the expanded signals for the second digital-I/O and counter units of the board. A flat ribbon cable and an extra mounting bracket with a 37pin D-Sub female connector are included with the package.

The base address for the ISA models is set by a DIP switch on the board. For the PCI and cPCI versions, the resources are assigned by the BIOS resp. the operating system (Plug&Play).

1.3 System Requirements

The ME-14 ISA boards can be installed into a PC (XT or newer) with a free 8 bit ISA expansion slot. The PCI models require a free standard PCI resp. CompactPCI slot. We recommend a computer system with Pentium® processor or compatibles.

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.

1.5 Software Support

For the newest versions and latest software releases, please consult the README files.

System Drivers	For all common operating systems (see README files)
ME-Software-Developer-Kit (ME-SDK):	
	Support for all common programming languages, demos, tools und test programs
Graphical programming tools	
	Meilhaus VEE Driver System for HP VEE, HP VEE Lab, Agilent VEE Pro and Agilent VEE OneLab
	LabVIEW™ Driver

2 Installation

Please read your computer manual instructions on how to install new hardware components **before installing the board**. Note the chapter „Hardware Installation“ in this manual (if applicable, e. g. for ISA boards).

• Installation under Windows (Plug&Play)

An installation guide describing how to install the driver software can be found in HTML format on CD-ROM. Please read **before installation** and print it on demand!

The following basic procedure should be used:

If you have received the driver software as an archive file please un-pack the software **before installing the board**. First choose a directory on your computer (e. g. C:\Meilhaus).

Then insert the board into your computer and install the driver software. This order of operation is important to guarantee the Plug&Play operation under Windows 95*/98/Me/2000/XP. Windows 95 and NT 4.0 need an analogous order of operation however the installation procedure differs slightly.

**If the Windows version is supported by the appropriate board type (see readme files).*

• Installation under Linux

Note the installation instructions included with archive file of the appropriate driver.

Note: If you want to run the PCI/cPCI versions with application software already written take care of the notes in the proper README file included with the driver software.

If you are using an ISA board please do the jumper settings on the board first (see the following chapters).

2.1 Hardware Installation of ISA Models

☛ **Make sure that the computer is turned off.**

Caution: some of the more sensitive components can be damaged by static electricity!

That´s why: Make sure to ground yourself by touching an exposed metal part of the PC case before handling the board.

☛ Unplug the power cable from your computer.

☛ Open the computer case.

2.1.1 Location of the Jumpers

The ISA versions ME-14A/B require that jumpers be checked/set before the board is installed into the computer. It must also be determined if the settings match the available resources in the computer before installation.

The locations of the jumpers and DIP switches are shown in the diagrams below. The jumper settings are described in the next chapters.

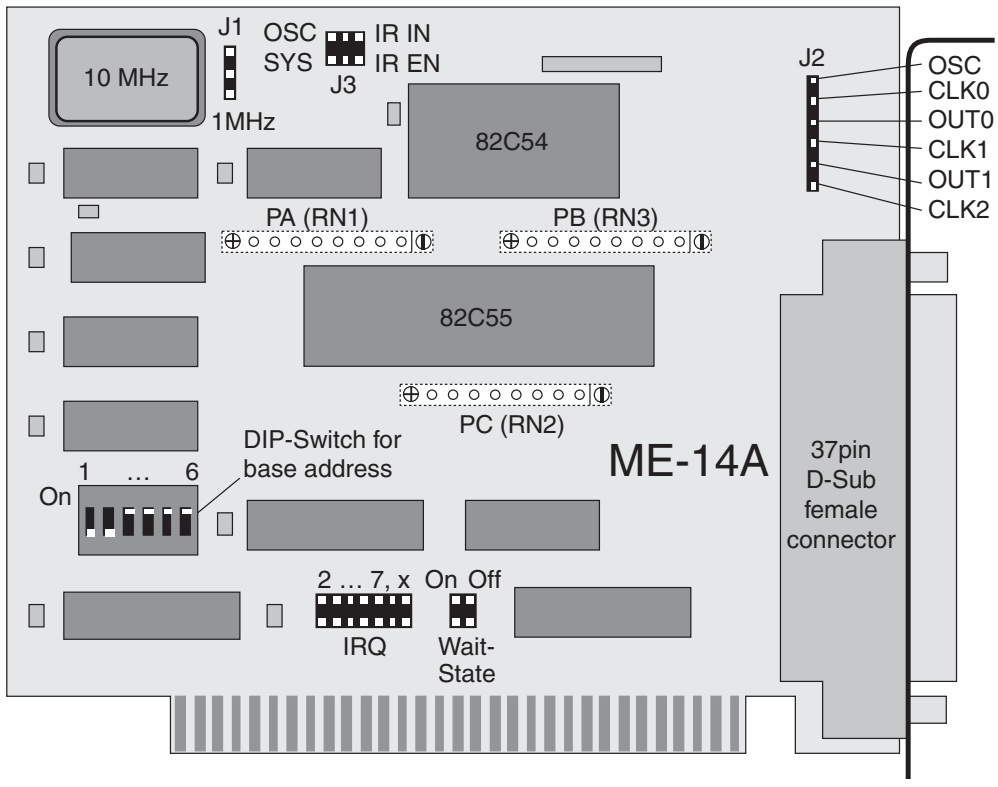


Diagram 1: Picture of the ME-14A ISA

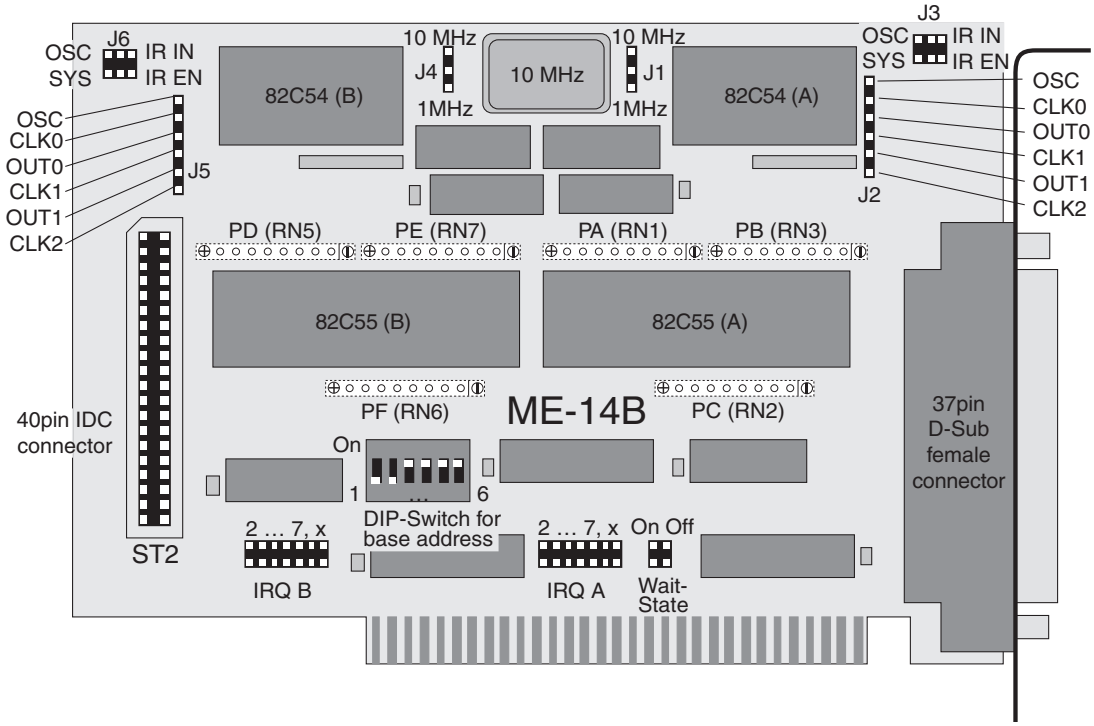


Diagram 2: Picture of the ME-14B ISA

2.1.2 Settings of the DIP-Switches and Jumpers

2.1.2.1 Base Address

The base address (BA) of the ME-14 is set by the DIP switch in 10Hex byte increments. Starting with the base address, the ME-14A occupies 8 bytes and the ME-14B occupies 16 bytes of I/O address space. Make sure that there are no address conflicts with other boards in the system before installing the board into the computer!

A switch that is "on" sets a logic "0" on the address line and a switch that is "off" sets a logic "1" on the address line. By the DIP switches 1…6 every address in the range of 000Hex to 3F0Hex could be set in steps of 10Hex.

Note that for the ISA models, like most ISA boards, only the lower 10 address bits are decoded. This results in the address being mirrored at BA+400Hex, BA+800Hex, BA+C00Hex, BA+1000Hex etc.

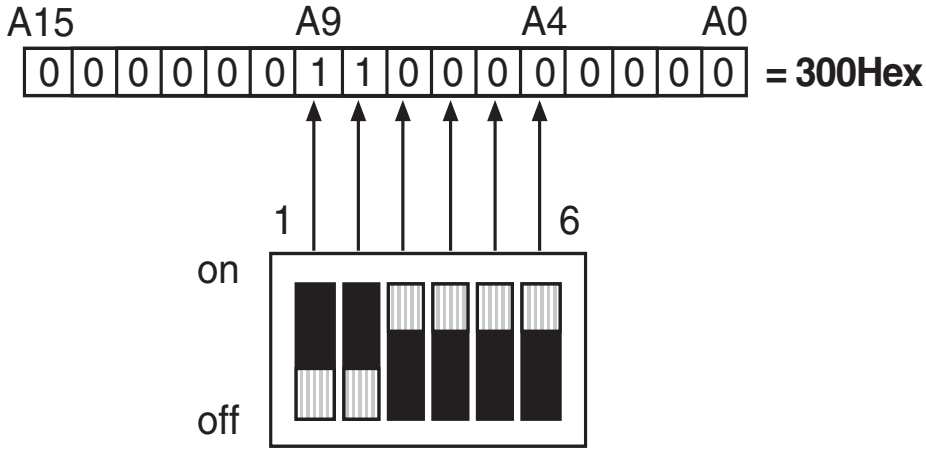


Diagram 3: Setting of base address (Default: 300Hex)

2.1.2.2 Interrupts

For interrupt operation, jumper settings are required. For the ME-14A use jumper „IR“ and for the ME-14B use the jumpers „IRQ A“ and „IRQ B“. Select an interrupt request line (IRQ) between 2…7, resp. „X“ for „none“:

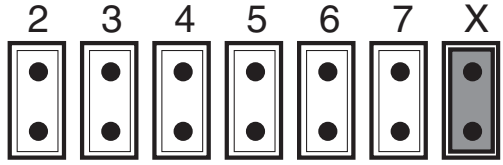
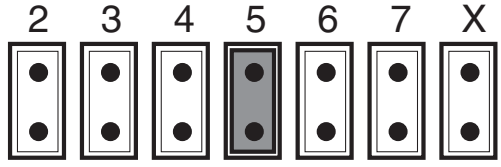
Jumper IR resp. IRQ A and B	Function
	no IRQ
	E. g.: IRQ5 selected

Table 2: Interrupt jumper settings

2.1.2.3 Wait-States

To activate the "wait state" logic of the ME-14, set the jumper labelled "WAIT STATE". This allows operation in systems with a bus speed of over 8 MHz.

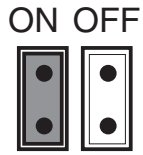
Jumper WAIT STATE	Function
	Wait state logic OFF
	Wait state logic ON

Table 3: Jumper for Wait-States

2.1.2.4 Counter Clock

The clock frequency of the counter device(s) 82C54 can be set to 10 MHz (default) or 1 MHz by the jumper block J1 (and J4 for the ME-14B). This may be required to make the board compatible with older versions of the ME-14.

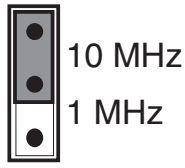
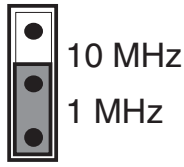
Jumper J1 resp. J4	Function
	10 MHz
	1 MHz (Compatibility mode with older ME-14 versions)

Table 4: Jumper for crystal oszillator clock

2.1.2.5 Cascading the Counters

Using the jumper block J2 (and J5 for the ME-14B) the counters can be cascaded. No external connections are required (see „Block diagram 8254" on page 21).

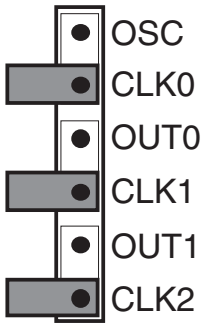
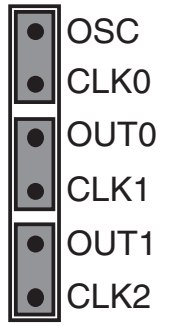
Jumper J2/J5	Function	Jumper J2/J5	Function
	The inputs and outputs of the counters 0…2 are connected to the D-Sub resp. the IDC connector. The counters are independent. (Compatibility mode with older ME-14 versions)		Example: Counter 0…2 are cascaded, counter 0 is driven from oscillator clock*

Table 5: Cascading of Counters

* The jumper settings shown on the right side connect …

- the CLK0 input of counter 0 to oscillator frequency
- the OUT0 output to the CLK1 input of counter 1
- the OUT1 output to the CLK2 input of counter 2.

These connections are made before the D-Sub connector so no external connections must be made.

2.1.2.6 Clock Output and Interrupt Control

The assignment on the D-Sub connector pins 18 and 36 are set on the jumper block J3 (and J6 on the ME-14B). These pins can be used either for clock output or for interrupt control as required. Only the jumper settings shown below are useful (see page 23):

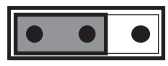
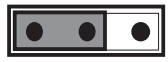
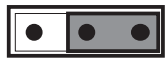
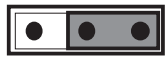
Jumper J3 resp. J6	Function
OSC  IR IN SYS  IR EN	System clock and oszillator clock of the ME-14 are connected to the pins 18 and 36 of the D-sub connector(s). Compatible to older ME-14 versions: Pin 36 = Oszillator Clock 1 MHz (OSC); Pin 18 = System Clock (SYS)
OSC  IR IN SYS  IR EN	Interrupt Enable input and IRQ input of the ME-14 are connected to the pins 18 and 36 of the D-sub connector(s): Pin 36 = Interrupt Input (IR_IN); Pin 18 = Interrupt Enable Input (IR_EN)

Table 6: Jumper clock output and interrupt control

2.1.2.7 Default Settings

Function	Jumper/Switches	Setting
Base address	DIP switch	300Hex
Interrupt	IRQ (IRQ A and B)	X (none)
Clock output/Interupt Control	J3 (additional J6 on the ME-14B)	OSC, SYS
Wait-States	WAIT STATE	OFF
Oscillator frequency	J1 (additional J4 on the ME-14B)	1 MHz
Cascading of counters	J2 (additional J5 on the ME-14B)	not connec-ted

Table 7: Default settings of the ME-14A/B by factory

3 Hardware

3.1 Block Diagram ME-14

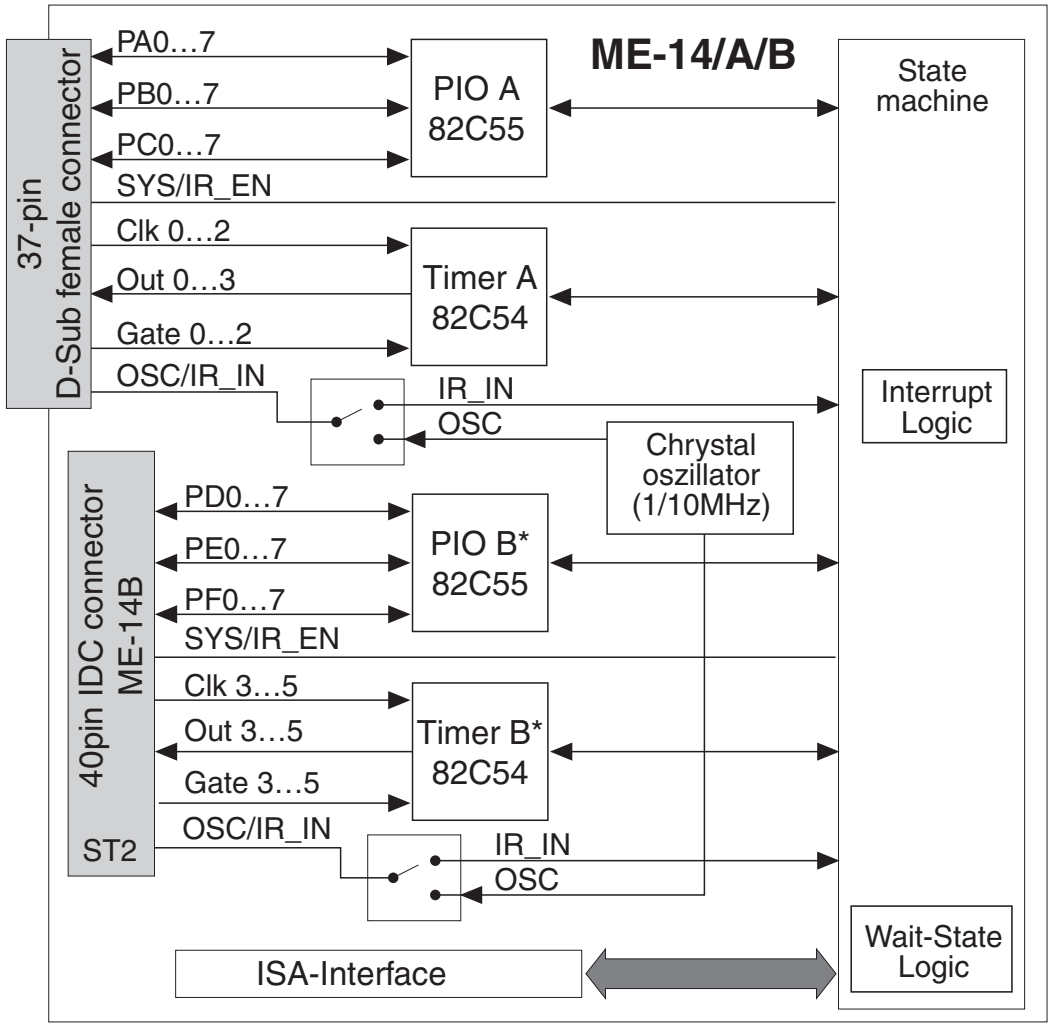


Diagram 4: Block diagram of ME-14

*Depending on the version not all functional groups included in the block diagram above are available:

ME-14A: 24 digital-I/Os (PIO A), 3 x 16 bit counters (Timer A), oszillator and interrupt input.

ME-14B: 48 digital-I/Os (PIO A, B), 6 x 16 bit counters (Timer A, B), oszillator and interrupt input.

3.2 Block Diagram ME-1400/A/B/E/EA/EB

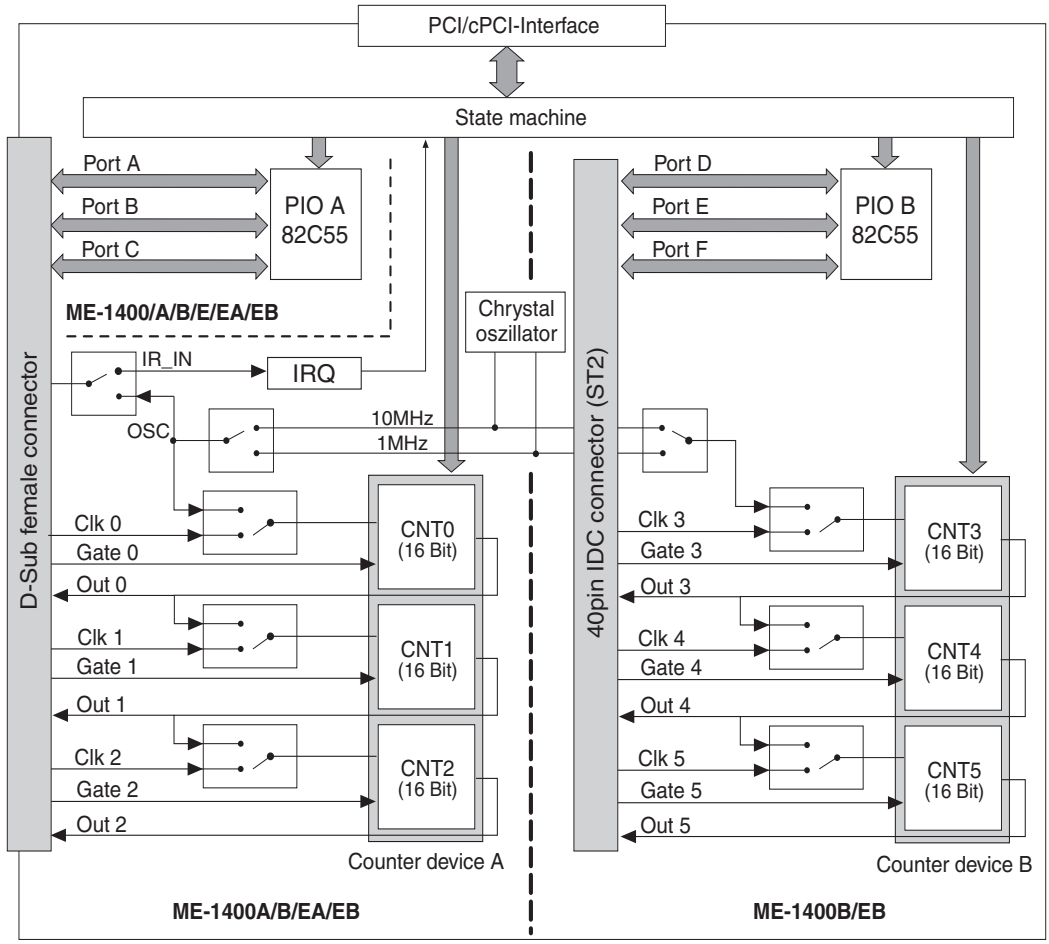


Diagram 5: Block diagram of ME-1400/A/B/E/EA/EB

*Depending on the version not all functional groups included in the block diagram above are available:

ME-1400/E: 24 digital-I/Os (PIO A) without oscillator and interrupt input.

ME-1400A/EA: 24 digital-I/Os (PIO A), 3 x 16 bit counters (CNT0…2), oszillator and interrupt input.

ME-1400B/EB: 48 digital-I/Os (PIO A, B), 6 x 16 bit counters (CNT0…5), oszillator and interrupt input.

3.3 Block Diagram ME-1400C/D

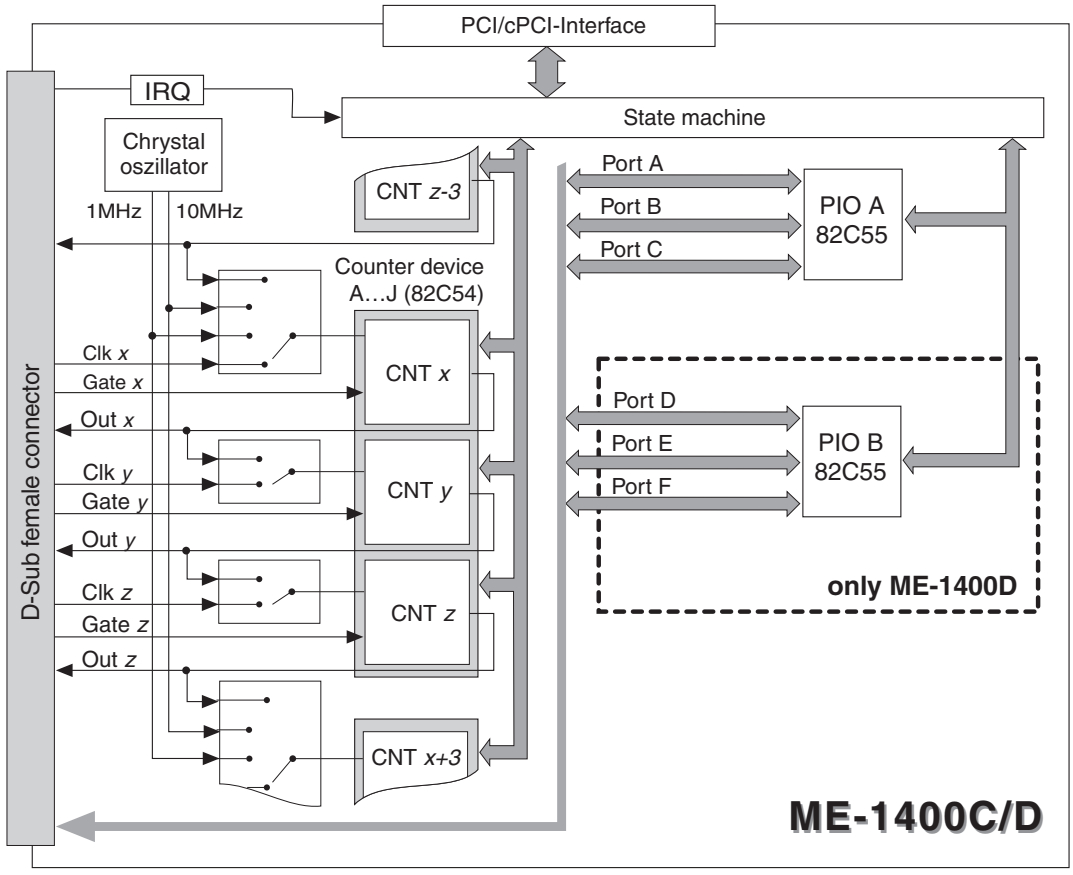


Diagram 6: Block diagram of ME-1400C/D

*Depending on the version not all functional groups included in the block diagram above are available:

ME-1400C: 24 digital-I/Os (PIO A), 15 x 16 bit counters (CNT0…14) and an interrupt input.

ME-1400D: Expansion board with 24 digital-I/Os (PIO B) and 15 x 16 bit counters (CNT0…14).

The counters can be cascaded by software. The first counter of every device can be sourced by the crystal oszillator. For each of the 3 counters (CNT *x, y, z*) per counter device (A…J) use the following indices. See Table 8:

	ME-1400C					ME-1400D				
Counter device ➜	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**
Counter No. CNT x	0	3	6	9	12	15	18	21	24	27
CNT y	1	4	7	10	13	16	19	22	25	28
CNT z	2	5	8	11	14	17	20	23	26	29

Table 8: Counter indices

3.4 General Notes

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

For the pin configuration of the D-Subs see „Pinout" on page 78.

3.5 Digital-I/O (8255)

For the programmable input/output component (PIO) the standard component 82C55 (fully compatible CMOS version of the 8255A) is used. This component has 3 x 8 bit wide programmable I/O-ports and is TTL/CMOS compatible.

For the digital ports of the board the following assignments to the ports (P*x*0…7) of the single PIO devices are valid:

I/Os of 8255 ➜	**PA0…7**	**PB0…7**	**PC0…7**
PIO A	Port A	Port B	Port C
PIO B	Port D	Port E	Port F

Table 9: Port assignment

The PIO device is used in mode 0 „Simple Input/Output". The configuration is done by software from the user. Therefore use the provided driver software; see chapter 5.3.2 „Digital I/O" on page 52ff.

3.6 Counter (8254)

The counter component is the standard type 82C54. This flexible device has 3 independent 16 bit counters. The following block diagram shows the function of the chip:

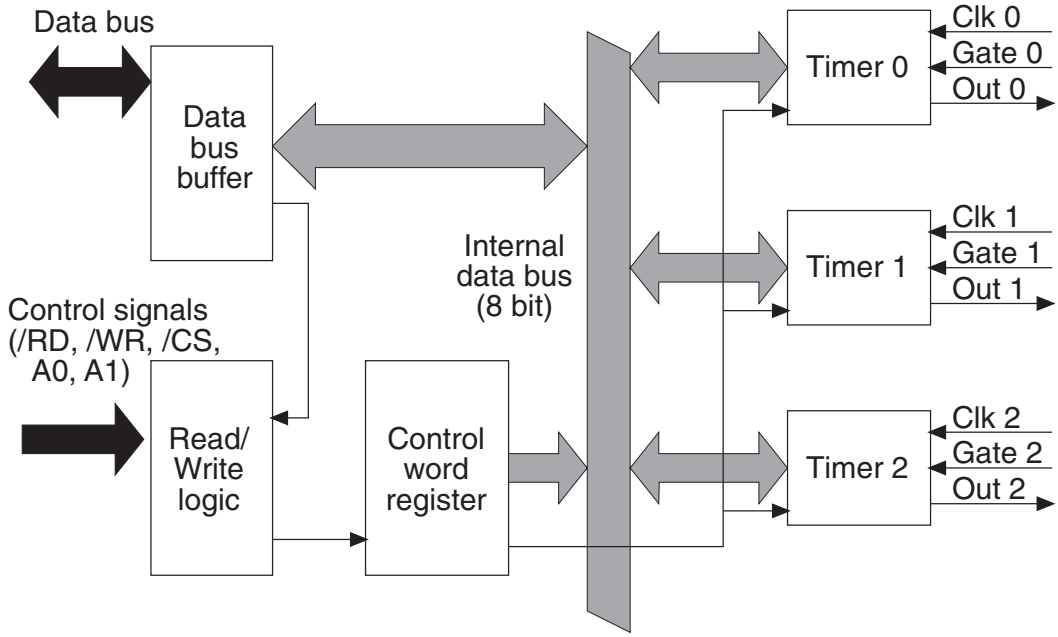


Diagram 7: Block diagram 8254

The configuration of the 82C54 is done by the software. With the function *…CntWrite* every counter must be loaded, configured and started seperately. After proper switching of the gate input (high level) the counter is counting downwards on negative edge of clock. It can be set to count in binary or BCD. Each counter can be configured independently in the following modes (a detailed description of the modes can be found on page 37ff:

- Mode 0: Change state at zero
- Mode 1: Retriggerable „One Shot“
- Mode 2: Asymmetric divider
- Mode 3: Symmetric divider
- Mode 4: Counter start by software trigger
- Mode 5: Counter start by hardware trigger

Make sure that the proper settings for cascading and clock source, etc. are done. Use the jumpers for the ISA versions and the function *me1400CntInitSrc* for the PCI and cPCI versions.

For programming, use the function library provided; see chapter 5.3.3 „Counter Functions“ on page 59 (for ISA models programming on register level is also possible, see page 36).

On the ME-14A/B the Clk-, Gate- and Out-line of the D-Sub connector are directly connected with the corresponding signals of the 82C54. On the PCI and cPCI versions the Gate- and Out-line of the D-Sub are also directly connected with the corresponding signals of the 82C54. In the Clk-lines „multiplexer“ are switched between.

3.6.1 Cascading the Counter

To cascade the counters, the outputs can be connected sequentially without external connections. On the ME-1400C and D the cascading is also possible from device to device (exception: from device E to device F). See block diagram on page 17ff.

This is done by software for the PCI and cPCI versions (after power up or after reset all counters are sourced by the external clock). On the ISA models use the jumper block J2 (additionally jumper block J5 for the ME-14B).

For example, if counters 0…2 are to be cascaded, the following settings must be made using the jumpers or by software:

- Connect the clock input of counter 0 (Clk 0) to the external oscillator clock
- Connect the output of counter 0 (Out 0) to the clock input of counter 1 (Clk 1)
- Connect the output of counter 1 (Out 1) to the clock input of counter 2 (Clk 2)
- To enable the counters, the gate inputs (Gate 0…2) must be switched properly (high level).
- On the output of counter 2 (Out 2), the cascaded counter signal is available.

The output lines of all counters are also available at the D-Sub connector(s).

3.6.2 Counter Clock

The counter clock alternativly can be sourced either by the internal oscillator (1 MHz/10 MHz), externally or by cascading (see also the next chapters and the block diagrams on page 17).

The internal oscillator can be set seperately for each device from 1 MHz (default) to 10 MHz. This setting is done by software for the PCI and cPCI board versions and by jumper J1 (and J4 on the ME-14B) for the ISA boards.

3.6.3 Clock Output and Interrupt Control

The pin labelled „OSC/IR_IN“ resp. „IR_IN“ is the interrupt input by default. Alternativly it can be used for clock output generated by the chrystal oscillator of the board (1 MHz or 10 MHz).

Exception: On the ME-1400C this pin is only for interrupt control, on the ME-1400, ME-1400D and ME-1400E the pin has no function.

OSC: **Oscillator Clock Output** - this signal connects the internal oscillator clock signal (1 MHz or 10 MHz) with the the D-Sub connector.

IR_IN: **IRQ Input** - a rising edge on this pin will cause an interrupt on the selected IRQ line. If IR_IN is held high, not connected, or if the “X” setting is chosen (only ISA models), the IR IN is ignored.

On the PCI/cPCI versions the interrupt logic is disabled after powerup and must be enabled before using by the function *me1400EnableInt*.

The pin labelled „SYS/IR_EN“ is only available on ISA versions.

SYS: **System Clock Output** - this pin outputs the system clock of the ISA bus over a driver to the D-Sub connector.

IR_EN: **Interrupt Enable** - to enable the interrupt logic on the ME-14 ISA, a low level must be placed to this pin. If the pin is not connected, an internal pull-up resistor enables the interrupt logic.

Model	Function	Default	Setting
ME-14A	OSC/IR_IN	OSC	J3 (see page 15)
	SYS/IR_EN	SYS	
ME-14B	OSC/IR_IN	OSC	J3 and J6 (see page 15)
	SYS/IR_EN	SYS	
ME-1400	n.c.	–	–
ME-1400E	n.c.	–	
ME-1400A	OSC/IR_IN	IR_IN	by software (see page 51)
ME-1400EA	OSC/IR_IN	IR_IN	
ME-1400B	OSC/IR_IN	IR_IN	by software (see page 51)
ME-1400EB	OSC/IR_IN	IR_IN	
ME-1400C	IR_IN	IR_IN	input
ME-1400D	not available (please don´t connect)		

Table 10: Overview clock output and interrupt control

3.7 Switching

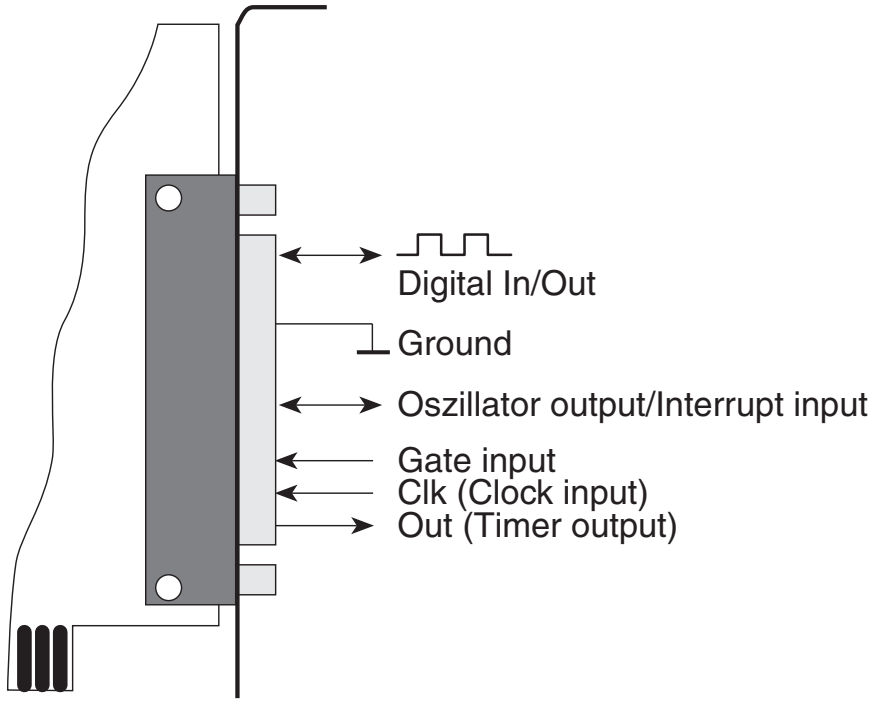


Diagram 8: Switching of ME-14/1400

Note: The signal must be TTL or CMOS compatible and have a reference to PC-GND.

3.7.1 Pull-Up/Pull-Down Resistors

On a power up, all digital ports are set to input. Because of this the corresponding input lines are all set to high impedance (without external switching). Depending on the application, it may be desirable to have the digital lines in a defined state on power up. The ME-14/1400 allows the user to add pull-up or pull-down resistors to the circuit board directly. Appropriate resistor arrays can be used (4.7 kΩ recommended) port by port. Note, that by using pull-up resistors, the output current is decreased accordingly (e. g. with R_{up}=4.7 kΩ, I_{max}=1.6 mA).

Depending on how the resistor arrays are placed on the board, the pull-up or pull-down state is selected. For pull-up, the end pin of the array must go to the "+" pin and for pull-down, the end pin must go to the "-" pin (see diagram 9 to 13).

Note:
Make sure to ground yourself before inserting the arrays to avoid a static discharge..

Port	Array No. ME-14A/B	Array No. ME-1400/A/B /E/EA/EB	Array No. ME-1400C/D
Port A	RN1	RN3	RN1
Port B	RN3	RN2	RN2
Port C	RN2	RN1	RN3
Port D	RN5	RN4	RN1
Port E	RN7	RN5	RN2
Port F	RN6	RN6	RN3

Table 11: Assignment of resistor arrays

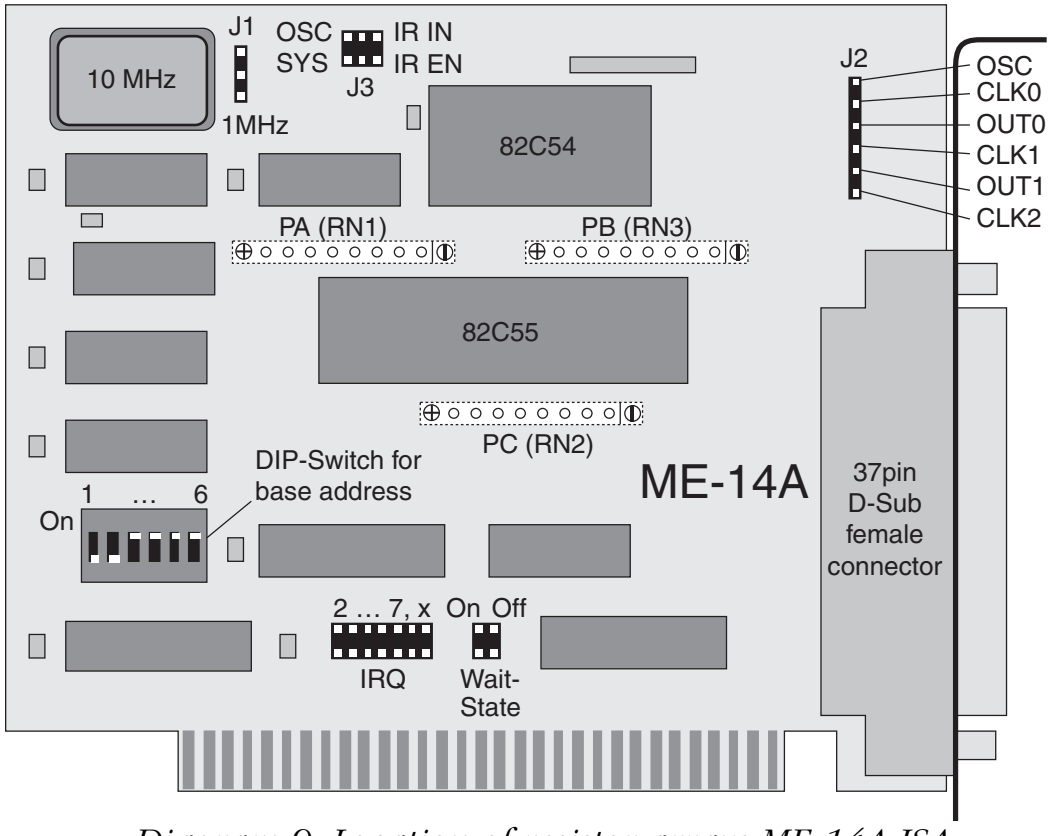


Diagram 9: Location of resistor arrays ME-14A ISA

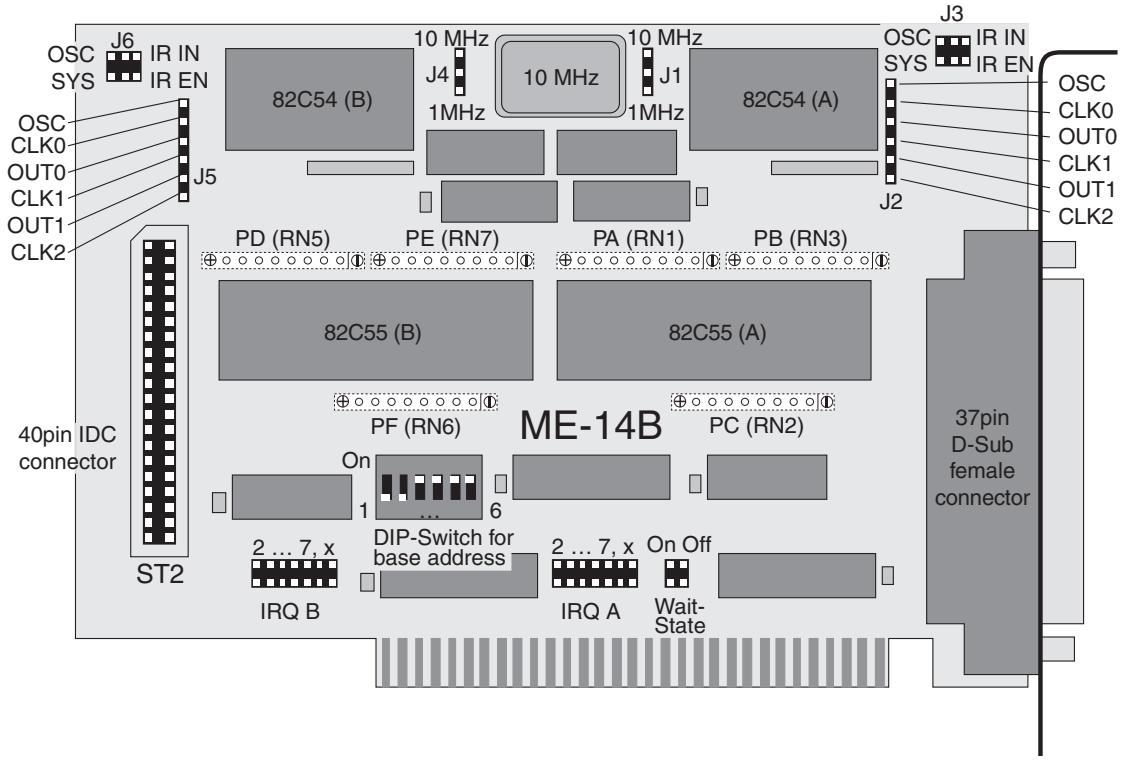


Diagram 10: Location of resistor arrays ME-14B ISA

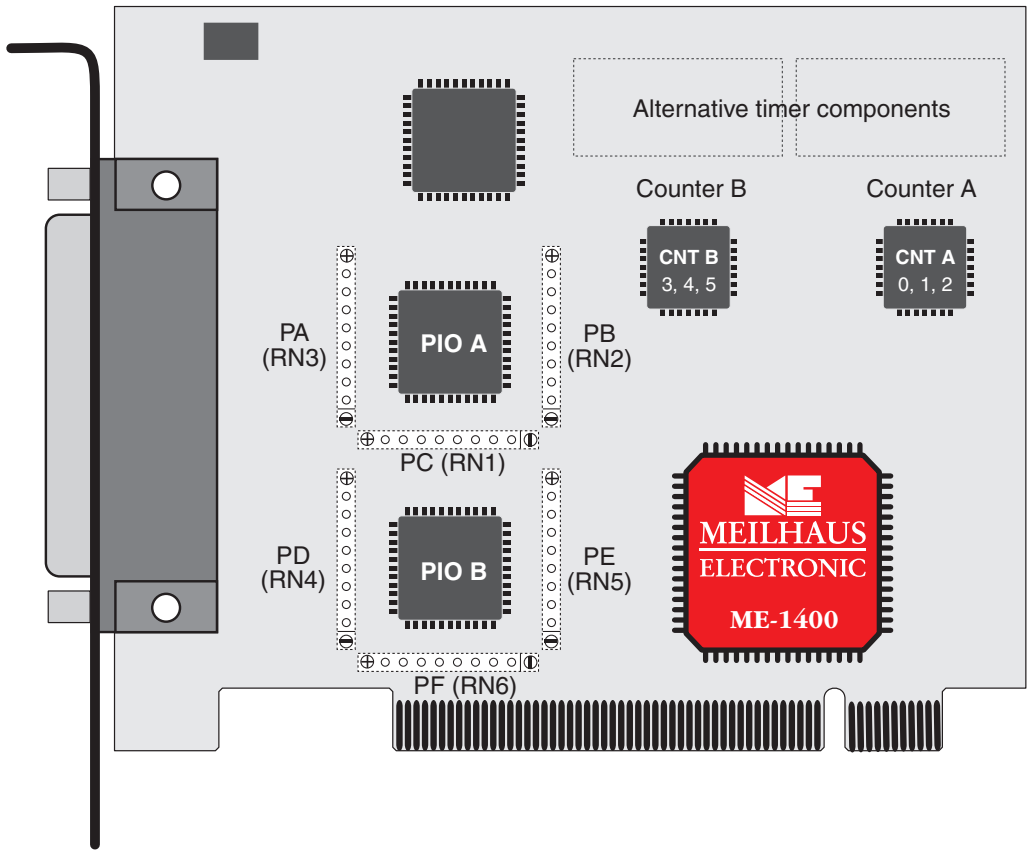


Diagram 11: Location of resistor arrays ME-1400/A/B PCI

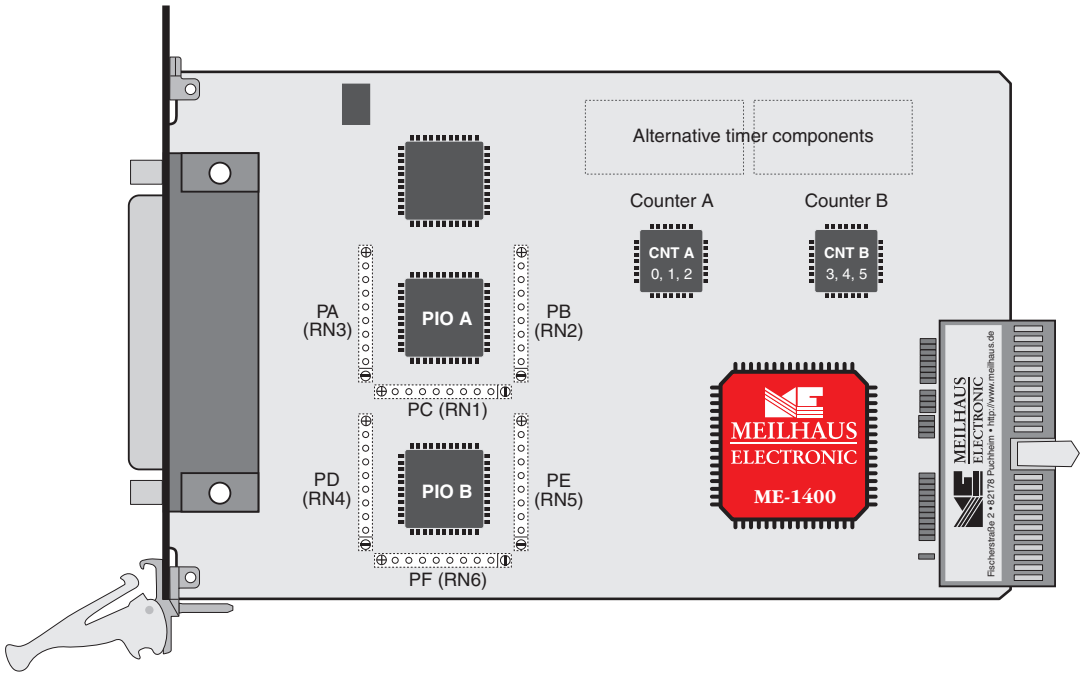


Diagram 12: Location of resistor arrays ME-1400/A/B cPCI

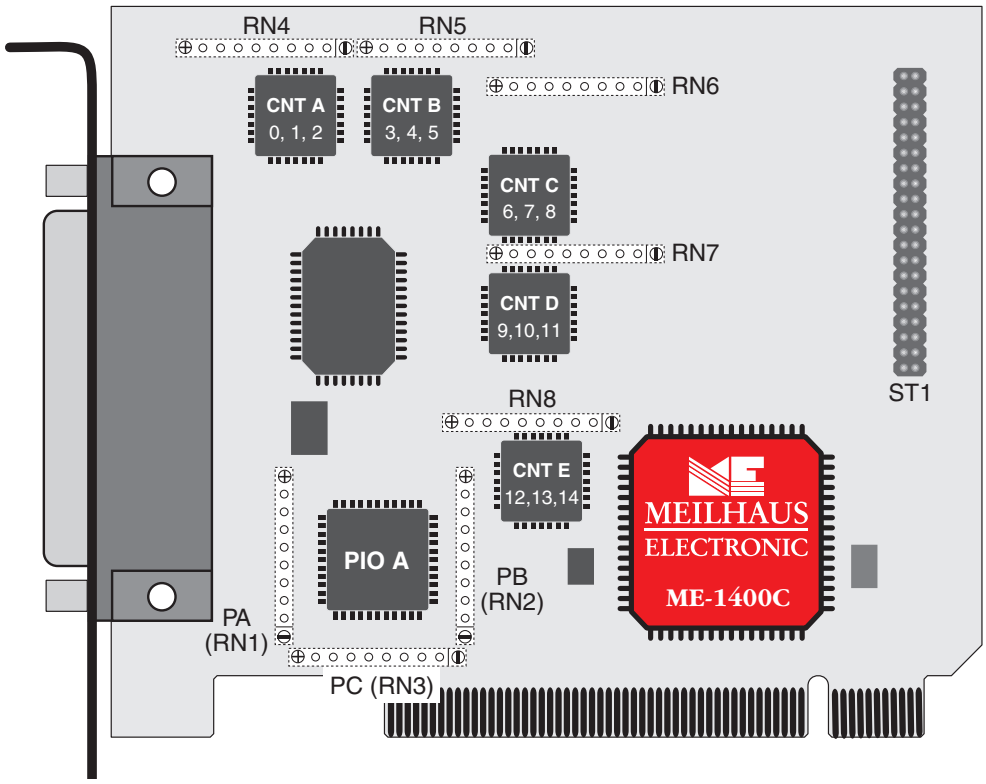


Diagram 13: Location of resistor arrays ME-1400C PCI

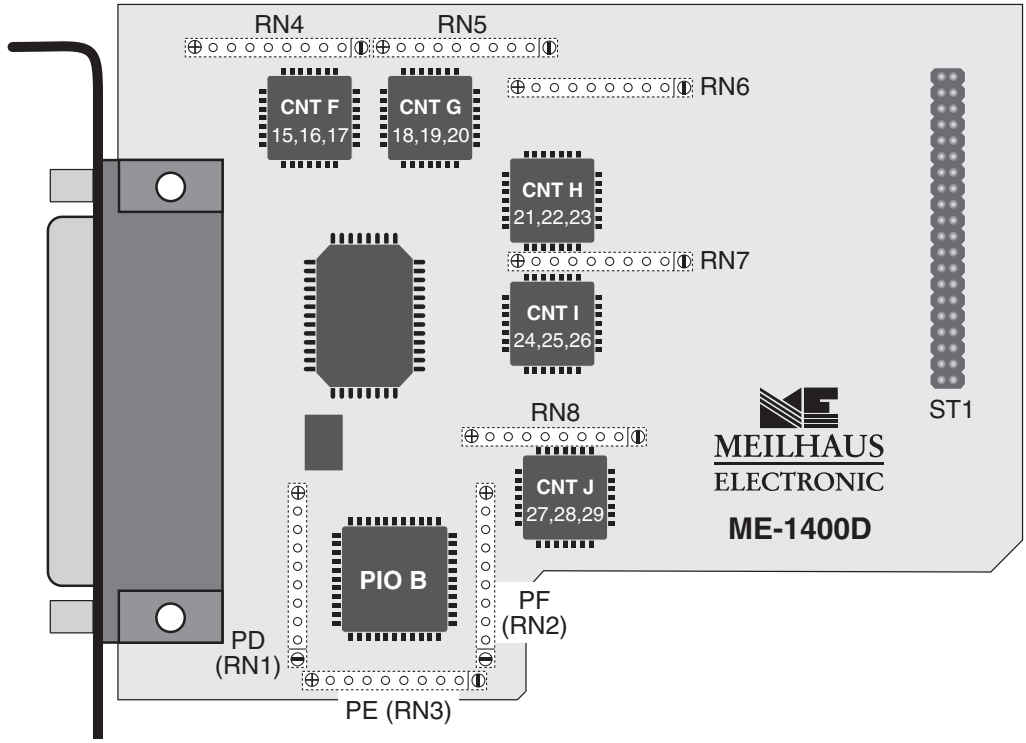


Diagram 14: Location of resistor arrays ME-1400D EXP

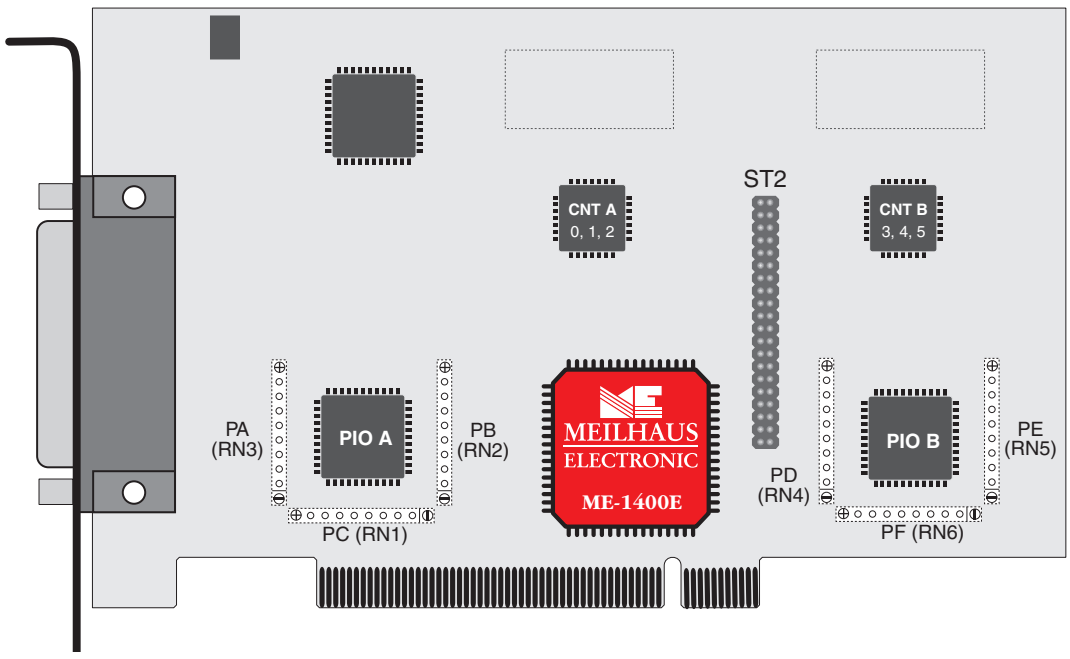


Diagram 15: Location of resistor arrays ME-1400E/EA/EB

3.8 Test Program

For test issues an test program is provided. The appropriate test program could 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).

4 Programming

The driver concept of the ME-14 resp. the ME-1400 family offers you the possibility without changing your software using an equivalent PCI board instead of an „old“ ISA board of the ME-14 family. This may be possible without recompilation because the functionallity and syntax are identical. (Assumption: your „old“ ISA board and the new PCI board are using the same value in parameter <iBoardNumber>). Please note the appropriate READ-ME files on the installation disk of the ME-1400 driver system for the exact order of operation.

It only works, if you used the function library of the ME-14 Driver System for programming your ISA board!

4.1 High Level Language Programming

The following high level languages are supported by standard:

- Visual C++ (version 4.0 or higher).
- Delphi (version 2.0 or higher).
- Visual Basic (version 4.0 or later).
- For further infos see the appropriate README files on the ME-Power-CD.

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 include them into your project. These demo programs can be found within the ME Software Developer Kit (ME-SDK), which is installed to directory `C:Meilhaus\me-sdk by` default. Please read the notes in the appropriate README files.

4.2 Agilent VEE Programming

The Agilent VEE components for your board are included with the „ME-Power-CD“ or are available for download under www.meilhaus.com.

The Meilhaus VEE Driver System supports the HP VEE full versions 4.x and 5.x, HP VEE Lab, Agilent VEE Pro and Agilent VEE OneLab. For installation of VEE components and for further information please read the documentation included with the VEE driver system. For basics of VEE programming please use your VEE documentation and the 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 VEE development environment. They can be placed and „wired“ in your application the same as standard 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 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 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 VEE menu by the „ME Board“ entry. It enables a convenient use of all driver functions available in VEE. From 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 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 VEE components any time (see „ME-Power-CD“).

4.3 LabVIEW™ Programming

The LabVIEW components for your board are included with the „ME-Power-CD“ or are available for download under www.meilhaus.com.

For installation of LabVIEW driver components please read the documentation includeed with the appropriate LabVIEW driver. For basics of LabVIEW programming please use your LabVIEW documentation and the LabVIEW online help.

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“. 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 „…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 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 Pulse Width Modulation

With proper external connections the three counters of each counter device can be used to output a signal with a variable duty cycle. The duty cycle can be set between 1…99% in 1% increments.

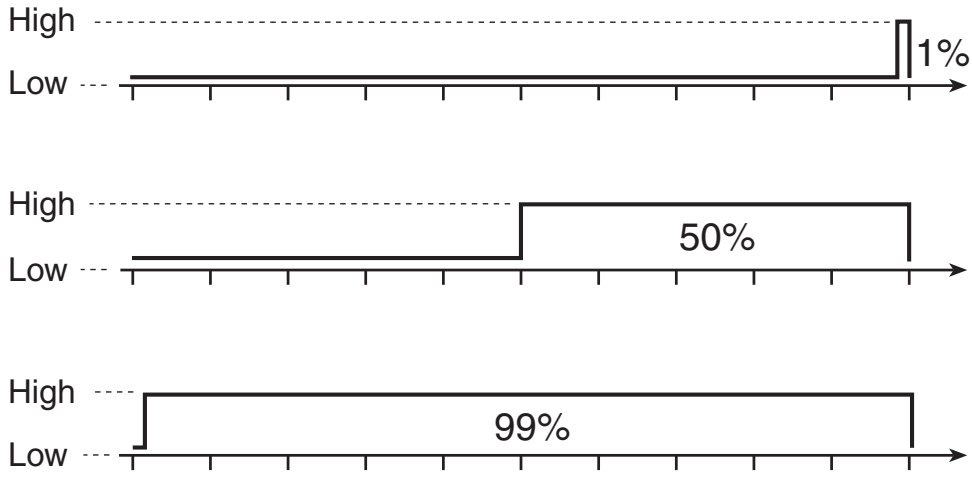


Diagram 16: Duty cycle

The base clock can be provided either by an external frequency generator (max.10MHz) or by the internal crystal oscillator (1MHz or 10MHz); see parameter `<ClockSource>`. Using the parameter `<Prescaler>` you can vary the frequency between base-clock/2 and base-clock/65535. Using the parameter `<DutyCycle>` you can set the duty cycle between 1…99% in steps of 1%. The output signal always is provided at the output of counter 2 of the corresponding counter device (OUT_2, OUT_5,…). The frequency of the output signal can be 50kHz maximum.

By using the connections shown in the diagram 17, the functions *me1400CntPWMStart/Stop* can be used which greatly simplify the programming (see page 61).

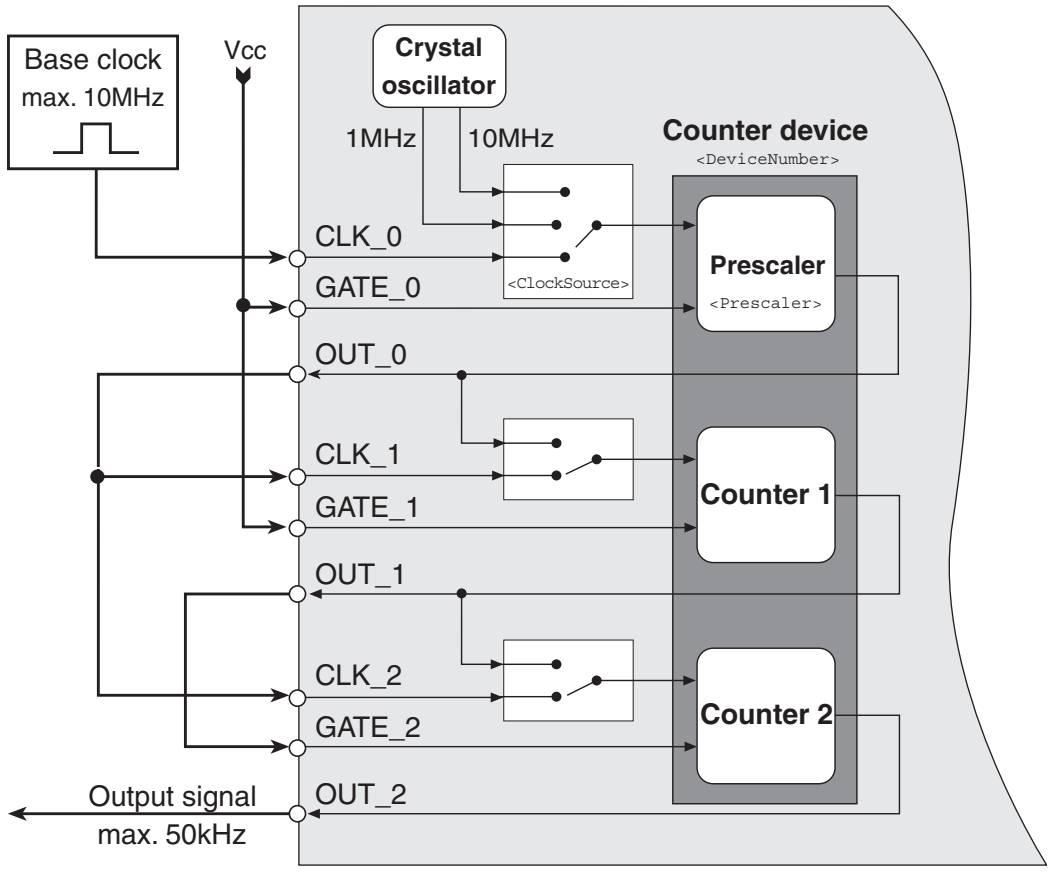


Diagram 17: Switching pulse width modulation

4.5 Register Programming

The ME-14 ISA models can be programmed at the register level (e. g. in DOS). Consult the manuals for the high level language of your choice for the proper commands and syntax. We recommend the use of the driver software shipped with the board. The registers are briefly described in this section:

4.5.1 Register Description

The boards require 16 consecutive bytes of I/O address space starting at the base address „BA“. The base address is set by the DIP switch on the board. The applicable board versions are given in brackets in the “Function” column in the following table.

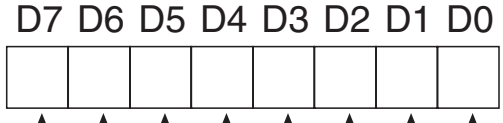
Addr.	Function	Addr.	Function
BA+0H	Port A of 8255 (all)	**BA+8H**	Port A of 8255 (B)
BA+1H	Port B of 8255 (all)	**BA+9H**	Port B of 8255 (B)
BA+2H	Port C of 8255 (all)	**BA+AH**	Port C of 8255 (B)
BA+3H	Control word for 8255 (all)	**BA+BH**	Control word for 8255 (B)
BA+4H	Data counter 0 (A+B)	**BA+CH**	Data counter 0 (B)
BA+5H	Data counter 1 (A+B)	**BA+DH**	Data counter 1 (B)
BA+6H	Data counter 2 (A+B)	**BA+EH**	Data counter 2 (B)
BA+7H	Control word for 8254 (A+B)	**BA+FH**	Control word for 8254 (B)

Table 12: Address space of the ME-14

4.5.1.1 Registers of 82C55

The registers of the 82C55 can only be accessed on the ISA models under DOS. For all other cases we recommend the use of the driver software shipped with the board, which guarantees full functionallity.

There are 4 user accessible registers on the 82C55 chip. Data is exchanged using 3 registers, 8 bits wide (BA+0H … BA+2H, and for the "B" board versions: BA+8H … BA+AH). The mode of operation is set by an 8 bit control register (BA+3H resp. BA+BH) 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 18: Control word of the 82C55

Basically there are 3 modes of operation; however only mode 0 is relevant for the ME-14/1400. For more detailed information about the 82C55 consult the manufacturers data sheets.

4.5.1.1.1 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-14/1400 (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 13: Control word of the 8255

Control word D7…D0	Hex	Port A	Port B	Port C
10000000	$80	Output	Output	Output
10001001	$89	Output	Output	Input
10000010	$82	Output	Input	Output
10001011	$8B	Output	Input	Input
10010000	$90	Input	Output	Output
10011001	$99	Input	Output	Input
10010010	$92	Input	Input	Output
10011011	$9B	Input	Input	Input

Table 14: Input /Output configuration of the 82C55 in mode 0

4.5.1.2 Registers of 82C54

The registers of the 82C54 can only be accessed on the ISA models under DOS. For all other cases we recommend the use of the driver software shipped with the board which guarantees full functionallity.

The control word (BA+7H resp. BA+FH) 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 15: Control word of the 82C54

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 16: 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 17: 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 18: 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 19: Selection of the counter mode

First, the counter is initialised by writing the control word to the appropriate register (BA+7H resp. BA+FH). 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+4H…BA+6H, and for the "B" versions: BA+CH… BA+EH). The counter is decremented by 1 on every falling edge of Clk-signal. The 6 available modes of operation are briefly described next.

4.5.1.2.1 Mode 0: Change state at zero

This mode of operation can be used to trigger an interrupt when the count reaches zero. The counter output (Out 0…2) is set to low when the counter is initialised or when a new start value is loaded. To enable the counting, the Gate input must be high. As soon as the start value is loaded and the counter is enabled, the counting begins (downwards) and the output remains low.

Upon zero axis crossing, the output goes high and stays high until the counter is reloaded or initialised again. The counter continues to count down, even after zero is reached. If a counter register is loaded during a count in progress the following occurs:

1. when the first byte is written, the count process is stopped
2. when the second byte is written, the count process begins again

4.5.1.2.2 Mode 1: Retriggerable „One-Shot“

The counter output (Out 0…2) is set high when the counter is initialised. When a start value is loaded the output will go low on the next trigger pulse (positive edge on the Gate signal). When the zero count is reached, the counter output goes high again.

With a positive edge on the Gate input, the counter can be set back (re-triggered) to the start value. The output will stay low until the counter reaches zero.

The counter value can be read at any time without effecting the counting process.

4.5.1.2.3 Mode 2: Asymmetric divider

In this mode, the counter functions as a frequency divider. The counter output (Out 0…2) is set to 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 counter is not affected at the moment. The new value is used on the following period.

4.5.1.2.4 Mode 3: Symmetric divider

This mode of operation is similar to mode 2 with the difference that the divided frequency is symmetric (only for even count values). The counter output (Out 0…2) goes high after initialisation. When the counter is enabled with a high level on the Gate signal, the counter counts down by 2. The output will cycle (change state), starting with high, every start value/2-periods of the input signal. As long as the Gate input remains high, the process is repeated, otherwise the output immediately goes to the high state.

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

4.5.1.2.5 Mode 4: Counter start by software trigger

The counter output (Out 0…2) is set to high when the counter is initialised. To enable the counter the Gate signal must be high. When the counter is loaded (software trigger) and enabled, it begins counting downwards, while the output remains high.

When zero is reached the output is set to low for one clock period and then goes high again. The output remains high until the counter is initialised again and a new start value is loaded.

If the counter is reloaded during a count process, the new start value is used in the next cycle.

4.5.1.2.6 Mode 5: Counter start by hardware trigger

The counter output (Out 0…2) is set to high when the counter is initialised. When a start value is loaded, the count process will start on the cycle which follows the first trigger pulse (positive edge on the Gate input). When zero is reached, the output goes low for one clock period. Then the output goes high and remains high until the next trigger pulse occurs.

If the count register is reloaded between trigger pulses, the new start value is used after the next trigger pulse.

The counter can be reset to the start value (retriggered) at any time by applying a positive edge to the Gate input. The output will remain high until the zero count is reached.

5 Function Reference

5.1 General

The driver concept for the ME-14/1400 family intends two independent drivers for ISA and for PCI/cPCI boards. Because of the driver concept ISA boards can be basically only accessed by the ME-14 driver and PCI/cPCI boards only by ME-1400 driver. To get a compatibility of PCI boards and even written application software, additional all functions of the ISA driver (*_me14...*) are declared in the PCI driver accessing to similar built functions of the the PCI driver (*me1400...*). If you want to use this feature, please read the notes for the exact order of operation in the appropiate README files.

The funcitons of the API-DLL (`ME14_32.DLL`) for the ME-14 are supported by the following 32 bit drivers:

- VxD driver (`ME14_32.VXD`) for Windows 95/98/Me
- Kernel driver (`ME14_32.SYS`) for Windows NT4.0/2000/XP

  further on required: dialogue DLL (`MEDLG32.DLL`).

The funcitons of the API-DLL (`ME1400.DLL`) for the ME-1400 are supported by the following 32 bit drivers:

- VxD driver (`ME1400.VXD`) for Windows 95
- Kernel driver (`ME1400.SYS`) for Windows NT4.0
- WDM driver (`ME1400.SYS`) for Windows 98/Me/2000/XP

After the driver is successfully loaded, the API functions allow convenient access to the hardware. Every function that accesses a ME-14/1400 board requires an integer value for identification of the board. In the following description of the functions this parameter is referred to as `<BoardNumber>`. It specifies board to be accessed.

5.2 Naming Conventions

These functions were written board specific. The prefix means the board(s) for which the function is valid. 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. "DI" for "Digital Input").

_me14… Function valid for ME-14 ISA models

me1400… Function valid for ME-1400 PCI/cPCI models

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

function name	will be italic in body text e. g. *me1400GetBoardVersion*
`<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]	will indicate optional variables
`FILE NAMES`	or `PATHS` will be capitalized in font `Courier`
`me1400...()`	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:

Functions	Short Description	Page
General Functions		
_me14GetBoardVersion me1400GetBoardVersion	Determine board version	49
_me14GetDLLVersion me1400GetDLLVersion	Determine DLL version number	50
_me14GetDriverVersion me1400GetDriverVersion	Determine driver version number	50
me1400SetMultifunctionPin	Configuration of pin 39 (OSC/IR IN) as an IRQ input or clock output	51
Digital I/O		
_me14DIOSetPortDirection me1400DIOSetPortDirection	Configures a port as an input or output	52
_me14DIGetBit me1400DIGetBit	Reading a bit	53
_me14DIGetByte me1400DIGetByte	Reading a byte	55
_me14DOSetBit me1400DOSetBit	Writing a bit	56
_me14DOSetByte me1400DOSetByte	Writing a byte	57

Table 20: Overview of library functions

Functions	Short Description	Page
Counter Functions		
me1400CntInitSrc	Configuring the counters 0…29	64
me1400CntPWMStart	Start PWM operation	61
me1400CntPWMStop	Stop PWM output	63
_me14CntRead me1400CntRead	Reading the current counter value	64
_me14CntWrite me1400CntWrite	Configuring counter and loading the start value	65
me1400InitModeTimerA	(do not use for new projects)	66
me1400InitModeTimerB	(do not use for new projects)	68
Interrupt Handling		
_me14DisableInt me1400DisableInt	Disable interrupt control	70
_me14EnableInt me1400EnableInt	Enable interrupt control	71
me1400GetIrqCnt	Acquires the number of interrupts	72
Error Handling		
_me14GetDrvErrMess me1400GetDrvErrMess	Error string corresponding to error code	73

Table 20: Overview of library functions

Note: In case a function is valid for ISA boards (prefix *_me14*) and for PCI boards (prefix *me1400*) the common prefix *me14xx* will be used in the following function description:

5.3.1 General Functions

_me14GetBoardVersion
me1400GetBoardVersion

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

Determines the board version number of an installed board of the board family ME-14/1400.

● Definitions

C: int me14xxGetBoardVersion (int iBoardNumber, int *piDevices;)

Delphi: Function me14xxGetBoardVersion (iBoardNumber: integer; Var iDevices: integer): integer;

Basic: Declare Function me14xxGetBoardVersion Lib "me14xx_32" Alias "_VBme14xxGetBoardVersion@8" (ByVal iBoardNumber As Long, iDevices As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<Version>

Pointer to an integer value where the board version is returned. Possible values:

014AHex:	ME-14A
014BHex:	ME-14B
1400Hex:	ME-1400 or ME-1400E
140AHex:	ME-1400A or ME-1400EA
140BHex:	ME-1400B or ME-1400EB
140CHex:	ME-1400C
140DHex:	ME-1400D

‹ 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 *me14xxGetDrvErrMess*.

_me14GetDLLVersion
me1400GetDLLVersion

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

Determines the version number of the DLL for the boards of the ME-14/1400 board family.

● Definitions

C: int me14xxGetDLLVersion();

Delphi: Function me14xxGetDLLVersion: integer;

Basic: Declare Function me14xxGetDLLVersion Lib "me14xx_32" Alias "_VBme14xxGetDLLVersion@0" () As Long

➔ Parameter 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.: 0x00020001 indicates the version number 2.01

_me14GetDriverVersion
me1400GetDriverVersion

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

Determines the version number of the ME-14/1400 driver.

● Definitions

C: int me14xxGetDriverVersion(int *piBuffer);

Delphi: Function me14xxGetDriverVersion (Var piBuffer: integer): integer;

Basic: Declare Function me14xxGetDriverVersion Lib „me14xx“ Alias "_VBme14xxGetDriverVersion@4" (ByRef lBuffer As Long) As Long

➔ Parameters

<Buffer>

Pointer to a integer value containing the driver version.

❮ 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 *me14xxGetDrvErrMess*

me1400SetMultifunctionPin

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	✔ (not ME-1400/E)	–

Configures Pin 39 (OSC/IR IN) as either interrupt input or clock output.

● Definitions

C: int me1400SetMultifunctionPin (int iBoardNumber, int iMultiPin)

Delphi: Function me1400SetMultifunctionPin (iBoardNumber: integer; MultiPin: integer): integer;

Basic: Declare Function me1400SetMultifunctionPin Lib "me1400" Alias "_VBme1400SetMultifunctionPin@8" (ByVal iBoardNumber As Long, ByVal MultiPin As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<MultiPin>

Configure the multi-function pin (OSC/IR_IN):

- ME1400_MULTIPIN_IRQ (00Hex)
  Pin 39 is used as interrupt input (IR IN) (Default)
- ME1400_MULTIPIN_INTERNALCLOCK (01Hex)
  Pin 39 is used as output for the internal oscillator clock (OSC)

‹ 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 *me14xxGetDrvErrMess*.

5.3.2 Digital I/O

_me14DIOSetPortDirection
me1400DIOSetPortDirection

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

Configures the digital ports as input or output.

☞ Important Note:

Before any bit/byte reading or writing can be done, this function must be called at least once for every port.

● Definitions

C: int me14xxDIOSetPortDirection (int iBoardNumber, int iPortNo, int iDir);

Delphi: Function me14xxDIOSetPortDirection (iBoardNumber, iPortNo, iDir: integer): integer;

Basic: Declare Function me14xxDIOSetPortDirection Lib "me14xx_32" Alias "_VBme14xxDIOSetPortDirection@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iDir As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<PortNo>

Port name; possible values are:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (only ME-14A/B)
- PORTCH (04Hex) Port C High (only ME-14A/B)
- PORTD (08Hex) Port D (only B/D versions)
- PORTE (09Hex) Port E (only B/D versions)
- PORTF (0AHex) Port F (only B/D versions)
- PORTFL (0BHex) Port F Low (only ME-14B)
- PORTFH (0CHex) Port F High (only ME-14B)

<Dir>

Direction of port; possible values are:

- 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 *me14xxGetDrvErrMess*.

_me14DIGetBit
me1400DIGetBit

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

Returns the status of a single input line.

☞ Important Note:

Before any bit/byte reading or writing can be done, the function *…DIOSetPortDirection* must be called at least once for every port.

● Definitions

C: int me14xxDIGetBit (int iBoardNumber, int iPortNo, int iBitNo, int *piBitValue);

Delphi: Function me14xxDIGetBit (iBoardNumber, iPortNo, iBitNo: integer; Var iBitValue: integer): integer;

Basic: Declare Function me14xxDIGetBit Lib "me14xx_32" Alias "_VBme14xxDIGetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByRef iBitValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<PortNo>

Port name; possible values are:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (only ME-14A/B)
- PORTCH (04Hex) Port C High (only ME-14A/B)
- PORTD (08Hex) Port D (only B/D versions)
- PORTE (09Hex) Port E (only B/D versions)
- PORTF (0AHex) Port F (only B/D versions)
- PORTFL (0BHex) Port F Low (only ME-14B)
- PORTFH (0CHex) Port F High (only ME-14B)

<BitNo>

The number the input line whose status is to be read in. Possible values are: 0…7 representing bits 0…7 of an port

<BitValue>

Pointer to an integer value representing the status of the selected input line:

0: input line is low

1: input line is high

❮ 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 *me14xxGetDrvErrMess*.

_me14DIGetByte
me1400DIGetByte

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

This function reads a byte from specified input port.

☞ Important Note:

Before any bit/byte reading or writing can be done, the function *…DIOSetPortDirection* must be called at least once for every port.

● Definitions

C: int me14xxDIGetByte (int iBoardNumber, int iPortNo, int *piValue);

Delphi: Function me14xxDIGetByte (iBoardNumber, iPortNo: integer; Var iValue: integer): integer;

Basic: Declare Function me14xxDIGetByte Lib "me14xx_32" Alias "_VBme14xxDIGetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, iValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (only ME-14A/B)
- PORTCH (04Hex) Port C High (only ME-14A/B)
- PORTD (08Hex) Port D (only B/D versions)
- PORTE (09Hex) Port E (only B/D versions)
- PORTF (0AHex) Port F (only B/D versions)
- PORTFL (0BHex) Port F Low (only ME-14B)
- PORTFH (0CHex) Port F High (only ME-14B)

<Value>

Pointer to an integer value representing the read in 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 *me14xxGetDrvErrMess*.

_me14DOSetBit
me1400DOSetBit

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

This function sets an output line to the chosen condition.

☞ Important Note:

Before any bit/byte reading or writing can be done, the function *…DIOSetPortDirection* must be called at least once for every port.

● Definitions

C: int me14xxDOSetBit (int iBoardNumber, int iPortNo, int iBitNo, int iBitValue);

Delphi: function me14xxDOSetBit (iBoardNumber, iPortNo, iBitNo, iBitValue: integer): integer;

Basic: Declare Function me14xxDOSetBit Lib "me14xx_32" Alias "_VBme14xxDOSetBit@16" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iBitNo As Long, ByVal iBitValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<PortNo>

Port name; possible values are:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (only ME-14A/B)
- PORTCH (04Hex) Port C High (only ME-14A/B)

- PORTD (08Hex) Port D (only B/D versions)
- PORTE (09Hex) Port E (only B/D versions)
- PORTF (0AHex) Port F (only B/D versions)
- PORTFL (0BHex) Port F Low (only ME-14B)
- PORTFH (0CHex) Port F High (only ME-14B)

<BitNo>

Number of the output line whose status is to be set; possible values are: 0…7 representing bits 0…7 of a port

<BitValue>

Possible values are:

0: output line is set to logical „0“

1: output line is set to logical „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 *me14xxGetDrvErrMess*.

_me14DOSetByte
me1400DOSetByte

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

This function writes a byte to the specified output port.

☞ Important Note:

Before any bit/byte reading or writing can be done, the function *…DIOSetPortDirection* must be called at least once for every port.

● Definitions

C: int me14xxDOSetByte (int iBoardNumber, int iPortNo, int iValue);

Delphi: function me14xxDOSetByte (iBoardNumber, iPortNo, iValue: integer): integer;

Basic: Declare Function me14xxDOSetByte Lib "me14xx_32" Alias "_VBme14xxDOSetByte@12" (ByVal iBoardNumber As Long, ByVal iPortNo As Long, ByVal iValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<PortNo>

Port name; possible values are:

- PORTA (00Hex) Port A
- PORTB (01Hex) Port B
- PORTC (02Hex) Port C
- PORTCL (03Hex) Port C Low (only ME-14A/B)
- PORTCH (04Hex) Port C High (only ME-14A/B)
- PORTD (08Hex) Port D (only B/D versions)
- PORTE (09Hex) Port E (only B/D versions)
- PORTF (0AHex) Port F (only B/D versions)

ü PORTFL (0BHex) Port F Low (only ME-14B)

- PORTFH (0CHex) Port F High (only ME-14B)

<Value>

Output value; possible values are: 0…255 (00Hex…FFHex)

❮ 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 *me14xxGetDrvErrMess*.

5.3.3 Counter Functions

me1400CntInitSrc

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	✔ (not ME-1400/E)	✔

This function determines the clock source for the specified counter. Basically 4 alternatives are available: external clock from D-sub connector, internal oszillator at 1 MHz resp. 10 MHz or the output clock from the previous counter. Note the following restrictions: counter 0 and counter 15 can be sourced from the previous counter and only the first counter of each device can be sourced by the internal clock.

● Definitions

C: int me1400CntInitSrc (int iBoardNumber, int iCounter, int iCounterSource);

Delphi: Function me1400CntInitSrc (iBoardNumber: integer; Counter: integer; CounterSource: integer): integer;

Basic: Declare Function me1400CntInitSrc Lib "me1400" Alias "_VBme1400CntInitSrc@12" (ByVal iBoardNumber As Long, ByVal Counter As Long, ByVal CounterSource As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-1400 (0…31)

<Counter>

Specifies the counter to be configured, possible values are:

- ME-1400A/EA (3 counters): 0…2
- ME-1400B/EB (6 counters):0…5
- ME-1400C (15 counters): 0…14
- ME-1400D (30 counters): 0…29

<CounterSource>

Source of clock signal for specified counter:

- COUNTER_SOURCE_SUBD (00Hex)

  External clock from D-Sub (Default)
  (available for all counters)
- COUNTER_SOURCE_1MHZ (01Hex)

  Internal 1 MHz clock (available for counter 0, 3, 6, 9, 12, 15, 18, 21, 24 or 27)
- COUNTER_SOURCE_10MHZ (02Hex)

  Internal 10 MHz clock (available for counter 0, 3, 6, 9, 12, 15, 18, 21, 24 oder 27)
- COUNTER_SOURCE_PREV (03Hex)

  Clock from previous counter (not available for counter 0 of all models, with counter 3 of ME-1400B/EB and counter 15 of the ME-1400D)

☞ Example

Counter 0 should be sourced internally by the 1 MHz clock and should be cascaded with counter 1. Counter 2 should be sourced externally from the D-sub connector and count independently.

```
iErrorCode =
   me1400CntInitSrc(BoardNumber, 0, COUNTER_SOURCE_1MHZ);

iErrorCode =
   me1400CntInitSrc(BoardNumber, 1, COUNTER_SOURCE_PREV);

iErrorCode =
   me1400CntInitSrc(BoardNumber, 2, COUNTER_SOURCE_SUBD);

...
```

‹ 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 *me14xxGetDrvErrMess*.

me1400CntPWMStart

✎ Description

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	✔ (not ME-1400/E)	✔

This function configures the 3 counter of a counter device (8254) for the operation mode „Pulse Width Modulation“ (PWM) and starts the output. Any previous programming of the counters is overwritten. Any previous programming of the counters is overwritten. The output signal always is provided at the output of counter 2 of the corresponding counter device (OUT_2, OUT_5,…). A base clock (max. 10MHz) must be supplied externally. Alternativly you can use the internal crystal oscillator providing 1 MHz or 10 MHz. Counter 0 can be used as a prescaler (see diagram 17 on page 35). The maximum frequency of the output signal can be 50 kHz maximum. The duty cycle can be set between 1…99% in increments of 1% (see diagram 16 on page 34). The operation is started immediately after calling the function *…CntPWMStart* and is stopped by the function *…CntPWMStop*. No further programming of the counters is required

☞ Note!

Using this function is only useful in combination with external switching shown in diagram 17 on page 35.

● Definitions

VC: me1400CntPWMStart(int iBoardNumber, int iDeviceNumber, int iClockSource, int iPrescaler, int iDutyCycle);

Delphi: Function me1400CntPWMStart(iBoardNumber: integer; iDeviceNumber: integer; iClockSource: integer; iPrescaler: integer; iDutyCycle: integer): integer;

Basic: Declare Function me1400CntPWMStart Lib "me1400" Alias "_VBme1400CntPWMStart@20" (ByVal iBoardNumber As Long, ByVal DeviceNumber As Long, ByVal ClockSource As Long, ByVal Prescaler As Long, ByVal DutyCycle As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-1400 (0…31)

<DeviceNumber>

Counter device whose counter should be configured for PWM operation, possible values depend on board type:

- COUNTER_DEVICE_A Counter device A
- COUNTER_DEVICE_B Counter device B
- COUNTER_DEVICE_C Counter device C
- COUNTER_DEVICE_D Counter device D
- COUNTER_DEVICE_E Counter device E
- COUNTER_DEVICE_F Counter device F
- COUNTER_DEVICE_G Counter device G
- COUNTER_DEVICE_H Counter device H
- COUNTER_DEVICE_I Counter device I
- COUNTER_DEVICE_J Counter device J

<ClockSource>

Source of clock signal for specified counter device:

- COUNTER_SOURCE_SUBD (00Hex)
  External clock from D-Sub
  (available for all counters)
- COUNTER_SOURCE_1MHZ (01Hex)
  Internal 1 MHz clock
- COUNTER_SOURCE_10MHZ (02Hex)
  Internal 10 MHz clock

<Prescaler>

Value for the prescaler (counter 0) within the range: 2…65535.

<DutyCycle>

Duty cycle of the output signal to be set between 1…99% in increments of 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 *me14xxGetDrvErrMess*.

me1400CntPWMStop

✎ Description

Modell:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	✔ (not ME-1400/E)	✔

This function is used to end an operation started with the function *…CntPWMStart*.

● Definitions

VC: me1400CntPWMStop(int iBoardNumber, int iDeviceNumber);

Delphi: Function me1400CntPWMStop(iBoardNumber: integer; iDeviceNumber: integer): integer;

Basic: Declare Function me1400CntPWMStop Lib "me1400" Alias "_VBme1400CntPWMStop@8" (ByVal iBoardNumber As Long, ByVal DeviceNumber As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-1400 (0…31)

<DeviceNumber>

Counter device whose operation should be stopped:

- COUNTER_DEVICE_A Counter device A
- COUNTER_DEVICE_B Counter device B
- COUNTER_DEVICE_C Counter device C
- COUNTER_DEVICE_D Counter device D
- COUNTER_DEVICE_E Counter device E
- COUNTER_DEVICE_F Counter device F
- COUNTER_DEVICE_G Counter device G
- COUNTER_DEVICE_H Counter device H
- COUNTER_DEVICE_I Counter device I
- COUNTER_DEVICE_J Counter device J

‹ 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 *me14xxGetDrvErrMess*

_me14CntRead
me1400CntRead

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔ (not ME-1400/E)	✔

Upon calling this function, the actual counter state is buffered and the value is read in.

● Definitions

C: int me14xxCntRead(int iBoardNumber, int iCounterNo, int *piValue);

Delphi: Function me14xxCntRead(iBoardNumber, iCounterNo: integer; Var iValue: integer): integer;

Basic: Declare Function me14xxCntRead Lib "me14xx_32" Alias "_VBme14xxCntRead@12" (ByVal iBoardNumber As Long, ByVal iCounterNo As Long, iValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<CounterNo>

Specifies the counter to be configured, possible values are:

- ME-1400A/EA (3 counters): 0…2
- ME-1400B/EB (6 counters):0…5
- ME-1400C (15 counters): 0…14
- ME-1400D (30 counters): 0…29

<Value>

16 bit value from specified counter

‹ 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 *me14xxGetDrvErrMess*.

_me14CntWrite
me1400CntWrite

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔ (not ME-1400/E)	✔

This function configures the specified counter for the operation mode and loads the start value. Counting operation starts automatically upon calling this function. See detailled description of the operation modes on page 41ff.

● Definitions

C: int me14xxCntWrite (int iBoardNumber, int iCounterNo, int iMode, int iValue);

Delphi: Function me14xxCntWrite(iBoardNumber, iCounterNo, iMode, iValue: integer): integer;

Basic: Declare Function me14xxCntWrite Lib "me14xx_32" Alias "_VBme14xxCntWrite@16" (ByVal iBoardNumber As Long, ByVal iCounterNo As Long, ByVal iMode As Long, ByVal iValue As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<CounterNo>

Specifies the counter to be configured, possible values are:

- ME-1400A/EA (3 counters): 0…2
- ME-1400B/EB (6 counters):0…5
- ME-1400C (15 counters): 0…14
- ME-1400D (30 counters): 0…29

<Mode>

Operation mode of counter, possible values are:

- Mode 0 (00Hex) "Change of state if zero axis crossing"
- Mode 1 (01Hex) "Retriggerable One-Shot"
- Mode 2 (02Hex) "Asymmetrical divider"
- Mode 3 (03Hex) "Symmetrical divider"
- Mode 4 (04Hex) "Counter start by software trigger"
- Mode 5 (05Hex) "Counter start by hardware trigger"

<Value>

16 bit start value for the specified counter; possible values are: 0…65535 (0000Hex…FFFFHex)

‹ 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 *me14xxGetDrvErrMess*.

me1400InitModeTimerA

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	(only ME-1400A/B)	–

This function should not be used for new applications and is only documented for reasons of downward compatibility (replaced by *me1400CntInitSrc*).

The function configures the clock source for the counters 0…2 of the ME-1400A and ME-1400B.

● Definitions

C: int me1400InitModeTimerA (int iBoardNumber, int iCtrlWordA);

Delphi: Function me1400InitModeTimerA (iBoardNumber: integer; CtrlWordA: integer): integer;

Basic: Declare Function me1400InitModeTimerA Lib "me1400" Alias "_VBme1400InitModeTimerA@8" (ByVal iBoardNumber As Long, ByVal CtrlWordA As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<CtrlWordA>

The control word for counter A will be generated by bit operation „OR“ of the following four constants:

Internal clock frequency

- ME1400_TIMERINTERNCLOCK_1MHZ (00Hex)
  Internal oscillator with 1 MHz (Default)
- ME1400_TIMERINTERNCLOCK_10MHZ (08Hex)
  Internal oscillator with 10 MHz

„OR“

Clock source counter 0

- ME1400_TIMERCLOCKSOURCE0_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE0_INTERN (04Hex)
  Clock from internal oscillator

„OR“

Clock source counter 1

- ME1400_TIMERCLOCKSOURCE1_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE1_OUT0 (02Hex)
  Clock from output counter 0

Clock source counter 2

„OR“

- ME1400_TIMERCLOCKSOURCE2_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE2_OUT1 (01Hex)
  Clock from output counter 1

☞ Example

Counter 0 should be sourced by the internal 1 MHz clock and all the 3 counters of component A should be cascaded:

```
iErrorCode = me1400InitModeTimerA(iBoardNumber,
    ME1400_TIMERINTERNCLOCK_1MHZ |
    ME1400_TIMERCLOCKSOURCE0_INTERN |
    ME1400_TIMERCLOCKSOURCE1_OUT0 |
    ME1400_TIMERCLOCKSOURCE2_OUT1);
```

❮ 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 *me14xxGetDrvErrMess*.

me1400InitModeTimerB

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	–	(only ME-1400B)	–

This function should not be used for new applications and is only documented for reasons of downward compatibility (replaced by me1400CntInitSrc).

The function configures the clock source for the counters 3…5 of the ME-1400B.

● Definitions

C: int me1400InitModeTimerB (int iBoardNumber, int iCtrlWordB);

Delphi: Function me1400InitModeTimerB (iBoardNumber: integer; CtrlWordB: integer): integer;

Basic: Declare Function me1400InitModeTimerB Lib "me1400" Alias "_VBme1400InitModeTimerB@8" (ByVal iBoardNumber As Long, ByVal CtrlWordB As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<CtrlWordB>

The control word for counter B will be generated by OR-linking the following four constants:

Internal clock frequency

- ME1400_TIMERINTERNCLOCK_1MHZ (00Hex)
  Internal oscillator with 1 MHz (Default)
- ME1400_TIMERINTERNCLOCK_10MHZ (08Hex)
  Internal oscillator with 10 MHz

„OR“

Clock source counter 3

- ME1400_TIMERCLOCKSOURCE0_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE0_INTERN (04Hex)
  Clock from internal oscillator

„OR“

Clock source counter 4

- ME1400_TIMERCLOCKSOURCE1_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE1_OUT0 (02Hex)
  Clock from output counter 3

„OR“

Clock source counter 5

- ME1400_TIMERCLOCKSOURCE2_SUBD (00Hex)
  Ext. clock from D-Sub (Default)
- ME1400_TIMERCLOCKSOURCE2_OUT1 (01Hex)
  Clock from output counter 4

☞ Example

Counter 3 should be sourced by the internal 1 MHz clock and all the 3 counters of component B should be cascaded:

```
iErrorCode = me1400InitModeTimerB(iBoardNumber,
    ME1400_TIMERINTERNCLOCK_1MHZ |
    ME1400_TIMERCLOCKSOURCE0_INTERN |
    ME1400_TIMERCLOCKSOURCE1_OUT0 |
    ME1400_TIMERCLOCKSOURCE2_OUT1);
```

‹ 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 *me14xxGetDrvErrMess*.

5.3.4 Interrupt Handling

_me14DisableInt
me1400DisableInt

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔ (not ME-1400/E)	✔

The external interrupt input (OSC/IR_IN) on the board previously started by *…EnableInt* is disabled.

☞ Important Note:

In Agilent VEE, use this function only in connection with *me1400GetIrqCnt*!

● Definitions

C: int me14xxDisableInt (int iBoardNumber, int iServiceNo);

Delphi: Function me14xxDisableInt (iBoardNumber, iServiceNo: integer): integer;

Basic: not realized

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<ServiceNo>

Interrupt channel; a „1“ must be passed

❮ 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 *me14xxGetDrvErrMess*.

_me14EnableInt
me1400EnableInt

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔ (not ME-1400/E)	✔

The external interrupt input (OSC/IR IN) is enabled. When an interrupt occurs the user defined interrupt service routine is automatically processed. On every function call of *…EnableInt* the function *…DisableInt* must be called at the end of the program. If this function is used with *me1400GetIrqCnt,* a null pointer is passed instead of the interrupt routine.

☞ Important Note:

In Agilent VEE, use this function only in connection with *me1400GetIrqCnt*!

● Definitions

C: int me14xxEnableInt (int iBoardNumber, pSERVICE_PROC IrqFunc, int ServiceNo)

Delphi: Function me14xxEnableInt (iBoardNumber: integer; IrqFunc: Pointer; iServiceNo: integer): integer;

Basic: not realized

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<IrqFunc>

Address of a user defined function of type (void SERVICE_PROC (void)) in C resp. of type pointer in Delphi, which is processed if an interrupt occures.

<ServiceNo>

Interrupt channel; a „1“ must be passed

‹ 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 *me14xxGetDrvErrMess*.

me1400GetIrqCnt

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔ (not ME-1400/E)	✔

This function acquires the number of interrupts since the device is running. Use this function to provide interrupt control for graphic programming languages like Agilent VEE or LabVIEW™. As usual the interrupt control must be enabled resp. disabled by the functions *me14xxEnableInt* and *me14xxDisableInt*. By reading the value of the parameter `IrqCnt` it is possible to determine whether an interrupt was initiated or not, relative to a previous request.

● Definitions

C: int me1400GetIrqCnt (int iBoardNumber, int* piIrqCnt);

Delphi: Function me1400GetIrqCnt (iBoardNumber: integer; Var iIrqCnt: integer): integer;

Basic: Declare Function me1400GetIrqCnt Lib "me1400" Alias "_VBme1400GetIrqCnt@8" (ByVal iBoardNumber As Long, ByRef iIrqCnt As Long) As Long

➔ Parameter

<BoardNumber>

Number of the board to be accessed of type ME-14 (0…3) resp. ME-1400 (0…31)

<IrqCnt>

Number of interrupts since the device is running

☞ Example

```
iErrorCode = me1400SetMultifunctionPin(0, ME1400_MULTIPIN_IRQ);
iErrorCode = me1400EnableInt(0, 0, 1);
iErrorCode = me1400GetIrqCnt(0, &iIrqCntBefore);

Sleep(1000);           //waiting for interrupts

iErrorCode = me1400GetIrqCnt(0, &iIrqCntAfter);
iErrorCode = me1400DisableInt(0, 1);
IrqCnt = (iIrqCntAfter-iIrqCntBefore);
```

❮ 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 *me14xxGetDrvErrMess*.

5.3.5 Error Handling

_me14GetDrvErrMess
me1400GetDrvErrMess

✎ Description

Model:	ME-14A/B	ME-1400/A/B/E	ME-1400C/D
available:	✔	✔	✔

If an error occurs during the processing of the previously called API function of the ME-14/1400 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 `ME14_32.DLL` or `ME1400.DLL` returned an error code (i. e. error code 0)!

● Definitions

C: int me14xxGetDrvErrMess (char *pcErrortext);

Delphi: Function me14xxGetDrvErrMess (Var errortext: errorstring): integer;

Basic: Declare Function me14xxGetDrvErrMess Lib "me14xx_32" Alias "_VBme14xxGetDrvErrMess@4" (ByVal errortext As String) As Long

➔ Parameter

`<Errortext>`

Pointer to a string; the return value is the error code.

❮ Return value

0 if there was no error or matching error code.

Appendix

A Specifications

PCI/cPCI Interface (ME-1400/A/B/C/D/E/EA/EB)

Bus system	Standard PCI (32 Bit, 33 MHz);
(depends on model)	CompactPCI (32 bit, 33 MHz)
Plug&Play functionallity	Automatic assignment of resources

ISA Interface (ME-14A/B)

Bus system	ISA bus (8 bit);
Base address	0Hex…3F0Hex in 10Hex steps (set by DIP switches)
Address space	8 bytes (ME-14A), 16 bytes (ME-14B)
Interrupt channel	IRQ2…7 set by jumper
Wait-States	On/Off (by jumper)

Digital I/O

Number	ME-14A, ME-1400/A/C/E/EA: 24, TTL compatible ME-14B, ME-1400B/D/EB: 48, TTL compatible
Type	82(C)55 in mode 0
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)

Counter

Number	ME-14A, ME-1400A/EA: 3 independent ME-14B, ME-1400B/EB: 6 independent ME-1400C: 15 independent ME-1400D: 15 additionally to ME-1400C
Type	82(C)54
Resolution	16 bit
Input voltage	Low: -0,5 V…+0,8 V (I_{ILmax}=±10 µA) High: +2,2 V…+6 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)

Crystal oszillator

Frequency	1 MHz or 10 MHz selectable (ISA: by jumper; PCI/cPCI: by software)
Accuracy	±100 ppm (±0,01%)
Output level	LS-TTL

General Information

Power consumption at +5 V (without load)	ME-14A: typ. 400 mA ME-14B: typ. 450 mA ME-1400: typ. 200 mA ME-1400A: typ. 220 mA ME-1400B: typ. 400 mA ME-1400C: typ. 1 A ME-1400D: typ. 800 mA ME-1400E: typ. 200 mA ME-1400EA: typ. 220 mA ME-1400EB: typ. 400 mA
VCC loading at the D-Sub connector	ME-14 ISA: 50 mA ME-1400 PCI/cPCI: 200 mA
Physical size (without mounting bracket and connector)	ME-14A: 140 x 106 mm ME-14B: 166 x 106 mm ME-1400/A/B: 132 x 99 mm ME-1400C/D: 129 x 99 mm ME-1400E/EA/EB: 175 x 99 mm cPCI models: 100 x 160 mm
Connectors	**ME-14A/B, ME-1400E/EA/EB:** 37pin D-Sub female connector at the mounting bracket of the board **additional for ME-14B, ME-1400EB:** 40pin IDC-connector to a 37pin D-Sub female connector on an additional mounting bracket (for pinout see D-Sub at the mounting bracket of the board) **ME-1400/A/B/C/D:** 78pin D-Sub female connector at the mounting bracket of the board
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 ME-1400/A/B

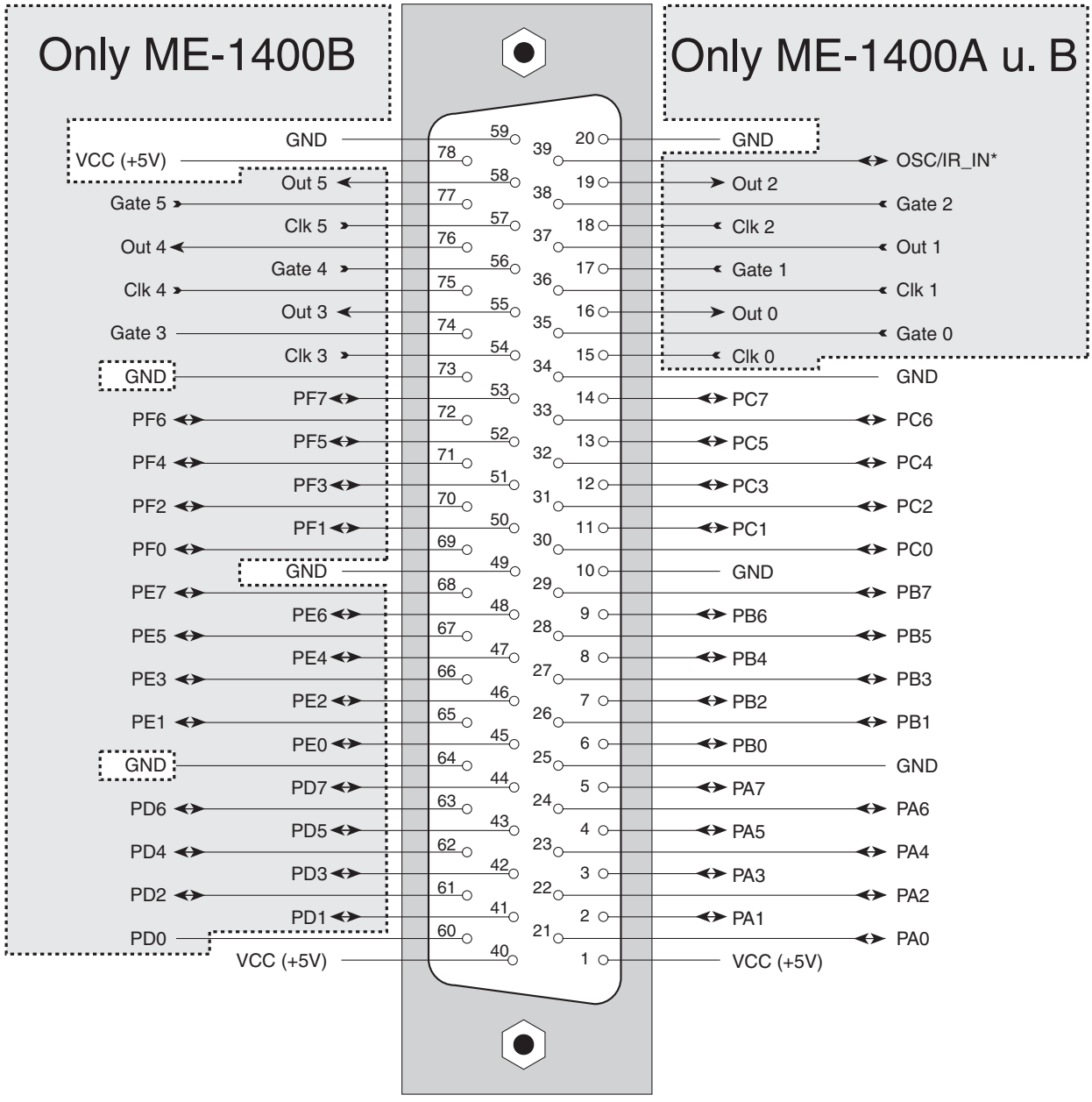


Diagram 19: 78pin female D-Sub connector ME-1400/A/B

*Only in operation on ME-1400/A/B.

B2 ME-1400C/D

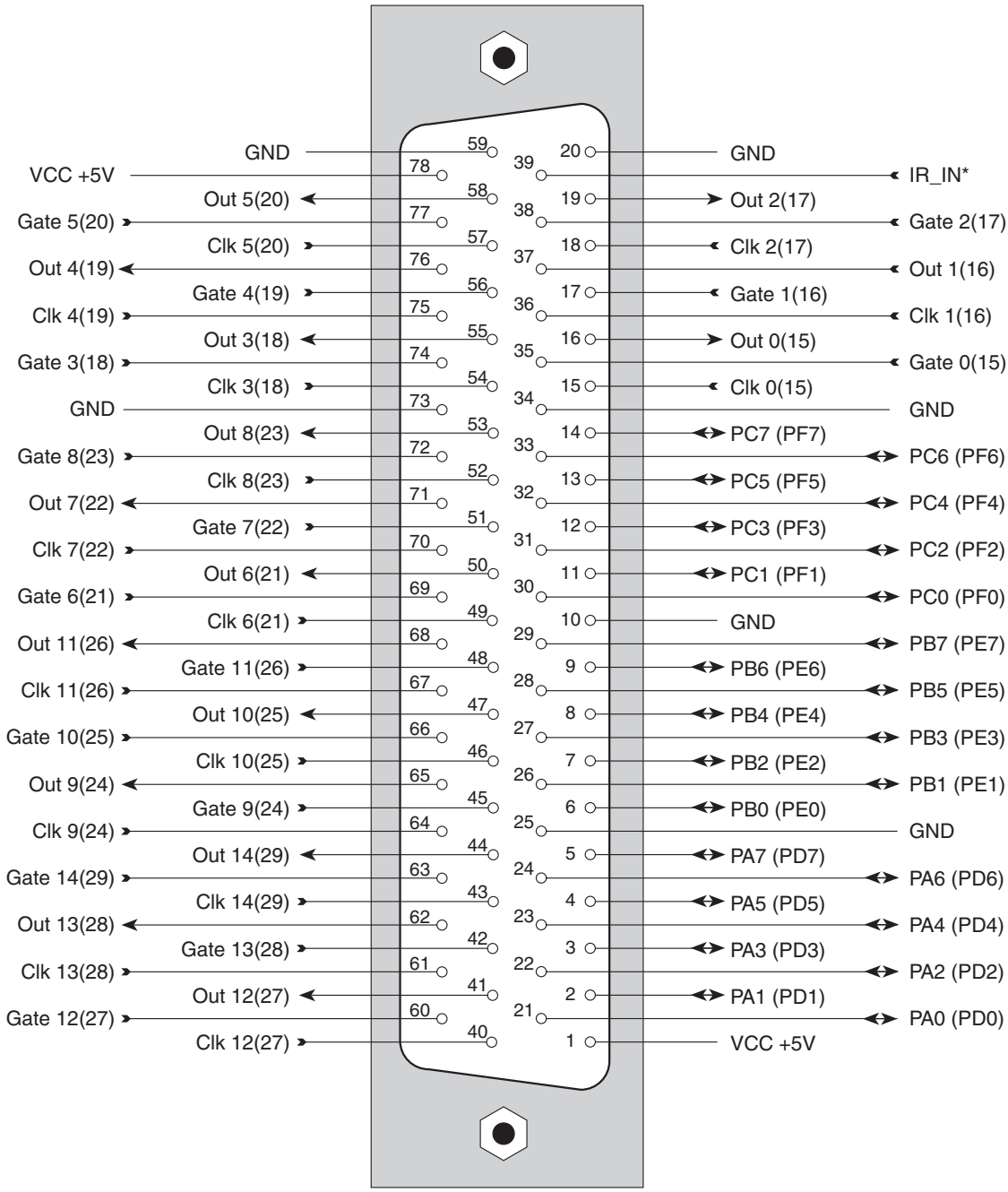


Diagram 20: 78pin female D-Subconnector ME-1400C/D

Note:
The ports D, E and F (in brackets) only available with the expansion board ME-1400D.

* The interrupt input „IR_IN“ is only available for the ME-1400C. **Please don´t connect** this pin on the ME-1400D!

B3 Special Cable for ME-1400C/D

Description	Color	Signal	D-Sub Pin	Description	Color	Signal	D-Sub Pin
Counter 0 (T0)	black	Clk 0	15	Counter 9 (T9)	black	Clk 9	64
	brown	Out 0	16		brown	Out 9	65
	red	Gate 0	35		red	Gate 9	45
Counter 1 (T1)	black	Clk 1	36	Counter 10 (T10)	black	Clk 10	46
	brown	Out 1	37		brown	Out 10	47
	red	Gate 1	17		red	Gate 10	66
Counter 2 (T2)	black	Clk 2	18	Counter 11 (T11)	black	Clk 11	67
	brown	Out 2	19		brown	Out 11	68
	red	Gate 2	38		red	Gate 11	48
Counter 3 (T3)	black	Clk 3	54	Counter 12 (T12)	black	Clk 12	40
	brown	Out 3	55		brown	Out 12	41
	red	Gate 3	74		red	Gate 12	60
Counter 4 (T4)	black	Clk 4	75	Counter 13 (T13)	black	Clk 13	61
	brown	Out 4	76		brown	Out 13	62
	red	Gate 4	56		red	Gate 13	42
	orange	GND	73	Counter 14 (T14)	black	Clk 14	43
Counter 5 (T5)	black	Clk 5	57		brown	Out 14	44
	brown	Out 5	58		red	Gate 14	63
	red	Gate 5	77	IRQ	black	Vcc	78
Counter 6 (T6)	black	Clk 6	49		brown	GND	59
	brown	Out 6	50		red	GND	20
	red	Gate 6	69		orange	IR_IN	39
Counter 7 (T7)	black	Clk 7	70				
	brown	Out 7	71				
	red	Gate 7	51				
Counter 8 (T8)	black	Clk 8	52				
	brown	Out 8	53				
	red	Gate 8	72				

(Digital ports see next page)

Table 21: Special cable ME-1400C/D

Special cable ME-1400C/D (continued)

Description	Color	Signal	D-Sub Pin	Description	Color	Signal	D-Sub Pin
DIO	white	PA0	21	DIO (continued)	orange	PC0	30
	white/black	PA1	2		orange/black	PC1	11
	black	PA2	22		orange/white	PC2	31
	black/white	PA3	3		orange/brown	PC3	12
	brown	PA4	23		yellow	PC4	32
	brown/white	PA5	4		yellow/black	PC5	13
	purple	PA6	24		yellow/white	PC6	33
	purple/white	PA7	5		yellow/brown	PC7	14
	red	PB0	6		green	Vcc	1
	red/black	PB1	26		green/black	GND	25
	red/white	PB2	7		green/white	GND	10
	red/brown	PB3	27		green/brown	GND	34
	pink	PB4	8				
	pink/black	PB5	28				
	pink/white	PB6	9				
	pink/brown	PB7	29				

Table 22: Special cable ME-1400C/D (continued)

B4 ME-14A/B, ME-1400E/EA/EB

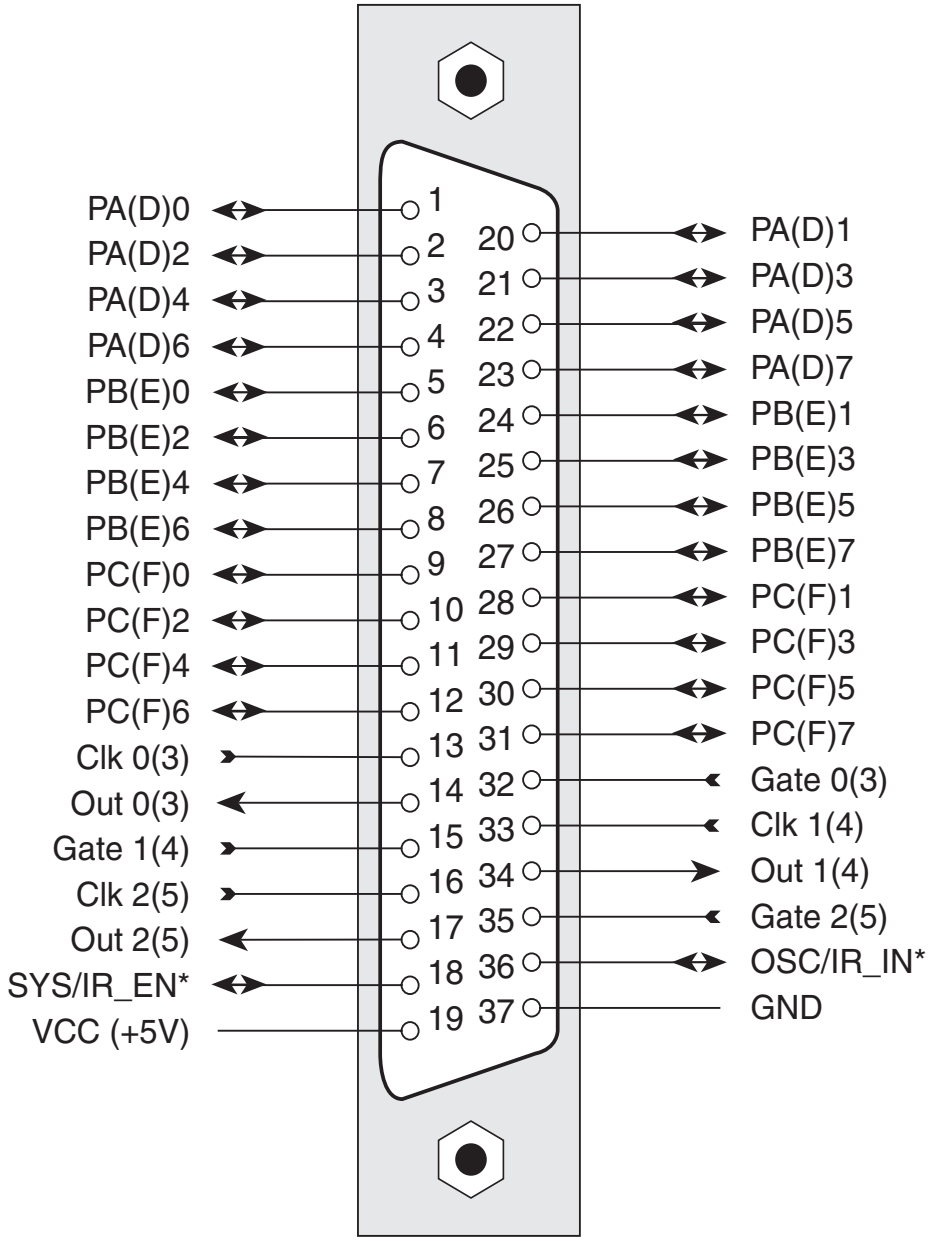


Diagram 21: Pinout of the 37pin female D-Sub

Note:
Ports D, E and F (in brackets) are only available on B-versions in combination with an additional mounting bracket (included with the package), see also B5 and B6.

* Functional overview see table on the next page.

B5 IDC Connector for B-Versions (ST2)

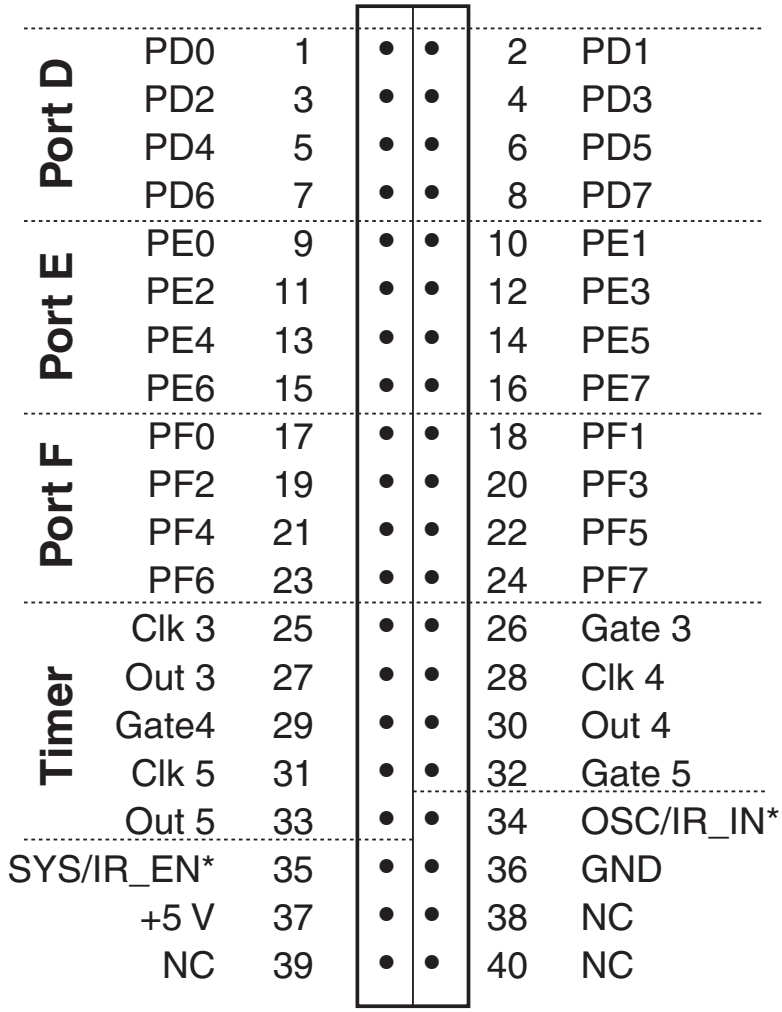


Abb. 22: Pinout of the 40pin IDC-connector

	37pin D-Sub		40pin IDC connector (ST2)	
	SYS/IR_EN (Pin 18)	OSC/IR_IN (Pin 36)	SYS/IR_EN (Pin 35)	OSC/IR_IN (Pin 34)
ME-14A	✔/✔	✔/✔	–	–
ME-14B	✔/✔	✔/✔	✔/❍	✔/❍
ME-1400E	n.c.	–	–	–
ME-1400EA	n.c.	✔/✔	–	–
ME-1400EB	n.c.	✔/✔	n.c.	n.c.

❍ Not supported by Windows driver.

B6 Additional Mounting Bracket

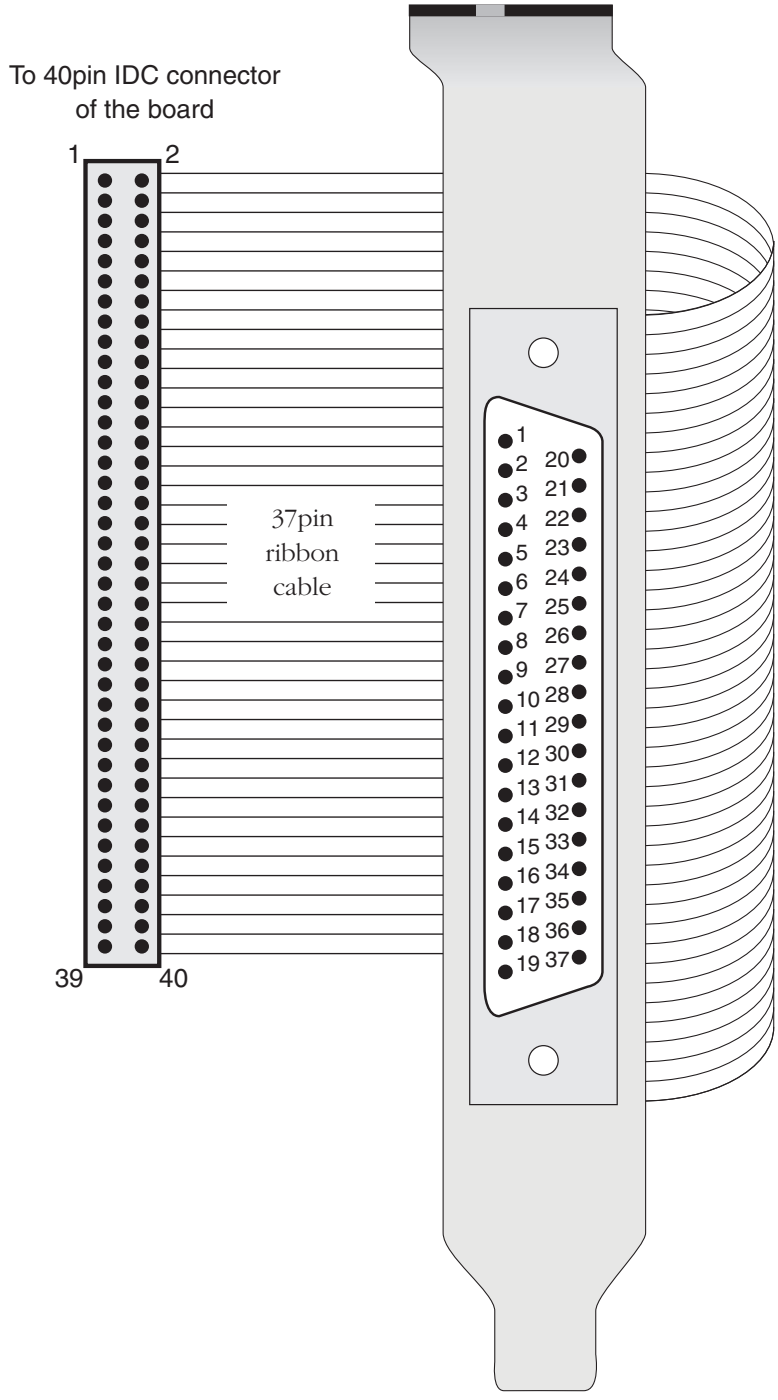


Diagram 23: Mounting bracket with female D-Sub for ME-14B and ME-1400EB (schematic diagram, see Diagram 21: for pinout)

C Accessories

Optionally the following products are available:

ME-63Xtend Series

External relay and digital-I/O boards for direct connect to ME-1400/A/B via an 1:1 cable (e. g. ME AK-D78).

ME-UB Series

External connector, relay and digital-I/O boxes for direct connect to ME-1400/A/B via the special cable ME AK-D7815/1400.

ME AB-D37M

37pin D-Sub connector block (male) for ME-14A/B, ME-1400E

ME AK-D37

37pin D-Sub cable (male - female), 2 m, for ME-14A/B, ME-1400E

ME AB-D78M

78pin D-Sub connector block (male) for ME-1400, ME-1400A, ME-1400B, ME-1400C, ME-1400D

ME AK-D78 (/1)

78pin D-Sub cable (male - female) for ME-1400, ME-1400A, ME-1400B, ME-1400C, ME-1400D; length: 2 resp. 1 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 a fax to our hotline:

Fax hotline: ++ 49 (0) 89/89 01 66 28
eMail: support@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 for Meilhaus boards and our manuals in PDF format are available under www.meilhaus.com.

E Index