Modular Computers ®

CP380

Digital I/O Board
for CompactPCI Applications

Manual ID 19313, Rev. Index 0100
Feb 99

The product described in this manual is in compliance with all applied CE standards.

This page was intentionally left blank.

Preface

Revision History ..... *0 - 4*
Trademarks ..... *0 - 4*
Explanation of Symbols ..... *0 - 5*
For Your Safety ..... *0 - 6*
High Voltage Safety Instructions ..... *0 - 6*
Special Handling and Unpacking Instructions ..... *0 - 6*
General Instructions on Usage ..... *0 - 7*
Two Years Warranty ..... *0 - 8*
Table of Contents ..... *0 - 9*

Revision History

Revision History			
Manual/Product Title:	**CP380**		
Manual ID Number:	**19313**		
Rev. Index	**Brief Description of Changes**	**Board Index**	**Date of Issue**
0100	Initial Issue	01	Feb. 99

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

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

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

Trademarks

PEP Modular Computers, the *PEP* logo and “CXM” are trade marks owned by *PEP Modular Computers GmbH*, Kaufbeuren (Germany). In addition, this document may include names, company logos and trademarks, which are registered trademarks and, therefore, proprietary of their respective owners.

Explanation of Symbols



CE Conformity

This symbol indicates that the item described in this manual is in compliance with all applied CE standards. See also the section “Applied standards” of this manual.



Caution!

This symbol and title warn you of hazards due to electrical shocks (> 60 V) when touching products or parts of them. The non-observance of the measures indicated and/or prescribed by the law may cause harm to your product and/or life/health.

See also the section “High Voltage Safety Instructions”.



ESD-Sensitive Device!

This symbol and title inform you that electronic boards and their components are sensitive to static electricity. Therefore, care must be taken during all handling operations and inspections of this product, in order to ensure product integrity at all times.

Please read also the section “Special Handling and Unpacking Instructions” of this manual.



Attention!

This symbol and title emphasize aspects which, if not read through carefully by the reader, might cause hazards to health and/or damages to material.



Note:

This symbol and title emphasize aspects the reader should read through carefully for his or her own advantage.



PEP Advantage

This symbol and title emphasize advantages or positive aspects of a product and/or procedure.



Troubleshooting

This symbol and title characterize a message containing useful information on troubleshooting and problem solving.

For your safety

Your new *PEP* product was developed and tested carefully to provide all features necessary to ensure the renown electrical safety requirements. It was also designed for a long fault-free life. However, the life expectancy of your product can be drastically reduced by improper treatment during unpacking and installation. Therefore, in the interests of your own safety and of the correct operation of your new *PEP* product, you are requested to conform with the following guidelines.

High Voltage Safety Instructions



Warning!

All operations on this device must be carried out by sufficiently skilled personnel only.



Caution!

However, serious electrical shock hazards exist during all installation, repair and maintenance operations with this product. Therefore, always unplug the power cable to avoid exposure to hazardous voltage.
Before installing your new *PEP* product into a system always ensure that your mains power is switched off. This applies also to the installation of piggybacks.

Special Handling and Unpacking Instructions



ESD Sensitive Device!

Electronic boards and their components are sensitive to static electricity. Therefore, care must be taken during all handling operations and inspections of this product, in order to ensure product integrity at all times.

- Do not handle this product out of its protective enclosure while it is not used for operational purposes, unless it is otherwise protected.
- Whenever possible, unpack or pack this product only at EOS/ESD safe work stations. Where safe work stations are not guaranteed, it is important for the user to be electrically discharged before touching the product with his/her hands or tools. This is most easily done by touching a metal part of your system housing.
- It is particularly important to observe standard anti-static precautions when changing piggybacks, ROM devices, jumper settings etc. If the product contains batteries for RTC or memory back-up, ensure that the board is not placed on conductive surfaces, including anti-static plastics or sponges. They can cause short circuits and damage the batteries or tracks on the board.

General Instructions on Usage

- In order to maintain *PEP's* product warranty, this product must not be altered or modified in any way. Changes or modifications to the device, which are not explicitly approved by *PEP Modular Computers* and described in this manual or received from *PEP* Technical Support as a special handling instruction, will void your warranty.
- This device should only be installed in or connected to systems that fulfill all necessary technical and specific environmental requirements. This applies also to the operational temperature range of the specific board version, which must not be exceeded. If batteries are present, their temperature restrictions must be taken into account.
- In performing all necessary installation and application operations, please, follow only the instructions supplied by the present manual.
- Keep all the original packaging material for future storage or warranty shipments. If it is necessary to store or ship the board please re-pack it as nearly as possible in the manner in which it was delivered.
- Special care is necessary when handling or unpacking the product. Please, consult the special handling and unpacking instruction on the previous page of this manual.

Two Years Warranty

PEP Modular Computers grants the original purchaser of *PEP* products a ***TWO YEARS LIMITED HARDWARE WARRANTY*** as described in the following. However, no other warranties that may be granted or implied by anyone on behalf of *PEP* are valid unless the consumer has the express written consent of *PEP Modular Computers*.

PEP Modular Computers warrants their own products, excluding software, to be free from manufacturing and material defects for a period of 24 consecutive months from the date of purchase. This warranty is not transferable nor extendible to cover any other users or long-term storage of the product. It does not cover products which have been modified, altered or repaired by any other party than *PEP Modular Computers* or their authorized agents. Furthermore, any product which has been, or is suspected of being damaged as a result of negligence, improper use, incorrect handling, servicing or maintenance, or which has been damaged as a result of excessive current/voltage or temperature, or which has had its serial number(s), any other markings or parts thereof altered, defaced or removed will also be excluded from this warranty.

If the customer's eligibility for warranty has not been voided, in the event of any claim, he may return the product at the earliest possible convenience to the original place of purchase, together with a copy of the original document of purchase, a full description of the application the product is used on and a description of the defect. Pack the product in such a way as to ensure safe transportation (see our safety instructions).

PEP provides for repair or replacement of any part, assembly or sub-assembly at their own discretion, or to refund the original cost of purchase, if appropriate. In the event of repair, refunding or replacement of any part, the ownership of the removed or replaced parts reverts to *PEP Modular Computers*, and the remaining part of the original guarantee, or any new guarantee to cover the repaired or replaced items, will be transferred to cover the new or repaired items. Any extensions to the original guarantee are considered gestures of goodwill, and will be defined in the "Repair Report" issued by *PEP* with the repaired or replaced item.

PEP Modular Computers will not accept liability for any further claims resulting directly or indirectly from any warranty claim, other than the above specified repair, replacement or refunding. In particular, all claims for damage to any system or process in which the product was employed, or any loss incurred as a result of the product not functioning at any given time, are excluded. The extent of *PEP Modular Computers* liability to the customer shall not exceed the original purchase price of the item for which the claim exists.

PEP Modular Computers issues no warranty or representation, either explicit or implicit, with respect to its products' reliability, fitness, quality, marketability or ability to fulfil any particular application or purpose. As a result, the products are sold "as is," and the responsibility to ensure their suitability for any given task remains that of the purchaser. In no event will *PEP* be liable for direct, indirect or consequential damages resulting from the use of our hardware or software products, or documentation, even if PEP were advised of the possibility of such claims prior to the purchase of the product or during any period since the date of its purchase.

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

Table of Contents

Chapter 1

1. Introduction ..... 1 - 1
1.1 System Overview ..... 1 - 3
1.2 System Components ..... 1 - 4
1.3 CP380 Models ..... 1 - 7
1.4 System Requirements ..... 1 - 7
1.5 Board Layouts ..... 1 - 8
1.6 Block Diagrams ..... 1 - 9
1.7 General Notes ..... 1 - 10
1.8 Optocouplers ..... 1 - 12
1.9 Specifications ..... 1 - 13
1.10 Applied Standards ..... 1 - 14
1.11 Related Publications ..... 1 - 15

Chapter 2

2. Installation ..... 2 - 3
2.1 Hardware Installation ..... 2 - 3
2.2 Driver Installation ..... 2 - 3
2.3 Uninstall ..... 2 - 4
2.4 Test Program ..... 2 - 5
2.5 Software ..... 2 - 6

Chapter 3

3. Configuration ..... 3 - 3
3.1 Jumper Settings ..... 3 - 3
3.2 Pinouts ..... 3 - 3

Chapter 4

4. Programming ..... 4 - 3
4.1 High Level Language Programming ..... 4 - 3
4.2 HP VEE Programming ..... 4 - 4
4.3 LabVIEW™-Programming ..... 4 - 5

Chapter 5

5. Function Reference ..... 5 - 3
5.1 Functional Overview of the 32 Bit Driver ..... 5 - 3
5.2 Naming Conventions ..... 5 - 3
5.3 Definitions ..... 5 - 3
5.4 Description of the API Functions ..... 5 - 4

Chapter 1

Introduction

1.1 System Overview ..... 1 - 3
1.1.1 Note on CompactPCI ..... 1 - 3
1.1.2 Main Features of CompactPCI Systems ..... 1 - 3
1.2 System Components ..... 1 - 4
1.3 CP380 Models ..... 1 - 7
1.4 System Requirements ..... 1 - 7
1.5 Board Layouts ..... 1 - 8
1.6 Block Diagrams ..... 1 - 9
1.7 General Notes ..... 1 - 10
1.7.1 Modes of Operation ..... 1 - 10
1.7.2 Wiring ..... 1 - 11
1.8 Optocouplers ..... 1 - 12
1.8.1 External Power for the Optocouplers ..... 1 - 12
1.9 Specifications ..... 1 - 13
1.10 Applied Standards ..... 1 - 14
1.10.1 CE Compliance ..... 1 - 14
1.10.2 Mechanical Compliance ..... 1 - 14
1.10.3 Environmental Tests ..... 1 - 14
1.11 Related Publications ..... 1 - 15
1.11.1 CompactPCI Systems/Boards ..... 1 - 15
1.11.2 PMC Add-On Modules/Carriers ..... 1 - 15

This page was intentionally left blank.

1. Introduction

1.1 System Overview

The *PEP Modular Computers* CompactPCI systems described in this chapter operate with the PCI bus architecture to support additional I/O and memory-mapped devices as required by various industrial applications. In the following you will find the most important information on all system relevant CompactPCI features. For more detailed information concerning the CompactPCI standard, please consult the complete Peripheral Component Interconnect (PCI) and CompactPCI Specifications. For further information regarding these standards and their use, visit the homepage of the *PCI Industrial Computer Manufacturers Group (PICMG)*.

1.1.1 Note on CompactPCI

CompactPCI is an extension of the PCI specification. It has been optimized for industrial and embedded applications that require a more robust mechanical form factor as compared to Desktop PCI. CompactPCI systems use industry standard mechanical components and high performance connector technologies to provide systems that are well suited for rugged applications. CompactPCI stands for systems that are electrically compatible with the PCI Specification, allowing low cost PCI components to be used. CompactPCI is an open specification supported by PICMG.

1.1.2 Main Features of CompactPCI Systems

Some of the outstanding features of the CompactPCI systems compliant with Specification 2.0, Release 2.1 are:

- PCI signalling
- 32 and 64 bit data transfer at 33 MHz
- up to 8 PCI slots per backplane
- industry standard software support
- 3U small form factor (100 mm by 160 mm)
- 6U form factor (233 mm by 160 mm)
- Eurocard packaging
- wide variety of available I/O functions
- industry support from over 350 members.
- possibility of PMC modules on some boards



PEP Advantage

PEP Modular Computers' CompactPCI systems are designed as open systems, able to be expanded at any time, so that their backplanes can be equipped with precisely the CPCI boards that correspond to a customer's specific needs. However, in order to supply you with an appropriate choice of workstations, the *PEP* basic equipment is divided into pre-configured sets and custom solutions.

1.2 System Components

PEP Modular Computers have devised their CompactPCI systems as a comprehensive open solution for industrial environments, offering different workstation configurations which are capable of including all the components necessary to fulfill the requirements of virtually all existing system functionalities.

CompactPCI Backplane(s)

- 4-slot backplane;
- 6-slot backplane;
- 8-slot backplane;
- 16-slot backplane.

CPU Function

PEP Modular Computers provides CPU boards corresponding in size and characteristics to the special features of the *PEP* systems.

Display Related Functions

Display related functions such as frame grabbing are supported by dedicated boards.

Communication

Communication boards are provided for the currently relevant industrial communication systems such as Fast Ethernet.

Fieldbus Control

Fieldbus control boards provide data exchange with field control and automation subsystems like *PEP Modular Computers'* SMART2 or VMEbus systems using up-to-date transmission standards for example:

- CAN fieldbus control
- PROFIBUS control

Industrial I/O Functions

PEP's CompactPCI systems support an ever increasing number of industrial I/O functionalities in the fields of:

- Digital I/O
- Analog I/O

Self-Testing

A special *PEP* self-testing board provides a trouble-spotting capability within your *PEP* system.

System Hardware

The *PEP* CompactPCI system hardware includes housings, storage devices, power supply units, network adapters etc. The most important system hardware elements with which to configure your CompactPCI system are as follows:

Multiprocessor

In a CompactPCI multi-processer system, a system controller (CP610) communicates with other CPU's through a non-transparent PCI / PCI bridge

Housings

- Board cages for 19-inch cabinets or wall mounting
- PCI tower
- PCI desktop housing

Mass Storage Devices

- Hard-disk drive
- Floppy-disk drive
- CD-ROM drive

Power Supply Units

- 19-inch "M", 180 W, 120/230 V AC, 3.3V/14A, 5.1V/20A, 12V/2A, -12V/1A
- "ATX", 235 W, 120/230 VAC, 3.3V/14A, 5V/22A, 12V/8A,-12V/1A, -5V/0.5A, 5VSB/0.1A

Form Factors

PEP Modular Computers CompactPCI boards are available in different form factors, mainly as 3U (single-height) and 6U (double-height) boards and as 4HP (single-width) and 8HP (double-width) boards.

Please note that the following units of measurement are used to express the dimensions of *PEP* CompactPCI card slots:

- Height: 1 U = 44.45 mm
- Width: 1 HP = 5.08 mm

On certain boards/carriers with PMC connectors, PMC add-on modules can also be installed. Please note that the form factor of *PEP Modular Computers* PMC modules is:

- 149mm x 74mm (single-size standard).

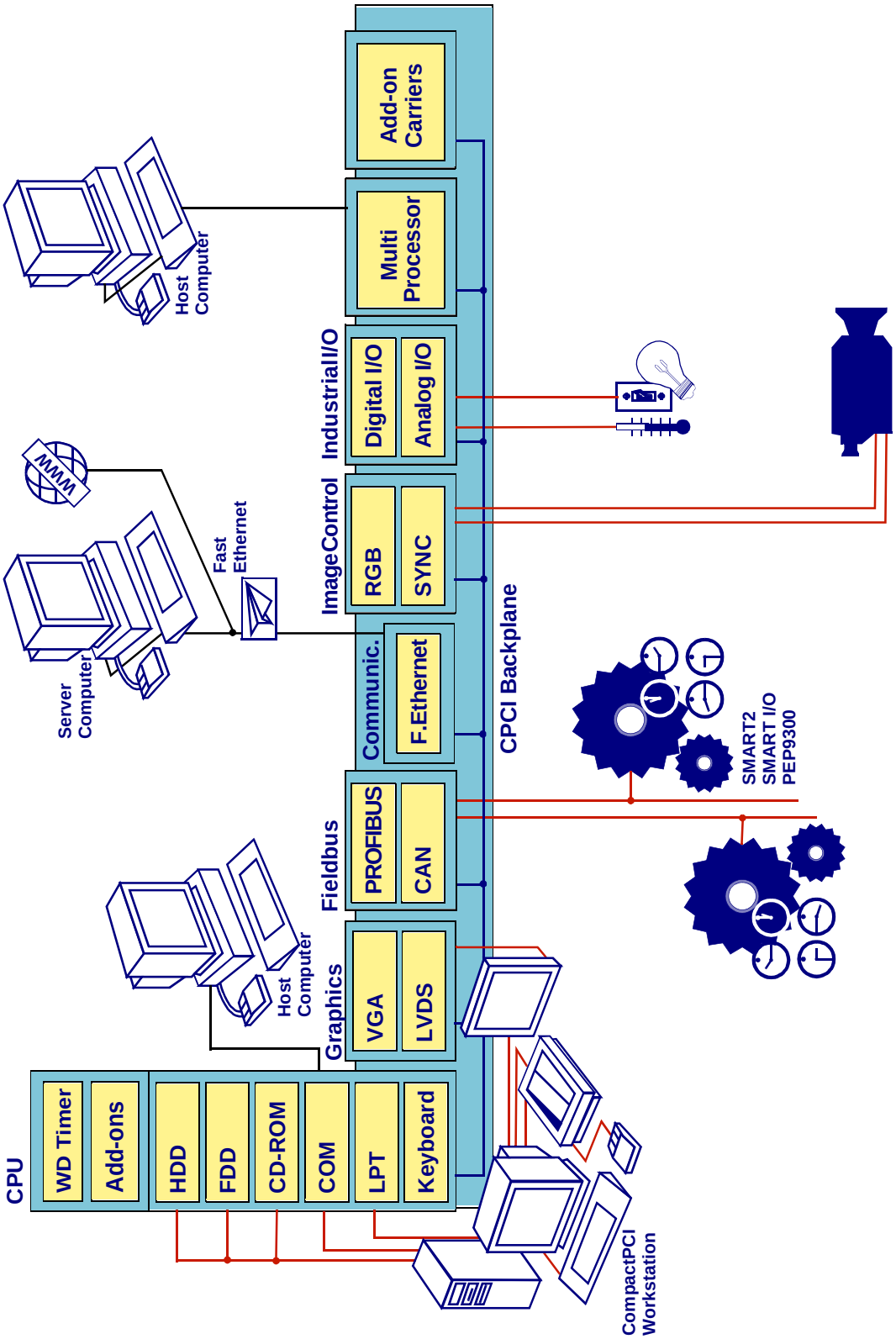
For a detailed description of the *PEP Modular Computers* CompactPCI modules please consult the specific component's manuals or data sheets.



PEP Advantage

As the *PEP Modular Computers* CompactPCI system provides comprehensive open solutions. New features and functionalities may be added to our range. To keep abreast of the latest developments, please contact your local *PEP Sales Office* or visit the ***PEP Web Site***.

Figure 1-1: Example of a *PEP Modular Computers* CompactPCI System



1.3 CP380 Models

The CP380 board family includes electrically isolated digital I/O boards for the CompactPCI System bus.

The electrical isolation on the boards prevents interference and high voltages (e. g. voltage peaks on the ground lines) from reaching the PC electronics. The inputs have been designed for the usual voltage, typically 24 V.

On the CP380-I and CP380-I/O there are 2 modes of operation: with or without interrupt control. The interrupt channel and base address are configured by the BIOS.

The software supplied with the board allows quick and easy installation of the board into data acquisition and control applications. PEP support is provided for Windows NT, HP VEE (Hewlett-Packard) and LabVIEW™ (National Instruments) (drivers are available on disk).

Table 1-1: Models in the CP380 Range

Model	Opto-Isolated Inputs	Opto-Isolated Outputs
CP380-I	18, typical voltage 24 V. Two of the inputs can be used as interrupt channels (i. e. 16 inputs and 2 strobes)	---
CP380-O	---	16, typical voltage 24 V; defined levels at power- up and power-down
CP380-I/O	10, typical voltage, 24 V. One input can be used as an interrupt channel, but then only the 8 inputs of data port B are available (i.e. 8 inputs and 1 strobe).	8, typical voltage 24V; defined levels at power-on and power-off

1.4 System Requirements

All the CP380 versions are designed for installation into CompactPCI systems only (e.g. the CP-ASM 3,4,6,7, Tower 3).

1.5 Board Layouts

Figure 1-2: CP380-I and CP380-O Board Layout

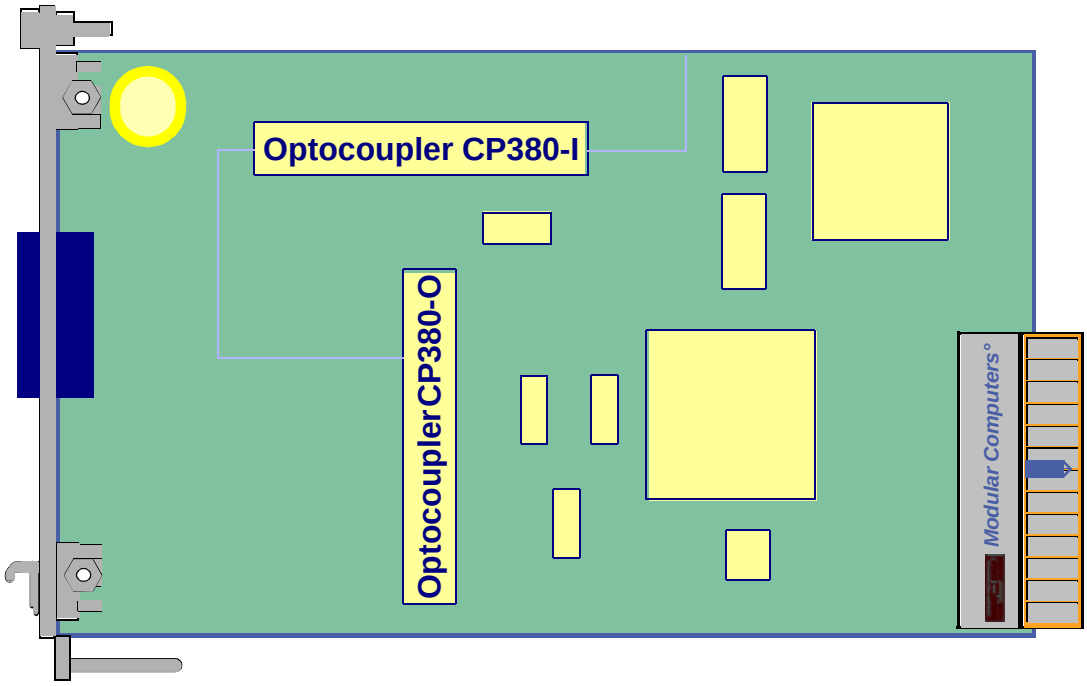
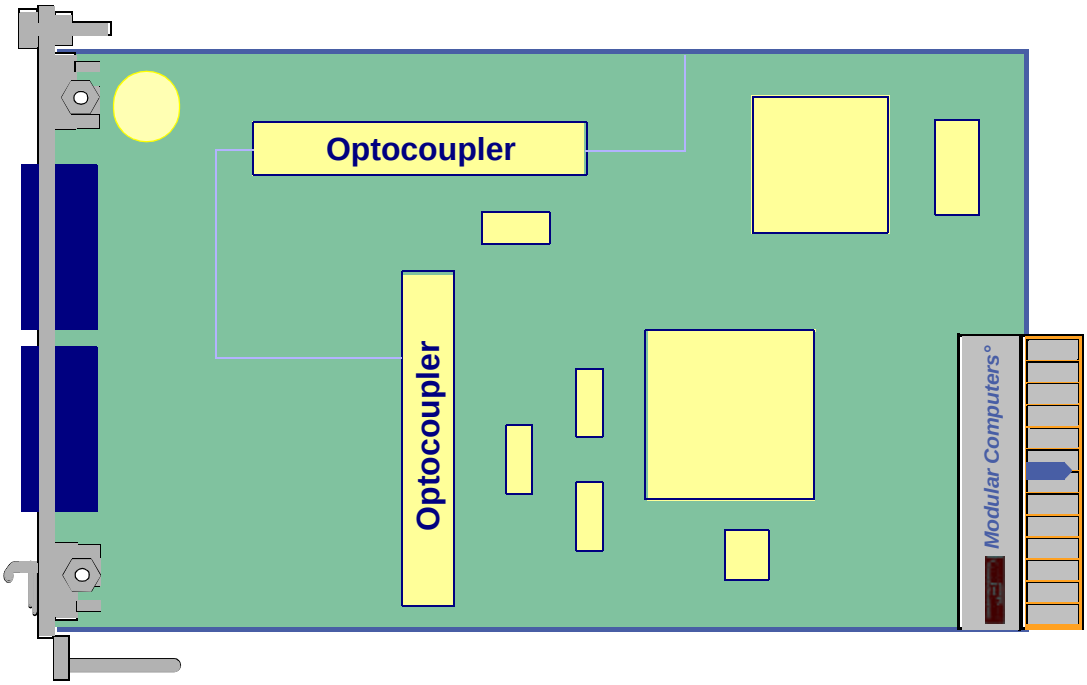


Figure 1-3: CP380-I/O Board Layout



1.6 Block Diagrams

Figure 1-4: CP380-I Functional Block Diagram

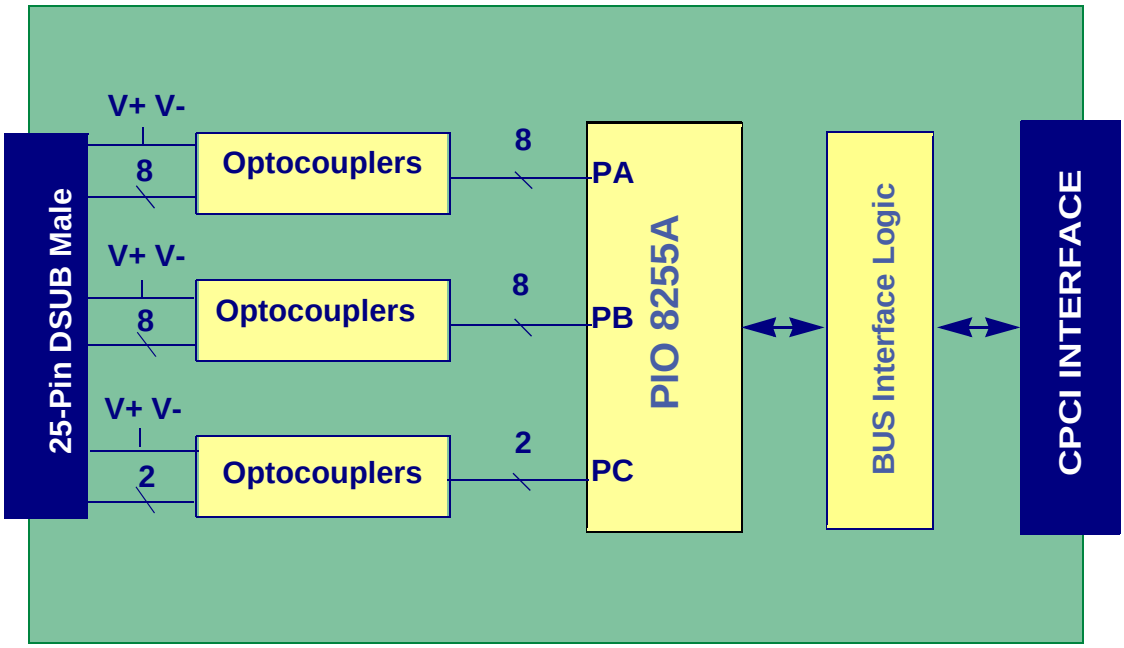


Figure 1-5: CP380-O Functional Block Diagram

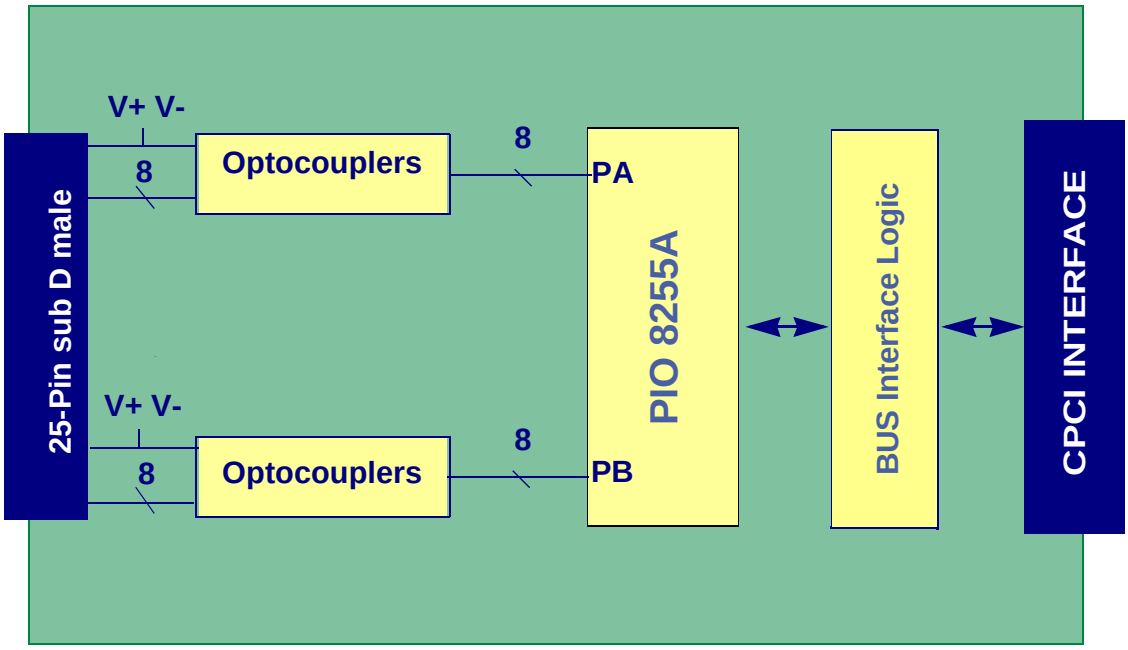
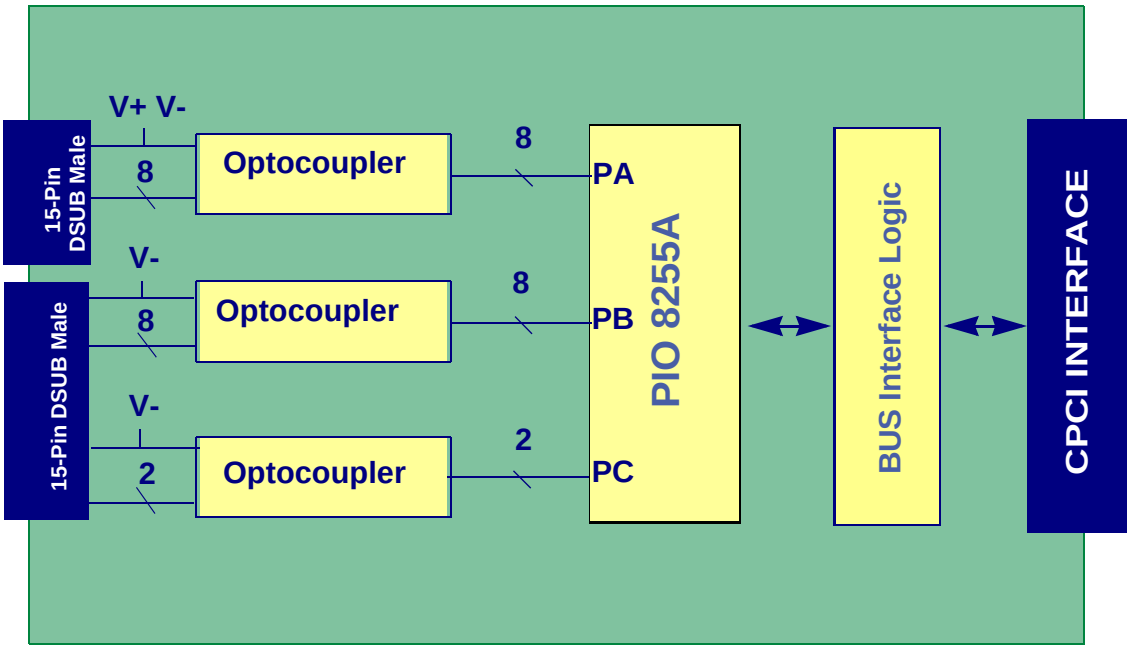


Figure 1-6: CP380-I/O Functional Block Diagram



1.7 General Notes

Important!:

The external connections to the board should only be made in a powered down state.

1.7.1 Modes of Operation

The CP380 uses an 8255 compatible PIO device. The operational modes of the device are described in the following chapters. For further information on the 8255 device consult the manufacturer's data sheet.

1.7.1.1 CP380-I

The CP 380-I uses 2 operational modes of the 8255 device which are configured by software.

Operation without Interrupt: 18 digital inputs

In this operation mode data input can be performed. The data is read in from the chosen port and no handshaking is required. By configuring the 8255 PIO device for mode 0 operation, two ports (port A and B, both 8 bits wide) and one port (PC2 and PC4, 2 bits wide) are available for data input.

Operation with Interrupt: 16 digital inputs and 2 interrupt channels

In this operation mode the 8255 PIO device is configured for mode 1. Two input groups (group A and group B) each containing an 8 bit data port and a 4 bit control/data port are available. The 8 bit ports are used for data input. The CP380-I uses only the PC4 (interrupt input for port A) and PC2 (interrupt input for port B) control port lines. All input ports are latched.

1.7.1.2 CP380-O

The CP380-O is implemented so that the ports A and B use the 8255 for simple data output (16 outputs). There is no interrupt control available on the CP380-O board.

1.7.1.3 CP380-I/O

The CP380-I/O operation is configured by software. Two operational modes of the 8255 device are used by the CP380-I/O.

Operation without Interrupt: 8 outputs and 10 inputs

In this operation mode, port A is used for data output and ports B and C are used for data input. Data is written to, or read from, the chosen port, and no handshaking is required. By configuring the 8255 for mode 0, one output port (port A: 8 bit wide) and two input ports (port B, 8 bit wide; port C, 2 bit wide) are available.

Operation with Interrupt: 8 outputs, 8 inputs and 1 interrupt channel

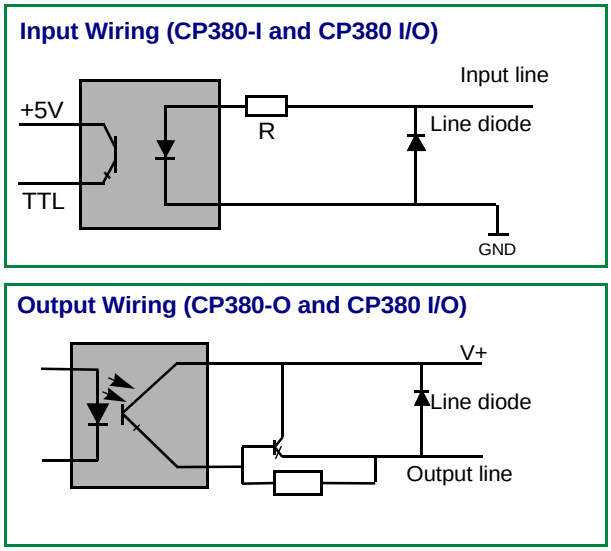
In this operation mode, port A of the 8255 is configured for mode 0 and port B is configured for mode 1. Port A (8 bit output) and port B (8 bit input) are used as data ports. The PC2 line of port C is used as the interrupt control line for port B. The PC4 input is not used.

1.7.2 Wiring

The inputs and outputs of the various boards are electrically isolated by optocouplers. There is no electrical connection between the hardware connected to the board and the PC.

1.8 Optocouplers

Figure 1-7: Output/Input Wiring of the CP380 Optocouplers



1.8.1 External Power for the Optocouplers

In order to run the boards properly, an external power supply is required. The power input pins on the D-sub connector are labelled “ext.V+“ and “ext.GND“ (see Chapter 3, Section 3.2 on page 3-3). The inputs and outputs of the CP380-I, CP380-O and CP380-I/O can be powered typically with 24V signals. The interrupt inputs of the CP380-I and CP380-I/O trigger on a falling edge. The external supply voltage can be anywhere from 20V to 28V so small variations in the voltage will have no effect on the board.

1.9 Specifications

Table 1-2: CP380 Specifications

	CP380	Specifications
PC Interface	**Type**	Compact-PCI
	Addressing	Automatic assignment of base address under Windows NT
	Interrupts	Automatic assignment of interrupt channel under Windows NT
Digital Inputs (CP380-I and CP380-I/O)	**Number**	CP380-I: 18;
		CP380-I/O: 10
	Type	8255 (PIO)
	Input Voltage	Low: 0…12 V
		High: 13…24 V
	Maximum Input Voltage	28V
	Input Current	7 mA at nominal input voltage
	Electrical Isolation	500 V DC (against PC-GND)
	Transfer Rate	Max. 1 kHz (effective suppression of voltage peaks in µs range)
	Handshake	Negative edge triggered signal
Digital Outputs (CP380-O and CP380-I/O)	**Number**	CP380-O: 16;
		CP380-I/O: 8;
	Defined Level at Power On/Off	Low
	Type	8255 (PIO)
	Output Voltage	Tristate output; min high level = Input Voltage - 0.1 volt
	Maximum Output Voltage	28V
	Output Current	Max. 220 mA per output channel Total max. output 1.5 A (Fuse F1 = 1.6 A, slow blow, 0.1 A margin)
	Electrical Isolation	500 V DC (against PC-GND)
	Transfer Rate	Max. 1 kHz (effective suppression of voltage peaks in µs range)

Table 1-2: CP380 Specifications

	CP380	Specifications
General Data	**Physical Size**	100 x 160 mm
	Weight	160g
	Connectors	25pin D-Sub male connector (CP380-I and CP380-O);
		2 x 15pin D-Sub male Connector (CP380-I/O)
	Temperature	Operation: 0 ... 60ºC (standard) E1: -25 ... 75ºC (extended) Storage: -40 ... 85ºC
	Humidity	0…90%, non-condensing
	Ext. Power Supply	24 V nominal; Fuse for external power: 1,6 AT/250 V
	Power Consumption **(of board only, does not include I/O channels)**	CP380-I: max. 250mA/5V
		CP380-O: max. 400mA/5V
		CP380-I/O: max. 380mA/5V

1.10 Applied Standards

1.10.1 CE Compliance

The *PEP Modular Computers'* CompactPCI systems comply with the requirements of the following CE-relevant standards:

- Emission EN50081-1
- Immission EN50082-2
- Electrical Safety EN60950

1.10.2 Mechanical Compliance

- Mechanical Dimensions IEEE 1101.10

1.10.3 Environmental Tests

- Vibration/Broad-Band Random Vibration IEC68-2-6 IEC68-2-64 (3U boards)
- Permanent Shock IEC68-2-29
- Single Shock IEC68-2-27

1.11 Related Publications

1.11.1 CompactPCI Systems/Boards

CompactPCI Specification, V. 2.0, Rev. 2.1

1.11.2 PMC Add-On Modules/Carriers

- Draft Standard for a Common Mezzanine Card Family, P1386/Draft 2.0
- Draft Standard Physical and Environment Layers for PCI Mezzanine Cards, P1386.1/Draft 2.0

This page was intentionally left blank.

Chapter 2

Installation

2.1 Hardware Installation ..... 2 - 3
2.2 Driver Installation ..... 2 - 3
2.2.1 Windows NT 4.0 ..... 2 - 3
2.2.2 Updating the System Driver ..... 2 - 4
2.2.3 Changing the Board Settings ..... 2 - 4
2.3 Uninstall ..... 2 - 4
2.3.1 Uninstall a Single Board ..... 2 - 4
2.3.2 Uninstall the Driver System ..... 2 - 5
2.4 Test Program ..... 2 - 5
2.5 Software ..... 2 - 6

This page was intentionally left blank.

2. Installation

If several boards are to be installed, please proceed board by board. After installing each board and registering the board with the operating system (Running SETUP.EXE), run a complete system check before installing the next board. The test program ME9x.EXE can be used for this purpose. Once the test has been successfully completed, proceed with the installation of the next board in an identical manner.

2.1 Hardware Installation

The CP380 family are Plug & Play compatible under Windows NT and will be configured by software. There are no settings required before installing the board into the computer. Make sure the computer is turned off before installing the board.

2.2 Driver Installation

2.2.1 Windows NT 4.0

- Please insert the disk “Windows NT Driver Disk for CP380“.
- From the start menu of Windows, choose Run.... Under the option Open:, enter the path for the SETUP.EXE file on the installation disk provided or click Browse.... Confirm the selection and follow the instructions of the setup program:
- The installation program will start.
- In the window "Install Options", choose "Install a new board" and confirm the selection with OK.
- Follow the installation program instructions and make sure you have your serial number ready.
- The following files will be installed:
  - The kernel driver ME9x_32.SYS into the path <*Windows directory*>\SYSTEM32\DRIVERS
  - The API-DLL ME9x_32.DLL into the path <*Windows directory*>\SYSTEM32
  - Dialogue DLL MEDLG32.DLL into the path <*Windows directory*>\SYSTEM32
  - Files for programming in Visual C++ into the path <*PEP working directory*>\ME-9x
  - Files for programming in Delphi into the path <*PEP working directory*>\ME-9x
  - Files for programming in Visual Basic into the path <*PEP working directory*>\ME-9x
  - Example programs for high level language programming into the path <*PEP working directory*>\ME-9x\DEMOS
  - Test program ME9x.EXE into the path <*PEP working directory*>\ME-9x\TEST
  - Registry entries will be made
- Reboot your computer. The system driver will automatically be loaded

2.2.2 Updating the System Driver

To update the system driver, follow the procedure described below for any board of the CP380 family.

- Please insert the disk “Windows NT Driver Disk for CP380“.
- From the start menu, choose the option Run.... Under the option Open:, enter the path for the SETUP.EXE file on the installation disk provided 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 driver and language libraries“ and confirm the selection with OK
- System drivers, API-DLL as well as high level language libraries, demo programs, and test programs will all be updated.
- Reboot your computer.

2.2.3 Changing the Board Settings

You can correct a false entry of the serial number by the following method.

- Please insert the disk “Windows NT Driver Disk for CP380“
- From the start menu, choose the option Run.... Under the option Open:, enter the path for the SETUP.EXE file on the installation disk provided 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 confirm the selection with OK
- Follow the setup program instructions
- Reboot your computer

2.3 Uninstall

2.3.1 Uninstall a Single Board

With the help of the install and uninstall program included with the disk "Windows NT Driver Disk for CP380", you can remove individual boards from the Windows registry. The individual components such as the system driver, API-DLL, and the high level language libraries are not removed by using this method.

- Please insert the disk “Windows NT Driver Disk for CP380“
- From the start menu, choose the option Run.... Under the option Open:, enter the path for the SETUP.EXE file on the installation disk provided 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 confirm the selection with OK
- Choose the board you want to remove from the Windows registry

- Reboot your computer
- The selected board will be removed from the Windows registry

2.3.2 Uninstall the Driver System

Using the following procedure, the entire CP380 Driver System will be removed from your computer. All files will be removed, including the system driver, API-DLL, high level language libraries, demo, and test programs (for **all** installed boards of the board family CP380), which are installed in the "ME-9x" subdirectory of the "`C:\PEP`" directory (default path).

- From the Windows Start menu, under the path `Settings Control Panel Add/ Remove Programs` in the property page `Install/Uninstall`, choose the “ME-9x Driver Uninstall“ option and confirm the selection with "OK".

This will remove the entire CP380 Driver System from your computer

2.4 Test Program

A simple test program is delivered with the board. This can be used to test all the basic functions of the CP380-I, CP380-O and CP380-I/O. The test program (`ME9x.EXE`) is automatically installed in the directory `C:\PEP\ME-9x\Test` (Default) and can be started with a double click (assuming the system driver was correctly installed).

2.5 Software

Table 2-1: Software

Windows NT 4.0 Platform	
SOFTWARE TYPE	**PRODUCT**
Driver(s)	CP380 system driver **(included)**
High Level Language Support (32-bit)	High Level Language Libraries for: Visual C++ V 4.0 or higher Delphi V 2.0 or higher Visual Basic V 4.0 or higher **(included)**
Graphical Programming Tools	HP VEE Driver System for HP VEE V 4.0 or higher **(included)**
	CP380 Driver System for LabVIEW™ V 4.0 or higher **(optional)**
Test and Demo Software	CP380 Demo and Test Program **(included)**

Table 2-1 provides an overview of the software which is available from *PEP Modular Computers GmbH* for the CP380 board. Included means that the software is delivered with the board. Optional means that the software must ordered separately from *PEP Modular Computers GmbH.*

Chapter 3

Configuration

3.1 Jumper Settings ..... 3 - 3
3.2 Pinouts ..... 3 - 3

This page was intentionally left blank.

3. Configuration

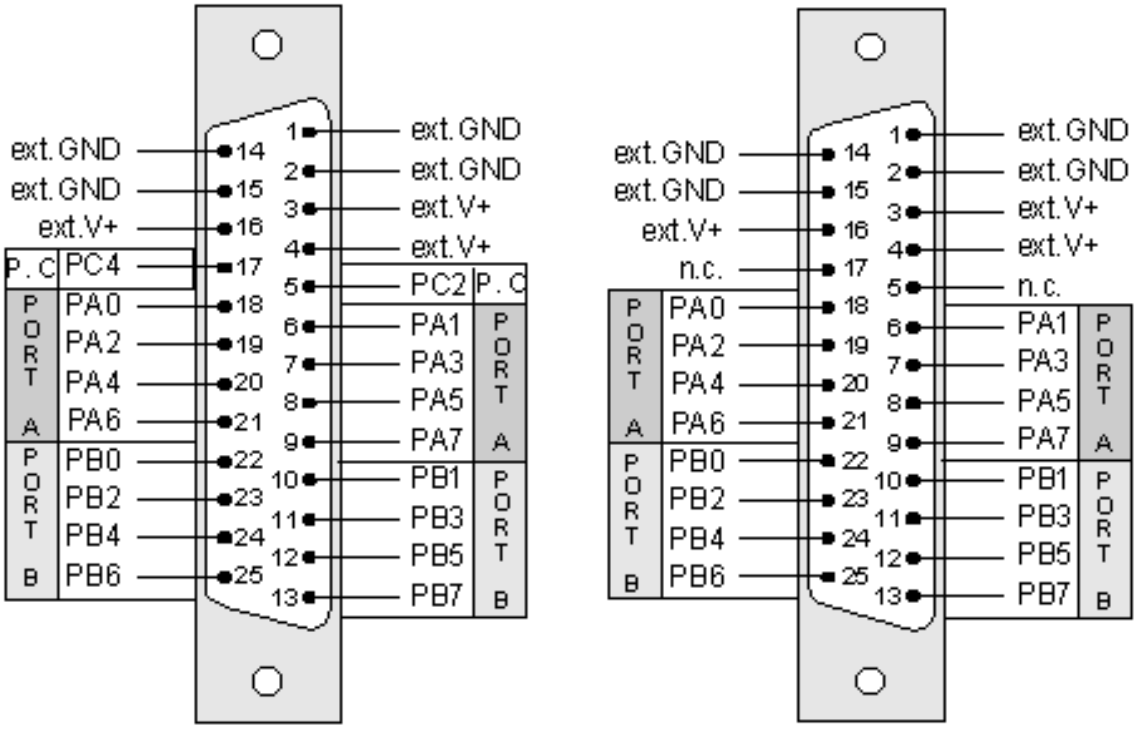
3.1 Jumper Settings

The boards of the CP380 family are Plug and Play compatible under Windows NT and are configured by software. There are no settings required before installing the board into the computer.

3.2 Pinouts

The CP380-I front panel is provided with a 25-Pin DSUB male input connector, the CP380-O with a 25-Pin DSUB male output connector.

Figure 3-1: Pinouts of the CP380-I Input Connector (Left) and the CP380-O Output Connector (Right)



The CP380-I/O front panel is provided with two connectors, an upper and a lower 15-pin DSUB male connector.

Figure 3-2: Pinout of the CP380-I/O Upper Connector (Output Connector)

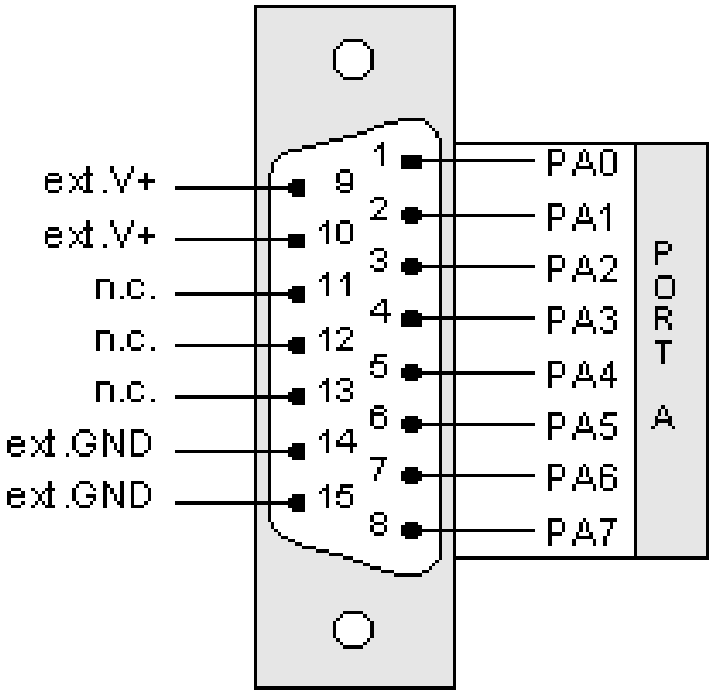
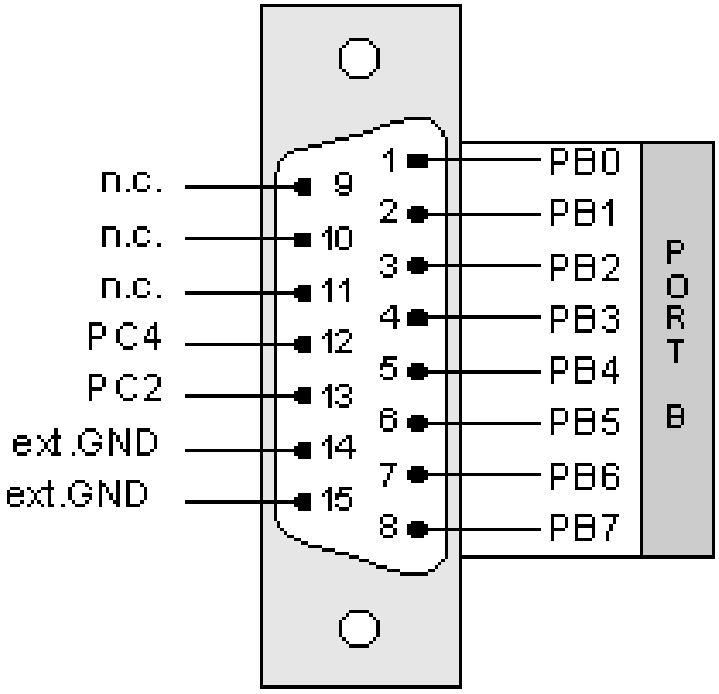


Figure 3-3: Pinout of the CP380-I/O Lower Connector (Input Connector)



Chapter 4

Programming

4.1 High Level Language Programming ..... 4 - 3
4.1.1 C Programming ..... 4 - 3
4.1.2 Delphi Programming ..... 4 - 3
4.1.3 Basic Programming ..... 4 - 3
4.1.4 Example Programs ..... 4 - 3
4.2 HP VEE Programming ..... 4 - 4
4.2.1 The ”ME Board” Menu ..... 4 - 4
4.2.2 HP VEE Example Programs ..... 4 - 4
4.2.3 User Objects ..... 4 - 4
4.3 LabVIEW™ Programming ..... 4 - 5
4.3.1 LabVIEW™ Example Programs ..... 4 - 5
4.3.2 Virtual Instruments ..... 4 - 5

This page was intentionally left blank.

4. Programming

4.1 High Level Language Programming

For high level language programming, the following files are required:

- ME9x_32.DLL (CP380 family system driver DLL in the Windows system directory)
- MEDLG32.DLL (Dialogue DLL in the Windows system directory)
- ME9x.H, ME9x.PAS or ME9x.BAS (CP380 family header/include files matching programming environment)
- MEDEFS.H, MEDEFS.PAS or MEDEFS.BAS (CP380 family macro definitions matching the programming environment)
- ME9x.32.LIB, ME9x.BC.LIB (CP380 library file for Visual C++/Borland C)

Make sure that the paths required to find the above files are set correctly in the compiler and linker.

4.1.1 C Programming

The header files (ME9x.H), including the definitions of the API function calls, have to be linked into your project for Visual C++, version 4.0 or later. If macro definitions are used, the header file MEDEFS.H must also be linked into the project. Please be sure to read the notes in these files.

4.1.2 Delphi Programming

The TPU source files (ME9x.PAS), including the definitions of the API function calls, have to be linked into your project for Delphi version 2.0 or later. If macro definitions are used, the macro definition file (MEDEFS.PAS) must also be linked into the project. Please be sure to read the notes in these files.

4.1.3 Basic Programming

The include files (ME9x.BAS), including the definitions of the API function calls, have to be linked into your project for Visual Basic version 4.0 or later. If macro definitions are used, the macro definition file (MEDEFS.BAS) must also be linked into the project. Please be sure to read the notes in these files.

Important Note:

Interrupt service functions (e. g. _me94EnableInt) are not supported under Visual Basic.

4.1.4 Example Programs

To facilitate understanding of programming the boards, some example programs are provided with source code. These example programs are automatically installed in the

directory `C:\PEP\ME-9x\Demos` (Default). The source code can be easily copied into your project by means of the clipboard.

4.2 HP VEE Programming

The HP VEE components for the CP380 family are delivered on (a) separate disk(s). The CP380 versions are supported under HP VEE. The full versions (4.0 or later) under Windows95/98/NT. For installation of HP VEE components and for further information please see the README file on the disks delivered with the HP VEE driver system. For basics of HP VEE programming use your HP VEE documentation and the HP VEE online help index.

4.2.1 The ”ME Board” Menu

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



Note:

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

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

If necessary, you can install the additional HP VEE components at any time. To do this use the disk “HP VEE Driver Disk for CP380“.

4.2.2 HP VEE Example Programs

For demonstration purposes and to assist understanding, demo programs using the important UOs have been written. They can be accessed via 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…" is used in their file name .

4.2.3 User Objects

For convenient use of the driver, predefined „User Objects“ (UOs) have been developed which internally call API functions. They can be accessed via 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 way as standard HP VEE objects.

The User Objects are self descriptive and based on the API functions documented in the chapter “Function Reference“. A short description of every UO is also available under the item “Description“ which can be summoned by moving the cursor over the UO and pressing the right mouse button.

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

4.3 LabVIEW™ Programming

LabVIEW™ components for the CP380 are delivered optionally on (a) separate disk(s). The CP380 is supported by LabVIEW™ full version 4.x or higher under Windows95/98/NT. For installation of LabVIEW™ driver components and for further information please see the README file on the disks delivered with the CP380 LabVIEW™ driver. For basics of LabVIEW™ programming please use your LabVIEW™ documentation and the LabVIEW™ online help index.

4.3.1 LabVIEW™ Example Programs

For demonstration purposes and to assist understanding, demo programs using the important Virtual Instruments (VIs) have been written. They can be accessed via the menu item “File - Open“.

4.3.2 Virtual Instruments

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

The source VIs are self descriptive and based on the API functions documented in the chapter “Function Reference“. A short description of every VI is also available in the VI `ME-9x Function Tree`. This VI is for documentation purposes only and can be opened by the menu “File - Open“. Under “Description“ you will find a brief description of every Virtual Instrument.



Note!

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

This page was intentionally left blank.

Chapter 5

Function Reference

5.1 Functional Overview of the 32 Bit Driver ..... 5 - 3
5.2 Naming Conventions ..... 5 - 3
5.3 Definitions ..... 5 - 3
5.4 Description of the API Functions ..... 5 - 4
5.4.1 General Functions ..... 5 - 4
5.4.2 Digital Input ..... 5 - 6
5.4.3 Digital Output ..... 5 - 13
5.4.4 Interrupt Handling ..... 5 - 20
5.4.5 Error Handling ..... 5 - 25

This page was intentionally left blank.

5. Function Reference

5.1 Functional Overview of the 32 Bit Driver

The 32-bit driver for the boards of the CP380 family was developed for Windows NT 4.0 and consists of the following components:

- Kernel driver (`ME9x_32.SYS`) for Windows NT, loaded automatically on power up.
- API-DLL (`ME9x_32.DLL`) with the driver functions for the CP380 family.
- Dialogue DLL (`MEDLG32.DLL`) with dialogue functions.

The installation program for the CP380 family registers every new board with the operating system. Upon every startup, the driver searches for a registered CompactPCI board and checks that it physically exists. 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 a CP380 board requires an integer value for identification of the board. In the following description of the functions this parameter is referred to as `iBoardNumber`.

5.2 Naming Conventions

These functions were written specifically for the CP380 board family. For Visual C and Delphi (Pascal) every API function starts with an underscore "_" (this is 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. "DI" for "Digital In").

`_me94...`	Function valid for board CP380-I
`_me95...`	Function valid for board CP380-O
`_me96...`	Function valid for board CP380-I/O
`_me9x...`	Function valid for the boards CP380-I, CP380-O, CP380-I/O

5.3 Definitions

All pre-defined constants and macros for Visual C++ will be found in the file `MEDEFS.H`, for Delphi in the file `MEDEFS.PAS` and for Visual Basic in the file `MEDEFS.BAS`. By linking the appropriate definition file into your project you can pass many parameters in the form of predefined constants (e. g. `MEINPUT, AO_PLUS_10`). As an alternative, you can pass the matching Hex value at any time.

5.4 Description of the API Functions

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

- General Functions
- Digital Input
- Digital Output
- Interrupt Handling
- Error Handling

5.4.1 General Functions

5.4.1.1 _me9xGetBoardVersion()

Syntax (C)

```
int _me9xGetBoardVersion (
                            int iBoardNumber,
                            int *piDevices,
                            )
```

Syntax (Delphi)

```
Function
_me9xGetBoardVersion (
                            iBoardNumber: integer;
                            Var iDevices: integer
                            ): integer;
```

Syntax (Basic)

```
Declare Function
 me9xGetBoardVersion Lib "me9x_32" Alias "_VBme9xGetBoardVersion@8" (
                            ByVal iBoardNumber As Long,
                            ByRef iDevices As Long,
                            ) As Long
```

Description

Determines the board version number of an installed board of the board family CP380.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<Version>	Pointer to an integer value returning the board version. Possible values: • 94Hex: for CP380-I • 95Hex: for CP380-O • 96Hex: for CP380-I/O

Return Values

1	Function executed successfully
0	Error

The cause of the error can be determined with the function _me9xGetDrvErrMess.

5.4.1.2 _me9xGetDLLVersion

Syntax (C)

```
int _me9xGetDLLVersion();
```

Syntax (Delphi)

```
Function
 _me9xGetDLLVersion: integer;
```

Syntax (Basic)

```
Declare Function
 me9xGetDLLVersion Lib "me9x_32" Alias "_VBme9xGetDLLVersion@0" () As Long
```

Description

Returns the version number of the board DLL for the board family CP380.

Parameters

None

Returns

Version number. This 32 bit value has the main version in the upper 16 bits and the sub version in the lower 16 bits.

For example a return value: 0x00010003 for the version 1.03

5.4.2 Digital Input

5.4.2.1 _me94DIGetBit

Syntax (C)

```
int _me94DIGetBit (
                            int iBoardNumber,
                            int iBitNo,
                            int *piBitValue
                            );
```

Syntax (Delphi)

```
Function
 _me94DIGetBit (
                        iBoardNumber: integer;
                        iBitNo: integer;
                        Var iBitValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me94DIGetBit Lib "me9x_32" Alias "_VBme94DIGetBit@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iBitNo As Long,
                        ByRef iBitValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I.

This function determines the status of a single input line.

Parameters

`<BoardNumber>`	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
`<BitNo>`	Input line (number); possible values: 0…7: for port A 8...15: for port B 16, 17: for bits C2 and C4 of port C
<BitValue>	Pointer to an integer value which corresponds with the line status: Possible return values: 0: Line is set to '0' 1: Line is set to '1'

Returns

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 `_me9xGetDrvErrMess`.

5.4.2.2 _me94DIGetByte

Syntax (C)

```
int _me94DIGetByte (
                        int iBoardNumber,
                        int iPortNo,
                        int *piValue
                        );
```

Syntax (Delphi)

```
Function
 _me94DIGetByte (
                        iBoardNumber: integer;
                        iPortNo: integer;
                        Var iValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me94DIGetByte Lib "me9x_32" Alias "_VBme94DIGetByte@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iPortNo As Long,
                        ByRef iValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I.

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

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15	
<PortNo>	Port name; possible values:	
	<PortNo>	Description
	PORTA (00hex)	Input port A
	PORTB (01hex)	Input port B
	PORTC (02hex)	(only bits C2 and C4 significant; the other bits of port C are not available)
<Value>	Pointer to an integer value which contains the input byte; for ports A and B only the lower 8 bits are significant; for port C the bits C2 and C4 are significant.	

Returns

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 _me9xGetDrvErrMess.

5.4.2.3 _me96DIGetBit

Syntax (C)

```
int _me96DIGetBit (
                        int iBoardNumber,
                        int iBitNo,
                        int *piBitValue
                        );
```

Syntax (Delphi)

```
Function
 _me96DIGetBit (
                        iBoardNumber: integer;
                        iBitNo: integer;
                        Var iBitValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me96DIGetBit Lib "me9x_32" Alias "_VBme96DIGetBit@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iBitNo As Long,
                        ByRef iBitValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I/O.

This function determines the status of a single input line.

Parameters

`<BoardNumber>`	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
`<BitNo>`	Input line (number); possible values: 0…7: for port B 8, 9: for bits C2 and C4 of ports C
`<BitValue>`	Pointer to an integer value which corresponds with the line status: Possible return values: 0: Line is set to '0' 1: Line is set to '1'

Returns

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 `_me9xGetDrvErrMess`.

5.4.2.4 _me96DIGetByte

Syntax (C)

```
int _me96DIGetByte (
                        int iBoardNumber,
                        int iPortNo,
                        int *piValue
                        );
```

Syntax (Delphi)

```
Function
 _me96DIGetByte (
                        iBoardNumber: integer;
                        iPortNo: integer;
                        Var iValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me96DIGetByte Lib "me9x_32" Alias "_VBme96DIGetByte@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iPortNo As Long,
                        ByRef iValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I/O.

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

Parameters

BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<PortNo>	Port name; possible values: <PortNo>Description PORTB (01hex)Input port B PORTC (02hex)only the bits C2 and C4 are significant; the other bits of port C are not available
<Value>	Pointer to an integer value which contains the input byte; only the lower 8 bits are significant; for port C only the bits C2 and C4 are significant.

Returns

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 _me9xGetDrvErrMess.

5.4.3 Digital Output

5.4.3.1 _me95DOSetBit

Syntax (C)

```
int _me95DOSetBit (
                        int iBoardNumber,
                        int iBitNo,
                        int iBitValue
                        );
```

Syntax (Delphi)

```
Function
 _me95DOSetBit (
                        iBoardNumber: integer;
                        iBitNo:integer;
                        iBitValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
me95DOSetBit Lib "me9x_32" Alias "_VBme95DOSetBit@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iBitNo As Long,
                        ByVal iBitValue As Long
                        ) As Long
```

Description

This function is only for the CP380-O.

Sets a single digital output line to 0 or 1.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<BitNo>	Output line (number) to be set; possible values: 0…7 (for port A) 8…15 (for port B)
<BitValue>	Bit value for output line to be set; possible values: 0: bit set to '0' 1: bit set to '1'

Returns

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 _me9xGetDrvErrMess.

5.4.3.2 _me95DOSetByte

Syntax (C)

```
int _me95DOSetByte (
                        int iBoardNumber,
                        int iPortNo,
                        int iValue
                        );
```

Syntax (Delphi)

```
Function
_me95DOSetByte (
                        iBoardNumber: integer;
                        iPortNo: integer;
                        iValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me95DOSetByte Lib "me9x_32" Alias "_VBme95DOSetByte@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iPortNo As Long,
                        ByVal iValue As Long
                        ) As Long
```

Description

This function is only for the CP380-O.

Writes a byte to a digital output port.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<PortNo>	Port name; possible values: <PortNo>Description PORTA (00hex)Output port A PORTB (01hex)Output port B
<Value>	Output value, possible values: 00Hex…FFHex (decimal: 0…255);

Returns

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 `_me9xGetDrvErrMess`.

5.4.3.3 _me96DOSetBit

Syntax(C)

```
int _me96DOSetBit (
                        int iBoardNumber,
                        int iBitNo,
                        int iBitValue
                        );
```

Syntax (Delphi)

```
Function
_me96DOSetBit (
                        iBoardNumber: integer;
                        iBitNo: integer;
                        iBitValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me96DOSetBit Lib "me9x_32" Alias "_VBme96DOSetBit@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iBitNo As Long,
                        ByVal iBitValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I/O.

Sets a single digital output line to 0 or 1.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<BitNo>	Output line (number) to be set; possible values: 0…7 (for port A)
<BitValue>	Bit value for output line to be set; possible values: 0: bit set to '0' 1: bit set to '1'

Returns

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 _me9xGetDrvErrMess.

5.4.3.4 _me96DOSetByte

Syntax (C)

```
int _me96DOSetByte (
                        int iBoardNumber,
                        int iValue
                        );
```

Syntax (Delphi)

```
Function
 _me96DOSetByte (
                        iBoardNumber: integer;
                        iValue: integer
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me96DOSetByte Lib "me9x_32" Alias "_VBme96DOSetByte@12" (
                        ByVal iBoardNumber As Long,
                        ByVal iValue As Long
                        ) As Long
```

Description

This function is only for the CP380-I/O.

Writes a byte to a digital output port.

Parameters

`<BoardNumber>`	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
`<Value>`	Output value, possible values: 00Hex…FFHex (decimal: 0…255);

Returns

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 `_me9xGetDrvErrMess`.

5.4.4 Interrupt Handling

5.4.4.1 _me94DisableInt



Important Note!

This function can be used for high level languages!

Syntax (C)

```
int _me94DisableInt (
                        int iBoardNumber
                        );
```

Syntax (Delphi)

```
Function
 _me94DisableInt (
                        iBoardNumber: integer
                        ): integer;
```

Syntax (Basic)

```
Not implemented
```

Description

This function is only for the CP380-I.

This function disables the interrupt control. It stops an interrupt processing started by `_me94EnableInt`.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15

Returns

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 _me9xGetDrvErrMess.

5.4.4.2 _me94EnableInt

Syntax (C)

```
int _me94EnableInt (
                      int iBoardNumber,
                      pSERVICE_PROC IrqFunc
                      );
```

Syntax (Delphi)

```
Function
_me94EnableInt (
                      iBoardNumber: integer;
                      IrqFunc: Pointer
                      ): integer;
```

Syntax (Basic)

```
Not implemented
```

Description

This function is only for the CP380-I.

This function enables the interrupt control. If an interrupt occurs a user defined interrupt routine will be processed.



Important Note!

As interrupt source, you can use one of the both interrupt inputs PC2 or PC4 of the CP380-I. Make sure, that the channel an interrupt signal was switched on first remains the interrupt input.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<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.

Returns

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 _me9xGetDrvErrMess.

5.4.4.3 _me96DisableInt

Syntax (C)

```
int _me96DisableInt (
                        int iBoardNumber
                        );
```

Syntax (Delphi)

```
Function
 _me96DisableInt (
                        iBoardNumber: integer
                        ): integer;
```

Syntax (Basic)

```
Not implemented
```

Description

This function is only for the CP380-I/O.

This function disables the interrupt control. It stops an interrupt processing started by _me96EnableInt.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15

Returns

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 _me9xGetDrvErrMess.

5.4.4.4 _me96EnableInt

Syntax (C)

```
int _me96EnableInt (
                        int iBoardNumber,
                        pSERVICE_PROC IrqFunc
                        );
```

Syntax (Delphi)

```
Function
 _me96EnableInt (
                        iBoardNumber: integer;
                        IrqFunc: Pointer
                        ): integer;
```

Syntax (Basic)

```
Not implemented
```

Description

This function is only for the CP380-I/O.

This function enables the interrupt control. If an interrupt occurs an user defined interrupt routine will be processed.



Important Note!

As interrupt source, you can use only the interrupt input PC2 of the CP380-I/O.

Parameters

<BoardNumber>	Board number for 1., 2., 3...16 installed CP380-I, CP380-O or CP380-I/O; possible values: 0…15
<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.

Returns

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 `_me9xGetDrvErrMess`.

5.4.5 Error Handling

5.4.5.1 _me9xGetDrvErrMess



Important Note!

This function can only be called, if the previously called API function of the ME9x.DLL returned an error code (i. e. error code 0)!

Syntax (C)

```
int
_me9xGetDrvErrMess (
                        char *pcErrortext
                        );
```

Syntax (Delphi)

```
Function
 _me9xGetDrvErrMess (
                        Var errortext: errorstring
                        ): integer;
```

Syntax (Basic)

```
Declare Function
 me9xGetDrvErrMess Lib "me9x_32" Alias "_VBme9xGetDrvErrMess@4" (
                        ByVal errortext As String
                        ) As Long
```

Description

This function is for the CP380-I, CP380-O, CP380-I/O.

If an error occurs during the processing of one of the API functions of the CP380 driver, this routine returns the matching error code and text.

Parameters

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

Returns

0, if there was no error or matching error code.

This page was intentionally left blank.