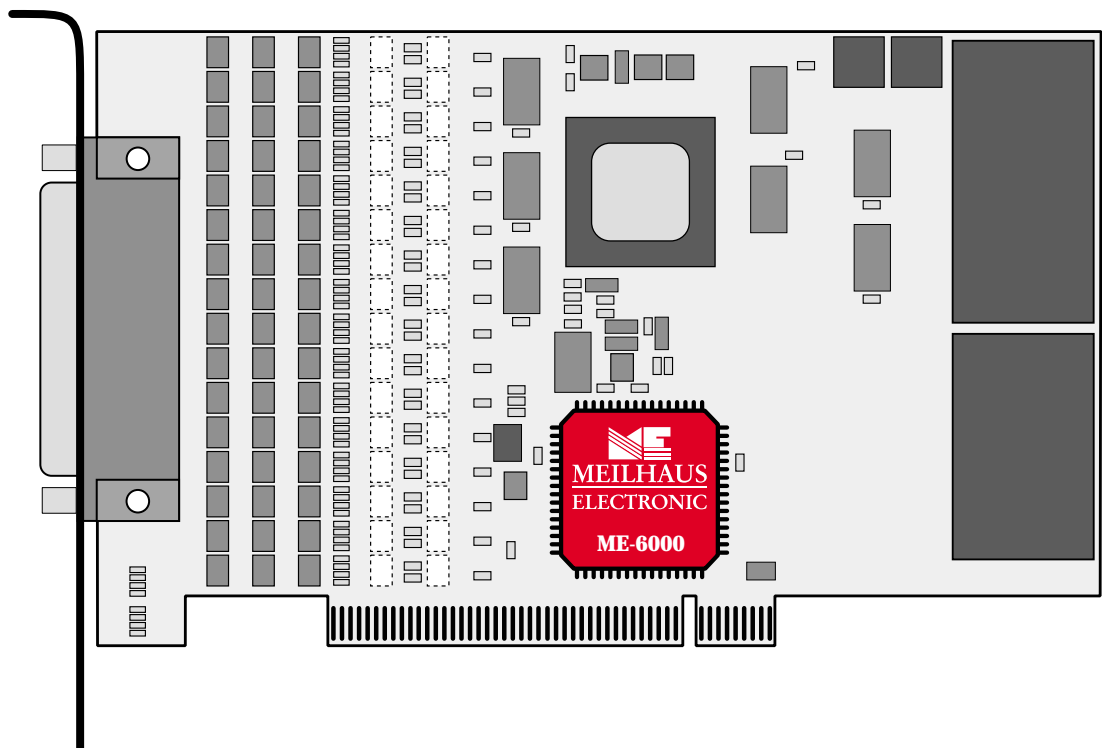


Meilhaus Electronic Manual

ME-6000/6100 1.11E for PCI Bus



**16 Bit D/A-Conversion Board with up to 16 Channels
and electrical Isolation; optional: „Island Channels“**

Imprint

Manual ME-6000/6100 PCI

Revision 1.11E

Revised: 1. June 2004

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

© Copyright 2004 Meilhaus Electronic GmbH

All rights reserved. No part of this publication may be reproduced or distributed in any form whether photocopied, printed, put on microfilm or be stored in any electronic media without the expressed written consent of Meilhaus Electronic GmbH.

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.

Borland Delphi is a trademark of Borland International Inc.

Turbo/Borland C is a trademark of Borland International Inc.

Visual C++ and Visual Basic are trademarks of the Microsoft Corporation.

VEE Pro and VEE OneLab are trademarks of Agilent Technologies.

ME-VEC and ME-FoXX are trademarks of Meilhaus Electronic.

Other company names and product names found in the text of this manual are also trademarks of the companies involved.



Table of Contents

1	Introduction.....	5
1.1	Package Contents	5
1.2	Features	6
1.3	System Requirements.....	8
1.4	Software Support.....	8
2	Installation.....	9
2.1	Test Program	9
3	Hardware	11
3.1	Block Diagram	11
3.2	General Notes	12
3.3	Hardware Description	13
3.3.1	External Trigger.....	13
3.3.2	Electrical Isolation	14
3.3.3	Island Channels	15
3.3.4	Option „High Current“	16
4	Programming.....	17
4.1	Operation Modes	17
4.1.1	Single Value Output.....	17
4.1.2	Continuous Mode	18
4.1.3	WrapAround Mode	21
4.2	High Level Language Programming.....	24
4.2.1	Example Programs	24
4.3	Agilent VEE Programming	24
4.3.1	User Objects	25
4.3.2	Example Programs	25
4.3.3	The "ME Board" Menu	25
4.4	LabVIEW™ Programming	26
4.4.1	Virtual Instruments.....	26
4.4.2	Example Programs	26
5	Function Reference	27
5.1	Overview	27
5.2	Naming Conventions	27
5.3	Description of the API Functions.....	29
5.3.1	General Functions.....	30
5.3.2	Analog Output.....	34
5.3.3	Error Handling.....	49

Appendix..... 51

- A Specifications 51**
- B Pinout 53**
 - B1 Pinout D-Sub Connector..... 53
- C Accessories..... 54**
- D Technical Questions..... 55**
 - D1 Hotline 55
 - D2 Service address 55
 - D3 Driver Update..... 55
- E Index 57**

1 Introduction

Valued customer,

Thank you for purchasing a Meilhaus PC plug-in board. You have chosen an innovative high technology product 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 us immediately.

Before you install the board in your computer, read this manual carefully, especially the chapter describing board installation.

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 content of the box. Your box should consist of:

- D/A conversion board of the ME-6000/6100 family for PCI-bus.
- Manual in PDF format on CD-ROM.
(optional as printed version)
- Driver software on CD-ROM.
- Special connector cable from 78pin D-Sub male connector to 16 single shielded lines with open end.

1.2 Features

Model Overview

Model	Short Description
ME-6000/4 PCI	4 D/A channels with electrical isolation; 16 bit, up to 500kHz per channel
ME-6000/8 PCI	8 D/A channels with electrical isolation; 16 bit, up to 500kHz per channel
ME-6000/16 PCI	16 D/A channels with electrical isolation; 16 bit, up to 500kHz per channel
ME-6100/4 PCI	4 D/A channels with electrical isolation; all 4 channels controlled by timer; 16 bit, up to 500kHz per channel
ME-6100/8 PCI	8 D/A channels with electrical isolation from which 4 channels can be controlled by timer; 16 bit, up to 500kHz per channel;
ME-6100/16 PCI	16 D/A channels with electrical isolation from which 4 channels can be controlled by timer; 16 bit, up to 500kHz per channel
Optional for all models:	„Island Channels“ (all channels isolated from each another)

Table 1: Model overview ME-6000/6100 familiy

The boards of the **ME-6000 family** provide 4-, 8- or 16 D/A channels. Each channel is using its own high accuracy, high speed 16 bit D/A converter. The voltage outputs can be driven in the output range ± 10 V.

On the topmodel **ME-6100** 4 channels can be used for signal curve generation. Each of the 4 channels has its own 8 kByte FIFO for output values. Sample rates up to **500 kHz per channel** are possible. You can choose between the operation mode „Continuous“ for putting out values continuously and the operation mode „Wraparound“ for periodically signal curves. On the models ME-6100/8 and ME-6100/16 additionally 4 resp. 12 „normal“ D/A channels are available.

For the **ME-6100** a virtual instrument panel with a graphical user interface is provided. Simply use your board as a „**Virtual Function Generator**“ for output of periodically signals. .

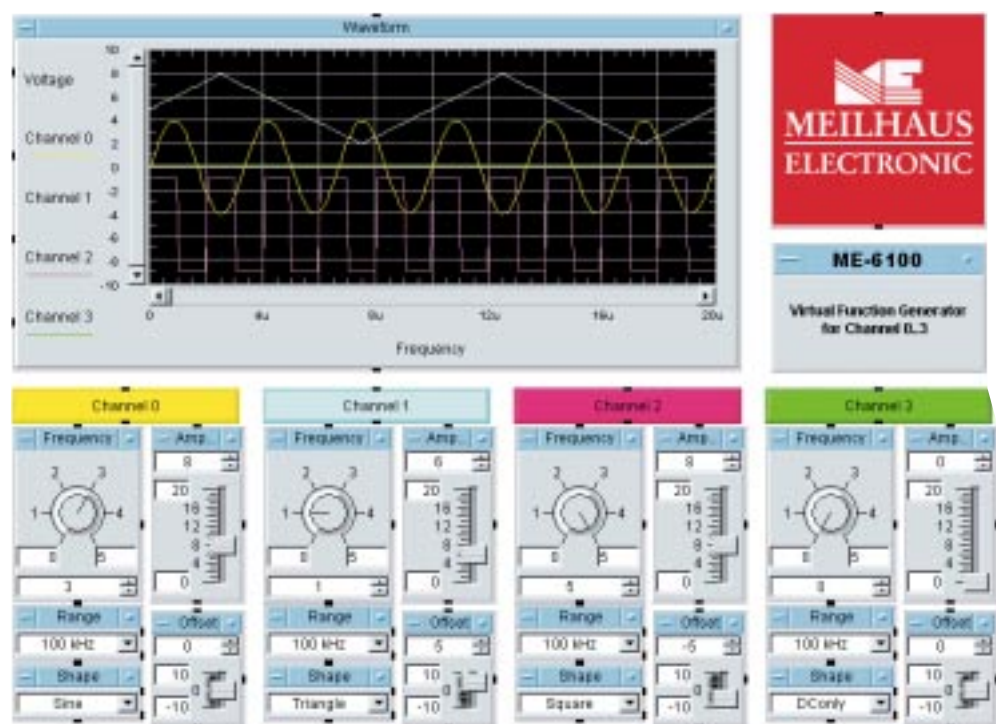


Diagram 1: Virtual Function Generator

The D/A section is **electrically isolated** as a whole from the rest of the board. As an option you can isolate the single channels from **each another (Island Channels)**. With that accuracies better $\pm 1\%$ are possible.

1.3 System Requirements

The ME-6000/6100 can be installed into any PC with Intel® Pentium® processor or compatible computers with a free standard PCI slot. The board is supported by all current Windows versions as well as Linux and VxWorks.

1.4 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.

2.1 Test Program

For simple testing of the board use the appropriate test programs provided with the ME Software Developer Kit (ME-SDK). After un-packing the ME-SDK the test programs can be found in corresponding subdirectories of C:\MEILHAUS (Default). Note that the system driver must be installed correctly.

3 Hardware

3.1 Block Diagram

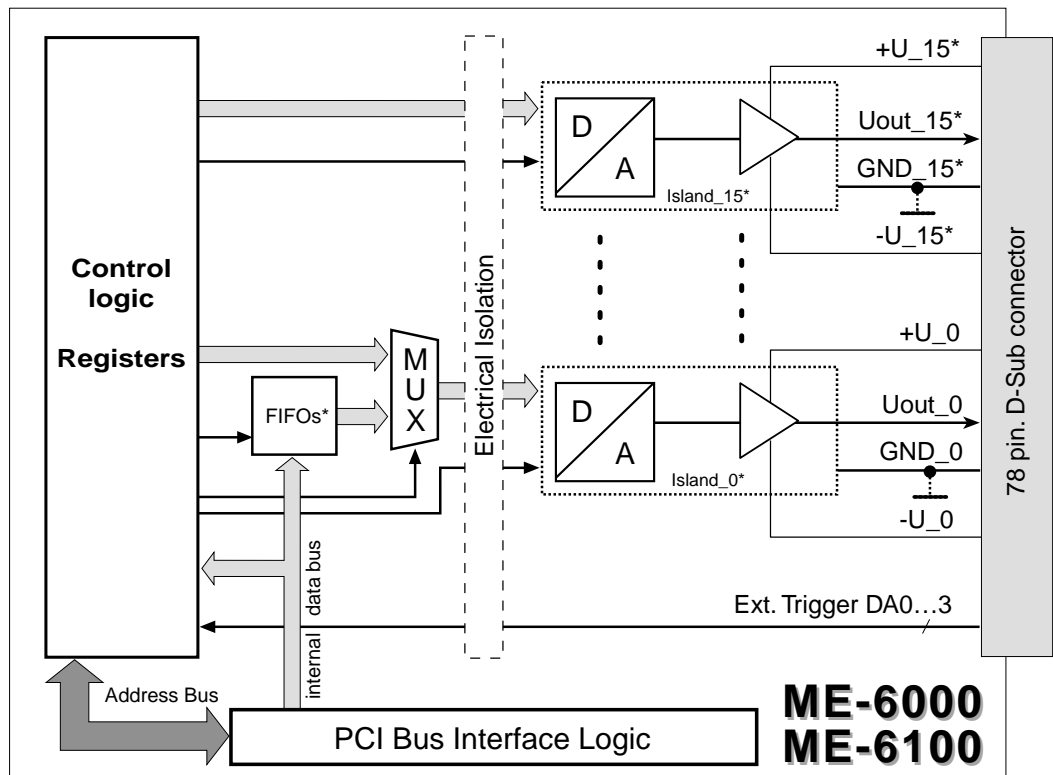


Diagram 2: Block diagram of ME-6000/6100

* Depending on the version not all functional groups available:

ME-6000/4: 4 voltage outputs (Uout_0...3).

ME-6000/8: 8 voltage outputs (Uout_0...7).

ME-6000/16: 16 voltage outputs (Uout_0...15).

ME-6100/4: 4 voltage outputs (Uout_0...3) with FIFO for signal curve generation

ME-6100/8: 8 voltage outputs (Uout_0...7), 4 of them with FIFO for signal curve generation (Uout_0...3)

ME-6100/16: 16 voltage outputs (Uout_0...15), 4 of them with FIFO for signal curve generation (Uout_0...3)

Optional: Electrical isolated channels from each another

3.2 General Notes

Attention:

After power up the D/A channels are putting out -10V. After starting the driver the output value changes to 0V. To guarantee a defined power up condition please start your host computer first and do not power up your external switching until the driver started.

Important Note!

Make sure that a reference from the external switching to PC ground is made. The pins +U_0...15 and -U_0...15 are only required, if the one of the options „Island Channels“ or „High Current“ (HC) will be used; else they must be **not connected**.

Before handling and installing the board and cable, make sure that you ground yourself. Static electricity can damage some of the more sensitive components on the board by static discharges.

The external connections to the board should only be made or removed in a powered down state. Look for proper connection of the cable. It must join the Sub-D jack completely and must be fixed with the both screws. Otherwise a perfect running of the board could not be guaranteed!

The pinout of the 78pin D-Sub connector is shown in Appendix B on page 53.

3.3 Hardware Description

The boards of the ME-6000 family are assembled with a 16 bit D/A converter per channel. The sample rate is up to 500 kHz. The output voltage serves a voltage range of ± 10 V.

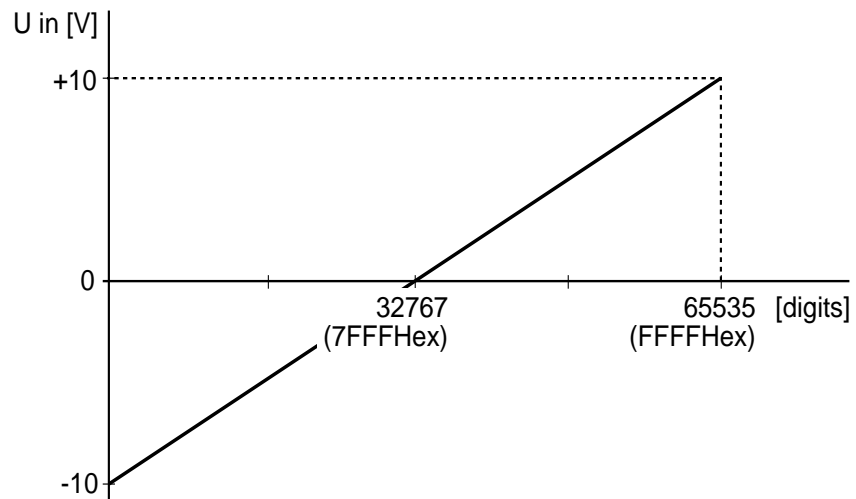


Diagram 3: Transfer characteristic of the D/A converter

3.3.1 External Trigger

On the ME-6100 the channels 0...3 can be started by an external trigger signal in the operation modes „Continuous“ and „Wrap-around“. You can choose between rising and falling edge by software. The external trigger signal must drive against ground (GND_x). Note that the trigger inputs have to be sourced by a current I_F of $7,5 \text{ mA} \leq I_F \leq 10 \text{ mA}$.

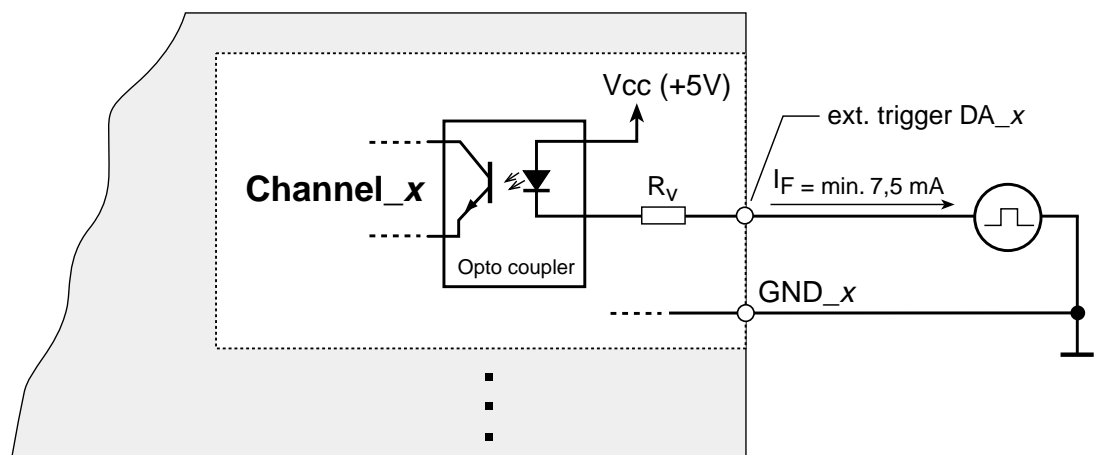


Abb. 4: Switching of the trigger inputs

3.3.2 Electrical Isolation

All D/A channels of the board are electrically isolated by optocouplers from PC-GND. I.e. GND_0...15 are connected with each other and have a common ground reference (ISO_GND). On the ME-6100 also the external trigger inputs are opto isolated.

The output current I_{\max} per channel depends of the number of assembled resp. used channels (see table below).

Channels	I_{\max}	Channels	I_{\max}
4	$\pm 15\text{mA}$	12	$\pm 10\text{mA}$
8	$\pm 15\text{mA}$	16	$\pm 3\text{mA}$

Table 2: Max. Ausgangsstrom

Note, the output current per channel may not exceed **$\pm 15\text{ mA}$** ! The pins for the external $\pm 15\text{V}$ power supply (+U_x and -U_x) are driven internally and may not be connected externally!

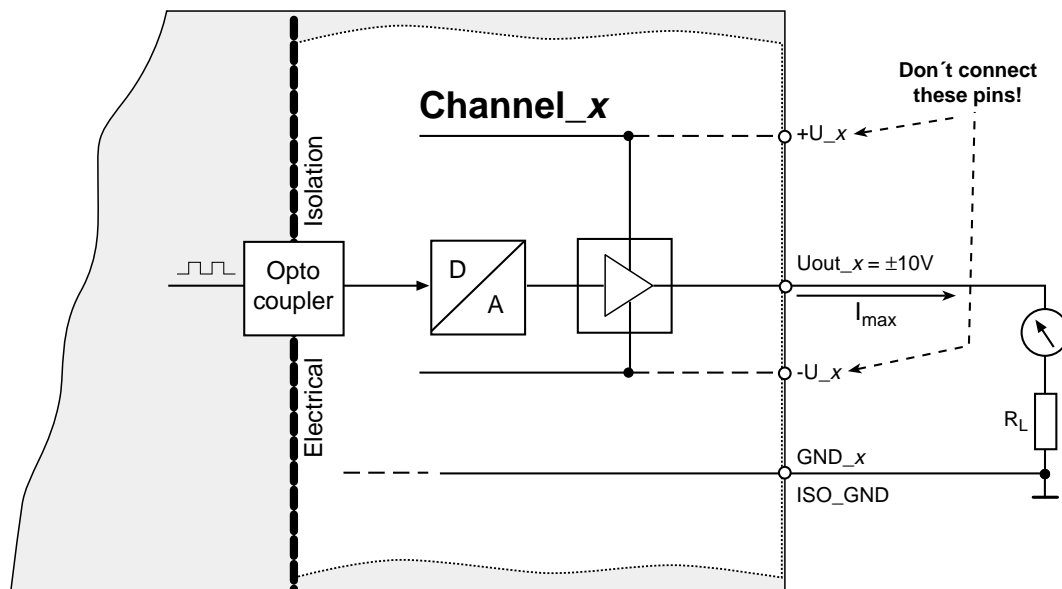


Diagram 5: Electrical isolation of all channels

3.3.3 Island Channels

With the option „Island Channels“ all the D/A channels use independent ground levels and supply pins. I.e. you have to connect the ground reference of each channel (GND_0...15) with the appropriate GND of your external application. Further on every island channel requires an independent, symmetrical power supply of $\pm 15\text{ V}$ ($\pm 22\text{ mA}$ per channel for $I_{\text{max}} = \pm 15\text{ mA}$). If you use a high end, low noise power supply, you can realize excellent accuracies better $\pm 1\%$. On the ME-6100 the external trigger inputs are also included with the „islands“. Maximum $\pm 15\text{ mA}$ can be driven per D/A channel.

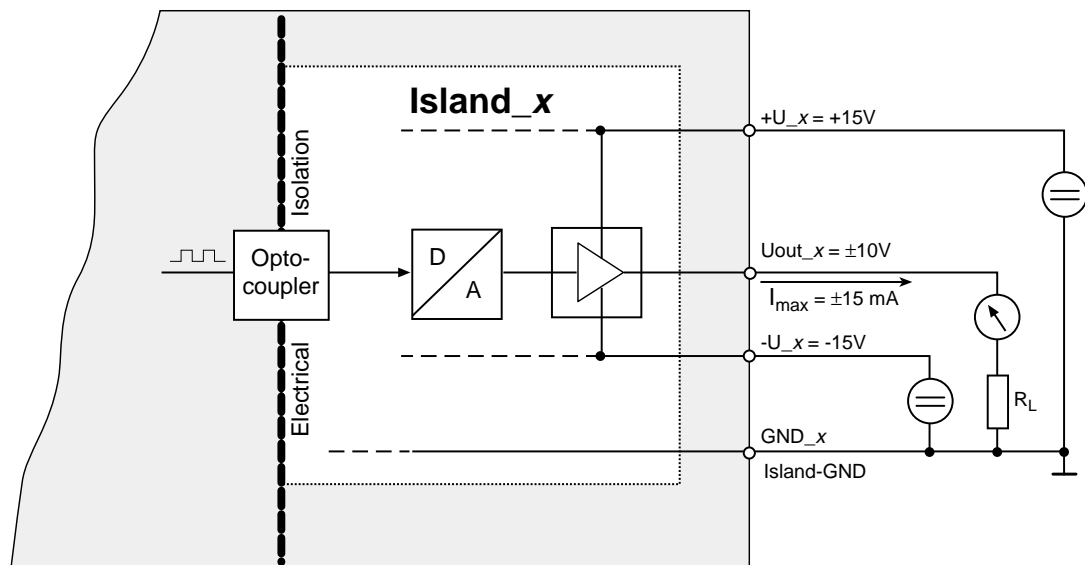


Diagram 6: Island Channels

3.3.4 Option „High Current“

The option „**High Current**“ (HC) can be combined with the board versions without „island channels“. It gives you the possibility to increase the output current per channel to $I_{\max} = \pm 15\text{mA}$. This requires an external, low noise power supply of $\pm 15\text{ V}$ ($\pm 22\text{mA}$ per channel for $I_{\max} = \pm 15\text{mA}$) and a change to your hardware – please contact our service department (see page 55).

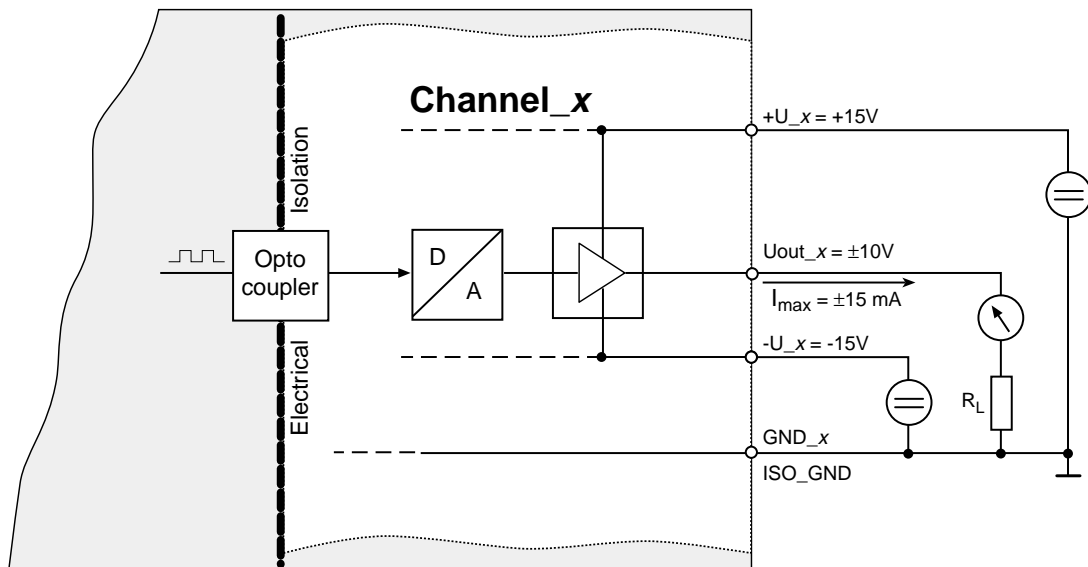


Diagram 7: Switching of the outputs with the option „High Current“ (with electrical isolation)

4 Programming

Before you can work with the board, it has to be configured for the proper operation mode. This is done by appropriate programming with the provided driver software, see chapter 5.3 „Description of the API Functions“ on page 29. Basically after every function call an error check by the function *me6x00GetDrvErrMess* should be done.

For the ME-6100 additionally a **Virtual Instrument Panel** with a graphical user interface is provided. You can use your board at once as a „virtual function generator“ for output of periodically signals.

4.1 Operation Modes

4.1.1 Single Value Output

This operation mode is available for all models of the ME-6000 family. The value to be output is loaded into the appropriate D/A converter and output immediatly (transparent).

Use the following function for configuration and start of the described operation mode:

- *me6x00AOSingle* (A single value is loaded into the D/A converter and output immediatly)

The following example should illustrate the basic order of operation, however it does not claim to be syntactical correct (see also description of the functions on page 29 as well as example programs included with the ME-SDK).

```
//In the following example all functions refer to the board with the
//logical <BoardNumber> „0“
iBoardNumber = 0;
//Transparent output of +10V
return = me6x00AOSingle(iBoardNumber, AO_CH05, 0xFFFF);
if (return == 0) then
    me6x00GetDrvErrMess("ME6000-Test");
endif;
```

4.1.2 Continuous Mode

This operation mode is only available for the model ME-6100.

Mode for independent output of arbitrary, continuously changing signal curves to the channels 0...3. As a rule they are not periodically in opposite to the operation mode „WrapAround“. The D/A-FIFOs of the single channels have to be reloaded continuously. Therefore for every channel a buffer with the values to be output has to be passed. The output operation is controlled by timer. This requires a proper configuration of the timer forcing the conversion of the single values into a fixed time grid (*me6x00AOSetTimer*).

If your program should process further tasks running parallel (depends on the performance of your system), use the function *me6x00AOContinuousEx* (background operation). In that case reloading of the FIFOs has to be done by a user defined callback function (ME-SDK).

The operation will be started calling the function *me6x00AOStart* or by an external trigger edge (see *me6x00AOSetTrigger*). The operation will be stopped **immediately** and **completely** by the functions *me6x00AOSTop(Ex)* and *me6x00AOSTop*.

Note, after calling the function *me6x00AOContinuous(Ex)* the output operation must be really started before you can stop the operation again.

(Diagram see next page)

The following diagram should illustrate the basic order of operation for **Continuous Mode**:

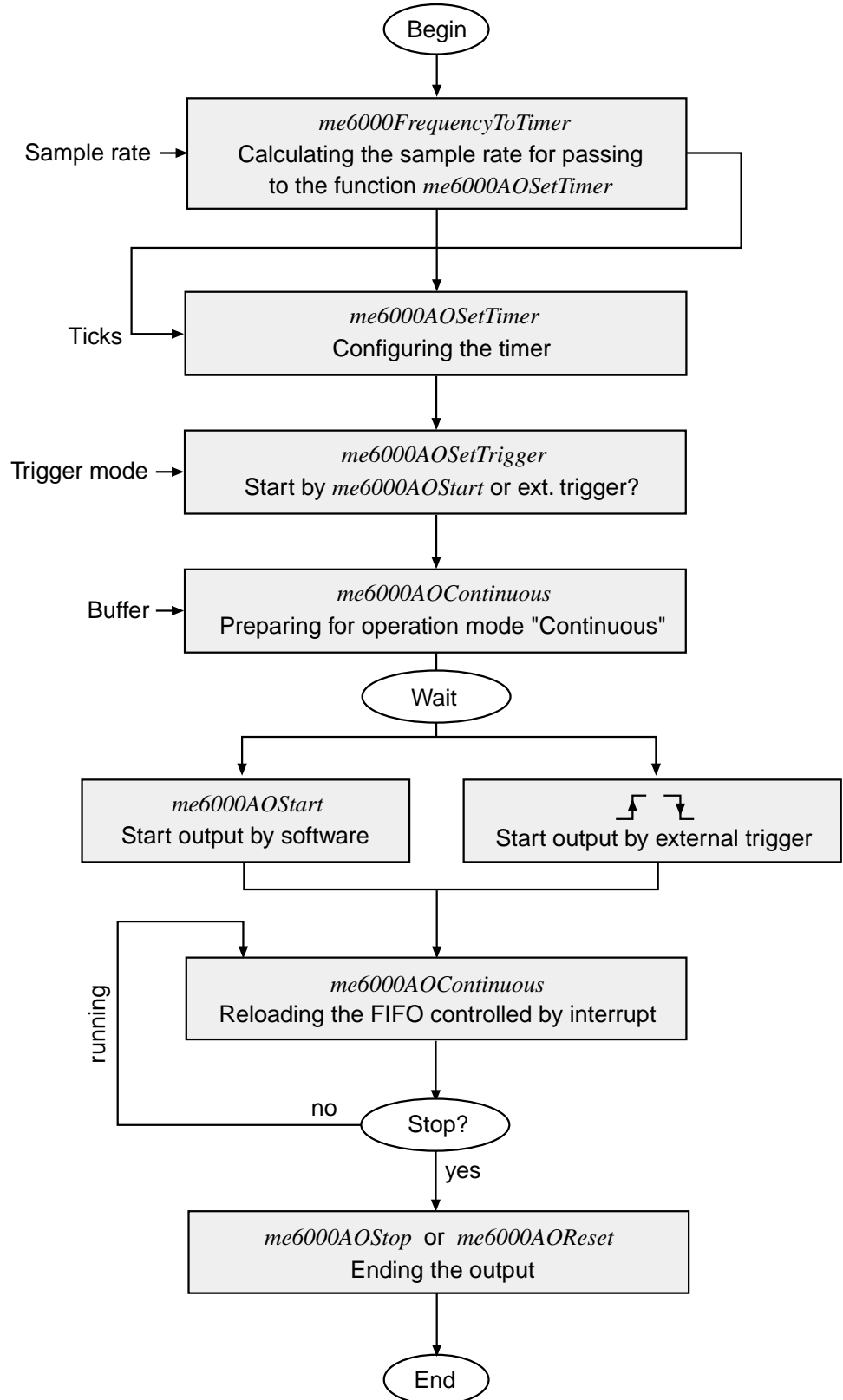


Diagram 8: Programming the „Continuous“ mode

The following example should illustrate the basic order of operation, however it does not claim to be syntactical correct (see also description of the functions on page 29 as well as example programs included with the ME-SDK).

```
//In the following example all functions refer to the board with the
//logical <BoardNumber> „0“. The sample rate should be 300kHz.

iBoardNumber = 0;
iChannel = AO_CH00;
dFreq = 300000;

//Conversion of the sample rate (300000 Hz) into ticks
iticks = me6x00FrequencyToTimer(dFreq);

//Passing the ticks (110 ticks) to the function me6x00AOSetTimer
//for configuring the timers
me6x00AOSetTimer(iBoardNumber, iChannel, iticks);

//The output should be started by software.
me6x00AOSetTrigger(iBoardNumber, iChannel, AO6000_TRIGGER_TIMER);

//Generate an array of values of size BUFFER_SIZE for the first time
...

//Preparing the „Continuous“ mode and passing a pointer to a buffer
//(max. 64k values)
me6x00AOContinuous(iBoardNumber, iChannel, BUFFER_SIZE, psbuffer);

//Starting the continuous output
me6x00AOStart(iBoardNumber, AO_CH00);

while (TRUE)
{
    //Rewriting the buffer of size BUFFER_SIZE
    ...
    me6x00AOContinuous(iBoardNumber, iChannel, BUFFER_SIZE,
psbuffer);
}

//Stopping the „Continuous“ mode completely
me6x00AOStop(iBoardNumber, iChannel);
```

4.1.3 WrapAround Mode

This operation mode is only available for the model ME-6100.

The „WrapAround“ mode is for independent output of periodically signal curves to the channels 0...3. Before starting the operation the D/A-FIFOs of the single channels have to be loaded once. Therefore for every channel a buffer with the values to be output has to be passed. The output operation is controlled by timer on firmware level. This requires a proper configuration of the timer forcing the conversion of the single values into a fixed time grid (*me6x00AOSetTimer*).

The operation will be started calling the function *me6x00AOSTart* or by an external trigger edge (see *me6x00AOSetTrigger*). The operation can be stopped for a while. I.e. the appropriate D/A-channel outputs the last value of the FIFO which results in a defined voltage value after stopping. If there was no change of the operation mode of this channel meanwhile the output operation can be restarted by the function *me6x00AOSTart*. Calling the function *me6x00AOReset* ends the operation **completely** after processing the FIFO.

Note, after calling the function *me6x00AOWrapAround* the output operation must be started at least once before you can stop the operation again.

(Diagram see next page)

The following diagram should illustrate the basic order of operation for **WrapAround Mode**:

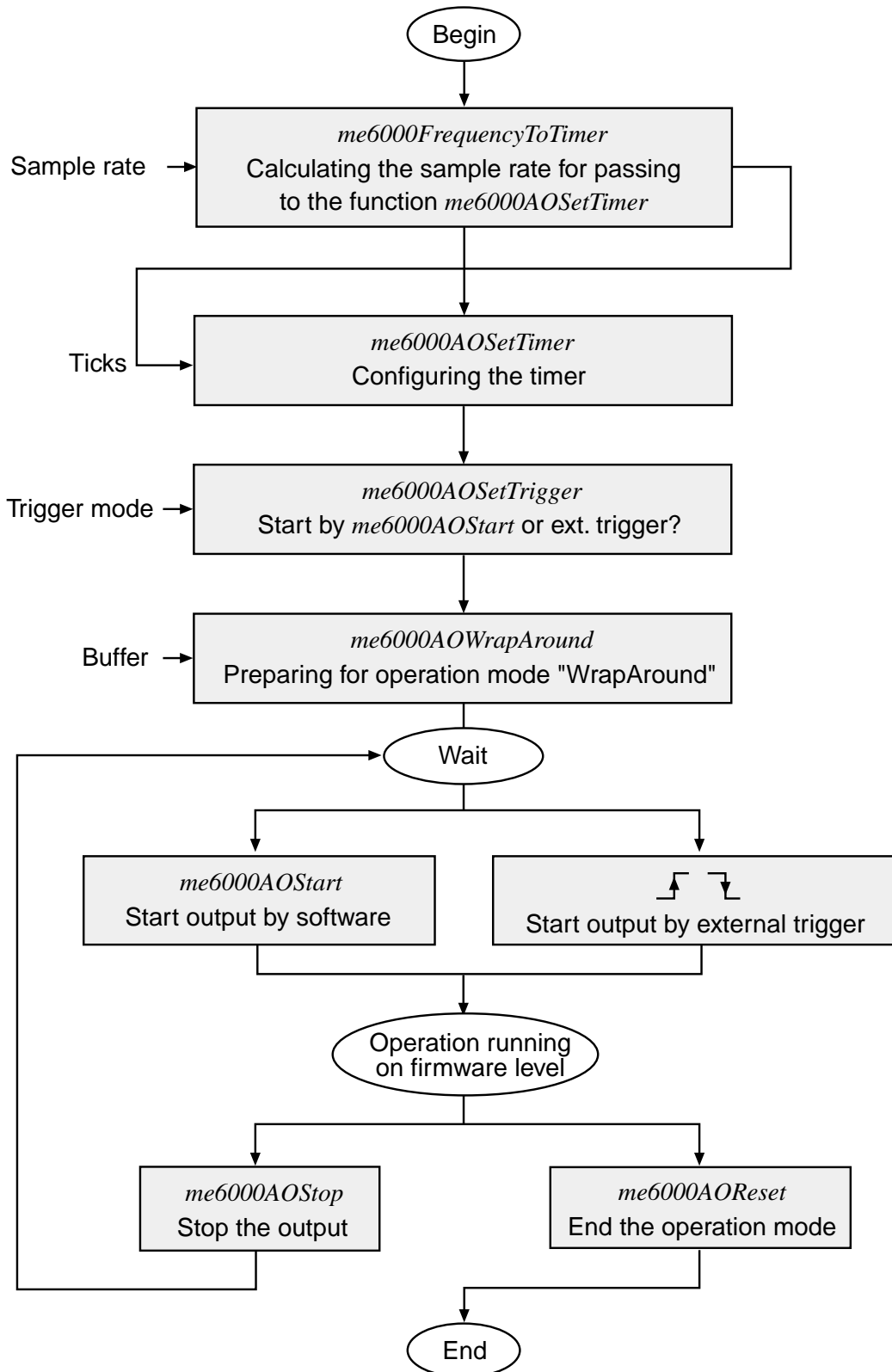


Diagram 9: Programming the „WrapAround“ mode

The following example should illustrate the basic order of operation, however it does not claim to be syntactical correct (see also description of the functions on page 29 as well as example programs included with the ME-SDK).

```
//In the following example all functions refer to the board with the
//logical <BoardNumber> „0“

iBoardNumber = 0;
iChannel = AO_CH00;
dFreq = 320000;

//Conversion of the sample rate in ticks
iticks = me6x00FrequencyToTimer(dFreq);

//Passing the ticks (1650 ticks) to the function me6x00AOSetTimer
//for configuring the timer
me6x00AOSetTimer(iBoardNumber, iChannel, iticks);

//The output should be started by a rising edge of the ext. trigger.
return = me6x00AOSetTrigger(iBoardNumber, iChannel,
AO6000_TRIGGER_EXT_HIGH);

//Output of a sine curve to be approximated by 16 „steps“
//=> frequency of the sine: 20kHz at a sample rate of 320kHz.
//An array of values of size BUFFER_SIZE = 16 has to be generated.
...

//Preparing WrapAround mode
me6x00AOWrapAround(iBoardNumber, iChannel, BUFFER_SIZE, psbuffer);

//Starting the output
me6x00AOSTart(iBoardNumber, iChannel);

...

//Stopping the „WrapAround“ mode temporarily
me6x00AOStop(iBoardNumber, iChannel);

//Restart possible if there was no change in operation mode
//meanwhile
me6x00AOSTart(iBoardNumber, iChannel);

//Stopping the „WrapAround“ mode completely
me6x00AOReset(iBoardNumber, iChannel);
```

4.2 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.

4.2.1 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.

4.3 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.

4.3.1 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.

4.3.2 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.

4.3.3 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 occurs:

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

If necessary, you can install the additional VEE components any time (see „ME-Power-CD“).

4.4 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 included with the appropriate LabVIEW driver. For basics of LabVIEW programming please use your LabVIEW documentation and the LabVIEW online help.

4.4.1 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.

4.4.2 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“.

5 Function Reference

5.1 Overview

The 32 bit driver for the ME-6000/6100 was developed for the Windows driver architecture called „Windows Driver Model“ (WDM). To support Windows NT4.0 additionally a conventional kernel driver is available. A VxD driver as still used under Windows 95 is not planned. The system driver consists of the following components:

- WDM driver (ME6000.SYS) for Windows 98/Me/2000/XP.
- Kernel driver (ME6000.SYS) for Windows NT 4.0.
- API-DLL (ME6000.DLL) with the driver functions for the ME-6000/6100.

After the driver is successfully loaded, the API functions allow convenient access to the hardware. Every function that accesses a ME-6000/6100 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 ME-6000/6100 board and they will as a rule begin with the prefix "*me6x00...*". The function names were selected to be as descriptive as possible. Every function name incorporates several components representing the different function groups (e. g. "AO" for "Analog Output").

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

<i>function name</i>	will be italic in body text e. g. <i>me6x00GetBoardVersion</i>
<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
me6x00... () ;	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:

„5.3.1 General Functions“ on page 30

„5.3.2 Analog Output“ on page 34

„5.3.3 Error Handling“ on page 49

Function	Short description	Page
<i>General Functions</i>		
me6x00FrequencyToTimer	Converts frequency for passing to me6x00AOSetTimer	30
me6x00GetBoardVersion	Determine board version	31
me6x00GetDLLVersion	Determine DLL version number	32
me6x00GetDriverVersion	Determine driver version	32
<i>Analog Output</i>		
me6x00AOContinuous/Ex	Start continuous output	34
me6x00AOReset	Reset channel to 0 V	37
me6x00AOResetAll	Total reset of the board	38
me6x00AOSetTimer	Configure D/A timer	39
me6x00AOSetTrigger	Configure trigger modi	40
me6x00AOSingle	Output a single value (transparent)	41
me6x00AOStart	Start continuous or wraparound mode	42
me6x00AOStop/Ex	Stop continuous or wraparound mode	43
me6x00AOWaveGen	Simple „function generator“	46
me6x00AOWraparound	Prepare periodic output	48
<i>Error Handling</i>		
me6x00GetDrvErrMess	Error string corresponding to error code	49

Table 3: Overview of library functions

5.3.1 General Functions

me6x00FrequencyToTimer

Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

Converts the sample rate (max. 500 kHz) to the number of „ticks“ for passing to the function *me6x00AOSetTimer*. A minimum of 66 ticks correspond with 500 kHz; max. $(2^{32}-1)$ ticks correspond with a period of approx. 130s. (see also *me6x00AOSetTimer*).

Note: The sample rate is not the frequency of a signal to be output periodically.

Definitions

C: int me6x00FrequencyToTimer(double dFreq);
 Delphi: Function me6x00FrequencyToTimer(dFreq: double):
 integer;
 Basic: Declare Function me6x00FrequencyToTimer Lib
 „me6x00“ Alias „_VBme6x00FrequencyToTimer@8“
 (ByVal dFreq As Double) As Long

Parameters

<Freq> Conversion rate in [Hz];
 value range: 0,008...500000

Return value

Returns the calculated „ticks“ (optimum approximation) as an integer value.

me6x00GetBoardVersion

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

Determines the version number of an installed board of the board family ME-6000/6100.

● Definitions

- C: int me6x00GetBoardVersion (int iBoardNumber, int *piVersion)
- Delphi: Function me6x00GetBoardVersion (iBoardNumber: integer; Var iVersion: integer): integer;
- Basic: Declare Function me6x00GetBoardVersion Lib "me6x00" Alias "_VBme6x00GetBoardVersion@8" (ByVal iBoardNumber As Long, iVersion As Long) As Long

➔ Parameters

- <BoardNumber> Number of the board to be accessed of type ME-6000/6100 (0...31)
- <Version> Pointer to an integer value where the device ID is returned. Possible values:
- | | |
|----------|---------------------|
| 6004Hex: | ME-6000/4 Standard |
| 6008Hex: | ME-6000/8 Standard |
| 600FHex: | ME-6000/16 Standard |
| 6014Hex: | ME-6000/4 Opto |
| 6018Hex: | ME-6000/8 Opto |
| 601FHex: | ME-6000/16 Opto |
| 6034Hex: | ME-6000/4 Island |
| 6038Hex: | ME-6000/8 Island |
| 603FHex: | ME-6000/16 Island |
| 6104Hex: | ME-6100/4 Standard |
| 6108Hex: | ME-6100/8 Standard |
| 610FHex: | ME-6100/16 Standard |
| 6114Hex: | ME-6100/4 Opto |
| 6118Hex: | ME-6100/8 Opto |
| 611FHex: | ME-6100/16 Opto |
| 6134Hex: | ME-6100/4 Island |
| 6138Hex: | ME-6100/8 Island |
| 613FHex: | ME-6100/16 Island |

< 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 *me6x00GetDrvErrMess*.

me6x00GetDLLVersion

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

Determines the version number of the driver DLL.

● Definitions

C: int me6x00GetDLLVersion();
 Delphi: Function me6x00GetDLLVersion: integer;
 Basic: Declare Function me6x00GetDLLVersion Lib
 "me6x00_32" Alias "_VBme6x00GetDLLVersion@0" () As
 Long

➔ **Parameters** none

< Return value

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

me6x00GetDriverVersion

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

Determines the driver version of the ME-6000/6100.

● Definitions

C: int me6x00GetDriverVersion(int *piBuffer);
Delphi: Function me6x00GetDriverVersion (Var iBuffer: integer):
 integer;
Basic: Declare Function me6x00GetDriverVersion Lib „me6x00“
 Alias "_VBme6x00GetDriverVersion@4" (ByVal pBuffer
 As Long) As Long

➔ Parameters

<Buffer> Pointer to an integer value where the driver
 version is returned.

< Return value

If the function is successfully executed, a '1' is returned. If an error occurs, a '0' is returned. The cause of the error can be determined with the function *me6x00GetDrvErrMsg*.

5.3.2 Analog Output

me6x00AOContinuous

Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

This function is for preparing the operation mode „Continuous“. Use it for independent output of arbitrary, continuously changing signal curves to the channels 0...3. As a rule they are not periodically (in opposite to the operation mode „WrapAround“). The D/A-FIFOs of the single channels have to be reloaded continuously. Generate a buffer of defined size (max. 64k values) for every channel with the values to be output. The output operation will be controlled by timer. This requires a proper configuration of the timer forcing the conversion of the single values into a fixed time grid (sample rate); see *me6x00AOSetTimer*.

Calling the function *me6x00AOContinuous* for the first time ends any operation possibly active on the appropriate channel. The channel will be set to 0V, the D/A-FIFO will be cleared and loaded for the first time afterwards. The operation will be started calling the function *me6x00AOStart* or by an external trigger edge (see *me6x00AOSetTrigger*). By the functions *me6x00AOStop* and *me6x00AOReset* the output will be stopped **immediately** and **completely**.

See also chapter “Operation Modes“ on page 17 and the programming examples included with the ME-Software-Developer-Kit.

Note!

Note, after calling the function *me6x00AOContinuous* the output operation must be really started before you can stop the operation again. Else a malfunction can occur!

Definitions

C: int me6x00AOContinuous (int iBoardNumber, int iChannel, int iSize, short *psBuffer);

Delphi: Function me6x00AOContinuous (iBoardNumber, iChannel, iSize: integer; psBuffer: tmeAOBuffer): integer;

Basic: Declare Function `me6x00AOContinuous` Lib „me6x00“
 Alias „_VBme6x00AOContinuous@12“ (ByVal
 IBoardNumber As Long, ByVal IChannel As Long, ByVal
 ISize As Long, ByVal piBuffer As Integer) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type
 ME-6000/6100 (0...31)

<Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel 0...3

<Size> Size of the output buffer (max. 64k values)

<Buffer> Pointer to the output buffer with the values to be
 output

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error
 occurs a „0“ is returned. The exact cause of the error can be deter-
 mined by the function `me6x00GetDrvErrMess`.

me6x00AOContinuousEx

🔪 Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

Version of `me6x00AOContinuous` (see above) for background
 operation. This function enables you to process further tasks running
 parallel (depends on the performance of your system). In that case
 reloading the FIFOs has to be done by a user defined callback
 function (see example included with the ME-SDK).

The operation will be started calling the function `me6x00AOStart` or
 by an external trigger edge (see `me6x00AOSetTrigger`). Stop the out-
 put by the function `me6x00AOStopEx`, if <Loops> is passed with the
 constant `AO6000_INFINITE`.

See also chapter “Operation Modes“ on page 17 and the pro-
 gramming examples included with the ME-Software-Developer-Kit.

☞ Note!

Note, after calling the function *me6x00AOContinuousEx* the output operation must be really started before you can stop the operation again (in dependency of parameter <Loops>). If you use AO6000_INFINITE stop the output by the function *me6x00AOSTopEx*. Else a malfunction can occur!

● Definitions

C: int me6x00AOContinuousEx (int iBoardNumber, int iChannel, int iSize, short *psBuffer, int iLoops, PVOID_PROC pCallback, PVOID_PROC pArgs);

Delphi: Function me6x00AOContinuousEx (iBoardNumber, iChannel, iSize: integer; psBuffer: tmeAOBuffer; iLoops: integer; pCallback, pArgs: PVOID_PROC): integer;

Basic: Declare Function me6x00AOContinuousEx Lib „me6x00“ Alias „_VBme6x00AOContinuousEx@24“ (ByVal lBoardNumber As Long, ByVal lChannel As Long, ByVal lSize As Long, ByVal piBuffer As Integer, ByVal lLoops As Long, ByVal pCallback As Long, ByVal pArgs As Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type ME-6000/6100 (0...31)

<Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH03	Channel number (00Hex...03Hex)

<Size> Size of the output buffer (max. 64k values)

<Buffer> Pointer to the output buffer with the values to be output

<Loops> Number of repeats. For an infinite loop use the constant AO6000_INFINITE

<Callback> Pointer to user defined callback function

<Args> Optional a parameter can be passed to the callback function (else pass NULL pointer)

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOReset

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

This function stops an operation in process of the appropriate channel in the operation modes „Continuous“ or „WrapAround“. In „Continuous“ mode the output will be stopped immediately and completely. In opposite to the „WrapAround“ mode the operation will be stopped at the last value of the FIFO. Then the corresponding FIFO will be cleared and the channel will be set to 0V.

👉 Note!

Before running this function, you have to guarantee, that the output has been really started by *me6x00AOSTart* or by an external trigger signal after calling the function *me6x00AOContinuous* resp. *me6x00AOWraparound*. Else a malfunction can occur!

● Definitions

C: int me6x00AOReset (int iBoardNumber, int iChannel);
 Delphi: Function me6x00AOReset (iBoardNumber, iChannel: integer): integer;
 Basic: Declare Function me6000AOReset Lib "me6x00" Alias "_VBme6x00AOReset@8" (ByVal iBoardNumber As Long, ByVal IChannel As Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type ME-6000/6100 (0...31)
 <Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH15	Channel number (00Hex...15Hex)

◀ 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 *me6x00GetDrvErrMess*.

me6x00AOResetAll

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

This function stops any operation of the appropriate board. A general reset will be done. All channels are set to 0V.

● Definitions

C: int me6x00AOResetAll (int iBoardNumber);
 Delphi: Function me6x00AOResetAll (iBoardNumber: integer):
 integer;
 Basic: Declare Function me6000AOResetAll Lib "me6x00" Alias
 "__VBme6x00AOResetAll@4" (ByVal iBoardNumber As
 Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type
 ME-6000/6100 (0...31)

< 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 *me6x00GetDrvErrMess*.

me6x00AOSetTimer

🔪 Description

Model:	ME-6000	ME-6100
available:	-	Channel 0...3

Configures the 32 bit counter used as a timer for the operation modes „Continuous“ and „WrapAround“. As time base a 33MHz clock is used. From that results a period of 30,30ns, which will be defined as the minimum time unit and is named „1 tick“ in the following. The sample rate has to be passed as a multiple of one tick to the parameter <Ticks>. With a given sample rate f_{Sample} the number of ticks will be calculated as follows:

General:
$$\frac{1}{f_{\text{Sample}} \cdot 30,30\text{ns}} = \text{Ticks}$$

Number of ticks for the **max. sample rate:**

$$\frac{1}{500\text{kHz} \cdot 30,30\text{ns}} = \underline{\underline{66\text{Ticks (42Hex)}}}$$

Min. sample rate (i. e. max. number of ticks = $(2^{32}-1)$):

$$f_{\text{Sample}} = \frac{1}{(2^{32} - 1)\text{Ticks} \cdot 30,30\text{ns}} = 0.00768\text{Hz} \Rightarrow T_{\text{Sample}} \approx \underline{\underline{130\text{s}}}$$

I. e. the sample rate can be set in steps of 30,30ns between the minimum and maximum sample rate. For passing the sample rate to the function *me6x00AOSetTimer* the function *me6x00FrequencyToTimer* offers a simple possibility of conversion. See chapter „Programming“ on page17 and the programming examples included with the ME Software Developer Kit.

● Definitions

- C: `int me6x00AOSetTimer (int iBoardNumber, int iChannel, int iTicks);`
- Delphi: `Function me6x00AOSetTimer (iBoardNumber, iChannel, iTicks: integer): integer;`
- Basic: `Declare Function me6x00AOSetTimer Lib „me6x00“ Alias „_VBme6x00AOSetTimer@8“ (ByVal IBoardNumber As Long, ByVal IChannel As Long, ByVal ITicks As Long) As Long`

→ Parameters

<BoardNumber>	Number of the board to be accessed of type ME-6000/6100 (0...31)				
<Channel>	Channel number; possible values: <table> <thead> <tr> <th><u><Channel></u></th> <th><u>Description</u></th> </tr> </thead> <tbody> <tr> <td>AO_CH00...AO_CH03 (00Hex...03Hex)</td> <td>Channel number</td> </tr> </tbody> </table>	<u><Channel></u>	<u>Description</u>	AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number
<u><Channel></u>	<u>Description</u>				
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number				
<Ticks>	Number of ticks for the 32 bit timer, which determines the sample rate for the operation modes „Continuous“ and „WrapAround“. The value range is between 66 and $(2^{32}-1)$ ticks (00000042...FFFFFFFFHex) (see above).				

< Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOSetTrigger

🔪 Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

Determines the type of conversion start for the operation modes „Continuous“ and „WrapAround“. You can choose between software start (see *me6x00AOSTart*) and external trigger (positive/negative edge).

● Definitions

C:	int me6x00AOSetTrigger (int iBoardNumber, int iChannel, int iModePolarity);
Delphi:	Function me6x00AOSetTrigger (iBoardNumber, iChannel, iModePolarity: integer): integer;
Basic:	Declare Function me6x00AOSetTrigger Lib „me6x00“ Alias „_VBme6x00AOSetTrigger@8“ (ByVal IBoardNumber As Long, ByVal IChannel As Long, ByVal IModePolarity As Long) As Long

➔ Parameters

<BoardNumber>	Number of the board to be accessed of type ME-6000/6100 (0...31)	
<Channel>	Channel number; possible values:	
	<u><Channel></u>	<u>Description</u>
	AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number
<ModePolarity>	Condition for conversion start by setting trigger source and trigger edge	
	<u><ModePolarity></u>	<u>Description</u>
	AO6X00_TRIGGER_TIMER (00Hex)	by software (<i>me6x00AOSTart</i>)
	AO6X00_TRIGGER_EXT_HIGH (10Hex)	Ext. trigger, rising edge
	AO6X00_TRIGGER_EXT_LOW (30Hex)	Ext. trigger, falling edge

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOSingle

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

Function is used for „transparent“ output of a voltage value to a certain channel. Each output running (Single, Continuous or Wrap-Around) will be stopped. The corresponding FIFO will be cleared and the value is output directly. No preparation by other functions is necessary.

● Definitions

C:	int me6x00AOSingle (int iBoardNumber, int iChannel, int iValue);
Delphi:	Function me6x00AOSingle (iBoardNumber, iChannel, iValue: integer): integer;

Basic: Declare Function me6x00AOSingle Lib „me6x00“ Alias
 „_VBme6x00AOSingle@12“ (ByVal IBoardNumber As
 Long, ByVal IChannel As Long, ByVal IValue As Long) As
 Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type
 ME-6000/6100 (0...31)

<Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH15 (00Hex...15Hex)	Channel number

<Value> 16 bit output value. The value range is between:
 0000Hex (-10 V) ... FFFFHex (+10 V)

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOSTart

🔪 Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

The function starts the analog output of a certain channel with the operation modes „Continuous“ or „WrapAround“. If you have chosen the option „external trigger“ in the function *me6x00AOSetTrigger*, the output operation will be started by a certain edge at the external trigger input. In that case *me6x00AOSTart* must not be called.

See chapter „Programming“ on page17 and the programming examples included with the ME Software Developer Kit.

● Definitions

C: int me6x00AOSTart (int iBoardNumber, int iChannel);
 Delphi: Function me6x00AOSTart (iBoardNumber, iChannel:
 integer): integer;

Basic: Declare Function me6x00AOSTart Lib „me6x00“ Alias
 "_VBme6x00AOSTart@8" (ByVal lBoardNumber As Long,
 ByVal lChannel As Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type
 ME-6000/6100 (0...31)

<Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOSTop

🔪 Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

The function stops the analog output of a certain channel started with the operation modes „Continuous“ or „WrapAround“. If running the operation mode „Continuous“ the operation is stopped immediately and completely. I.e. any voltage value is assigned to the appropriate channel after ending the output and the FIFO will be cleared. Using the operation mode „WrapAround“ the output will be stopped with the last value in the FIFO, i.e. a known voltage value is assigned to the appropriate D/A-channel after ending the output. If there was no change of the operation mode of this channel the output can be restarted by the function *me6x00AOSTart*.

See chapter „Programming“ on page17 and the programming examples included with the ME Software Developer Kit.

👉 Note!

Before running this function, you have to guarantee, that the output has been really started by *me6x00AOSTart* or by an external trigger signal after calling the function *me6x00AOCContinuous* resp. *me6x00AOWraparound*. Else a malfunction can occur!

● Definitions

C: int me6x00AOStop (int iBoardNumber, int iChannel);
 Delphi: Function me6x00AOStop (iBoardNumber, iChannel: integer): integer;
 Basic: Declare Function me6x00AOStop Lib „me6x00“ Alias "_VBme6x00AOStop@4" (ByVal IBoardNumber As Long, ByVal IChannel As Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type ME-6000/6100 (0...31)
 <Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOStopEx

🔪 Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

The function ends a background process, if the „Continuous“ mode has been started by the function *me6x00AOContinuousEx* and the constant AO6000_INFINITE was passed in parameter <Loops>. The operation is stopped immediately, i.e. any voltage value is assigned to the appropriate channel after ending the output.

👉 Note!

Before running this function, you have to guarantee, that the output has been really started by *me6x00AOStart* or by an external trigger signal after calling the function *me6x00AOContinuousEx*. Else a malfunction can occur!

● Definitions

C: int me6x00AOSTopEx (int iBoardNumber, int iChannel);
 Delphi: Function me6x00AOSTopEx (iBoardNumber, iChannel:
 integer): integer;
 Basic: Declare Function me6x00AOSTopEx Lib „me6x00“ Alias
 " _VBme6x00AOSTopEx@4" (ByVal IBoardNumber As
 Long, ByVal IChannel As Long) As Long

➔ Parameters

<BoardNumber> Number of the board to be accessed of type
 ME-6000/6100 (0...31)
 <Channel> Channel number; possible values:

<u><Channel></u>	<u>Description</u>
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel number

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

me6x00AOWaveGen

Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

Function serves you a virtual function generator (rectangle, sine, triangle,...) which can be easily programmed. The complete configuration of the certain channel is done by that function.

The output will be started automatically on calling this function and will be stopped by the function *me6x00AOSTop*. The signal is always output with the max. number of „steps“ (minimum 5) in dependency of the selected frequency and shape (rectangle max. 250kHz, others max. 100kHz). Exception: A rectangle signal is always output with 2 steps per period. See also chapter “WrapAround Mode“ on page 21. See programming examples included with the ME-SDK.

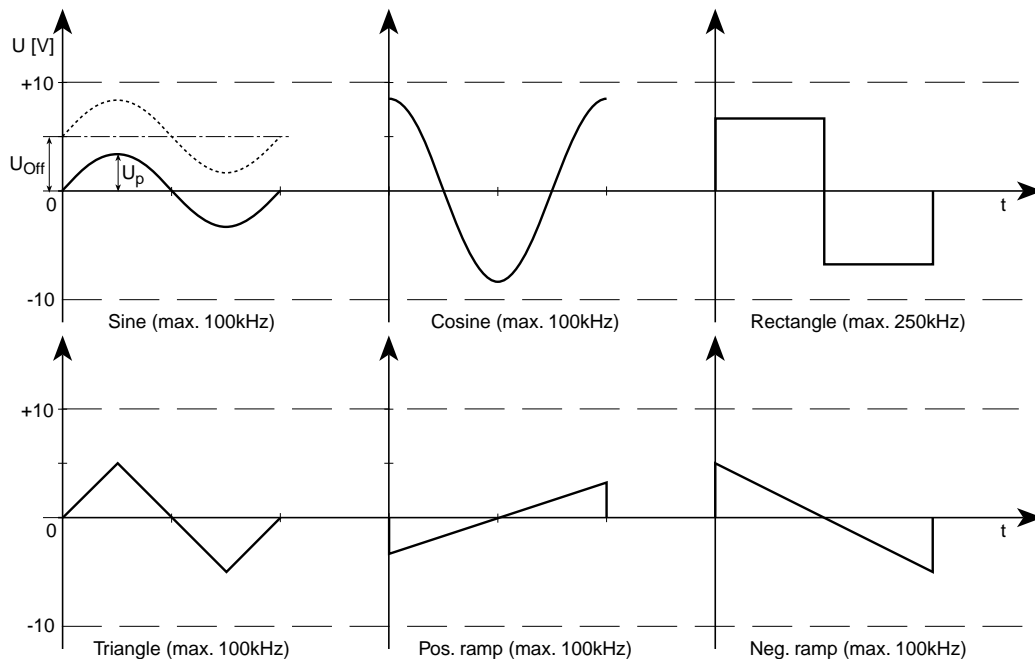


Abb. 10: Offset, amplitude, shapes

Note!

If all parameters together exceed the limits given by the hardware the driver displays an error message.

Using the external trigger can not be combined with that function. Please use the functions *me6x00AOCContinuous* and *me6x00AOWrapAround*.

● Definitions

- C: `int me6x00AOWaveGen (int iBoardNumber, int iChannel, int iShape, double dAmplitude, double dOffset, double dFreq);`
- Delphi: `Function me6x00AOWaveGen (iBoardNumber, iChannel, iShape: integer; dAmplitude, dOffset, dFreq: double): integer;`
- Basic: `Declare Function me6x00AOWaveGen Lib „me6x00“ Alias „_VBme6x00AOWaveGen@24“ (ByVal lBoardNumber As Long, ByVal lChannel As Long, ByVal lShape As Long, ByVal dAmplitude As Double, ByVal dOffset As Double, ByVal dFreq As Double) As Long`

➔ Parameters

- <BoardNumber> Number of the board to be accessed of type ME-6000/6100 (0...31)
- <Channel> Channel number; possible values:
- | <u><Channel></u> | <u>Description</u> |
|-----------------------------------|--------------------|
| AO_CH00...AO_CH03 (00Hex...03Hex) | Channel number |
- <Shape> signal curve; possible values:
- | <u><Shape></u> | <u>Description</u> |
|----------------------|--------------------|
| AO_RECTANGLE (00Hex) | Rectangle signal |
| AO_TRIANGLE (01Hex) | Triangle signal |
| AO_SINUS (02Hex) | Sine signal |
| AO_COSINUS (03Hex) | Cosine signal |
| AO_POS_RAMP (04Hex) | Positive ramp |
| AO_NEG_RAMP (05Hex) | Negative ramp |
- <Amplitude> Signal amplitude as a 16 bit value; the value range is between: 0000Hex (-10 V) ... FFFFHex (+10 V)
- <Offset> Offset voltage U_{Off} [V] for moving the signal in positive or negative direction; value range: -10,00...+10,00
- <Freq> Frequency in [Hz] of the signal to be output periodically; value range: 0...100000 (rectangle up to 250000)

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function `me6x00GetDrvErrMess`.

me6x00AOWrapAround

Description

Model:	ME-6000	ME-6100
available:	–	Channel 0...3

This function is for preparing the operation mode „WrapAround“. Use it for independent output of periodic signals to the channels 0...3. Before starting the operation the single channel's FIFOs have to be loaded once. Generate a buffer of defined size (max. 64k values) for every channel with the values to be output. The output operation is controlled by timer on firmware level. This affords a proper configuration of the timer forcing the conversion of the single values into a fixed time grid (*me6x00AOSetTimer*).

Calling the function *me6x00AOWrapAround* ends any operation possibly active on the appropriate channel. The channel will be set to 0V, the D/A-FIFO will be cleared and reloaded afterwards. The operation will be started by the function *me6x00AOSTart* or by an external trigger edge (see *me6x00AOSetTrigger*). By the function *me6x00AOSTop* the output can be stopped **for a while** at the last value of the FIFO. I.e. a defined voltage value is assigned. If there was no change of the operation mode of this channel meanwhile, the output operation can be restarted by the function *me6x00AOSTart*. In comparison to *me6x00AOSTop* the function *me6x00AOReset* also clears the FIFO and ends the operation **completely**.

See also chapter “Programming“ on page 17 and the programming examples included with the ME-Software-Developer-Kit.

Note!

Note, after calling the function *me6x00AOWraparound* the output operation must be started at least once before you can stop the operation by the functions *me6x00AOSTop* or *me6x00AOReset*. Else a malfunction can occur!

Definitions

- C: `int me6x00AOWrapAround (int iBoardNumber, int iChannel, int iSize, short *psBuffer);`
- Delphi: `Function me6x00AOWrapAround (iBoardNumber, iChannel, iSize: integer; psBuffer: tmeAOBuffer): integer;`
- Basic: `Declare Function me6x00AOWrapAround Lib „me6x00“ Alias „_VBme6x00AOWrapAround@12“ (ByVal lBoardNumber As Long, ByVal lChannel As Long, ByVal lSize As Long, ByVal piBuffer As Integer) As Long`

➔ Parameters

<code><BoardNumber></code>	Number of the board to be accessed of type ME-6000/6100 (0...31)				
<code><Channel></code>	Channel number; possible values:				
	<table border="0" style="width: 100%;"> <tr> <td style="text-align: left;"><u><code><Channel></code></u></td> <td style="text-align: left;"><u>Description</u></td> </tr> <tr> <td>AO_CH00...AO_CH03 (00Hex...03Hex)</td> <td>Channel 0...3</td> </tr> </table>	<u><code><Channel></code></u>	<u>Description</u>	AO_CH00...AO_CH03 (00Hex...03Hex)	Channel 0...3
<u><code><Channel></code></u>	<u>Description</u>				
AO_CH00...AO_CH03 (00Hex...03Hex)	Channel 0...3				
<code><Size></code>	Size of the output buffer (max. 64k values)				
<code><Buffer></code>	Pointer to the output buffer with the values to be output				

◀ Return value

If the function is successfully executed, a „1“ is returned. If an error occurs a „0“ is returned. The exact cause of the error can be determined by the function *me6x00GetDrvErrMess*.

5.3.3 Error Handling

me6x00GetDrvErrMess

🔪 Description

Model:	ME-6000	ME-6100
available:	✓	✓

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

👉 Important Note!

This function can only be called if the previously called API function of ME6000.DLL returned an error (error code „0“)!

● Definitions

C:	int me6x00GetDrvErrMess (char *pcErrortext, int iBufferSize);
Delphi:	Function me6x00GetDrvErrMess (Var errortext: errorstring; iBufferSize: integer): integer;
Basic:	Declare Function me6x00GetDrvErrMess Lib "me6x00" Alias "_VBme6x00GetDrvErrMess@4" (ByVal errortext As String, ByVal iBufferSize As Long) As Long

→ Parameters

- <ErrorText> Pointer to a string; returns the error code.
- <BufferSize> Buffer size in number of characters for <ErrorText> will be allocated (recommended: max. 128 characters).

< Return value

The function returns the DLL global variable <DLLErrorCode>

Appendix

A Specifications

(Ambient temperature 25°C)

PC Interface

Bus system Standard PCI (32 Bit, 33 MHz);
Plug&Play function resources assigned automatically

Voltage Outputs

Number of channels 4, 8 or 16 (depends on model)
D/A converter 1 serial converter (500 kHz) per channel
Resolution 16 bit
Output range ± 10 V
Output current Without external power supply: depends on the number of assembled resp. used channels:

Channels	I_{\max} per channel
4	15mA
8	15mA
12	10mA
16	3mA

With external power supply (± 15 V) only in connection with the options „Island channels“ and „High Current“: max. ± 15 mA per channel

Ext. power supply ± 15 V (optional); current per channel: 7mA + load (max. ± 15 mA)
Operation modes Single value output (transparent)
Optional:
Continuous-Mode (output continuously)
WrapAround-Mode (output periodically)

Total accuracy:
„With electrical isolation“ max. ± 20 mV
„With island channels“ max. ± 10 mV
Settling time (DAC) max. 2 μ s at full scale (-10V \rightarrow +10V)

External Trigger (only ME-6100)

Input current I_F	$7,5\text{mA} \leq I_F \leq 10\text{mA}$ (against GND)
Voltage level	typ. 5V
Delay time	max. 80ns
Time uncertainty on synchronous start	max. 15ns

Timer controlled output (only ME-6100)

Channels	0...3 (independent from each another)
D/A-FIFOs	8k values per channel
Time range	from $2\mu\text{s}$ up to 130s in steps of $30, \overline{30}\text{ns}$

Electrical Isolation, Island Channels (optional)

Overvoltage protection	max. 500V
------------------------	-----------

General Information

Power consumption at +5V (with 16 channels; without ext. load):	
„With electrical isolation“	max. 3,6A
„With island channels“	max. 1,2A
Physical size	PCI: 174 mm x 99 mm (without mounting bracket and connector)
Connectors	78pin D-Sub female connector
Operating temperature	0...70°C
Storage temperature	0...50 °C
Relative humidity	20...55% (non condensing)

CE Certification

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

B Pinout

B1 Pinout D-Sub Connector

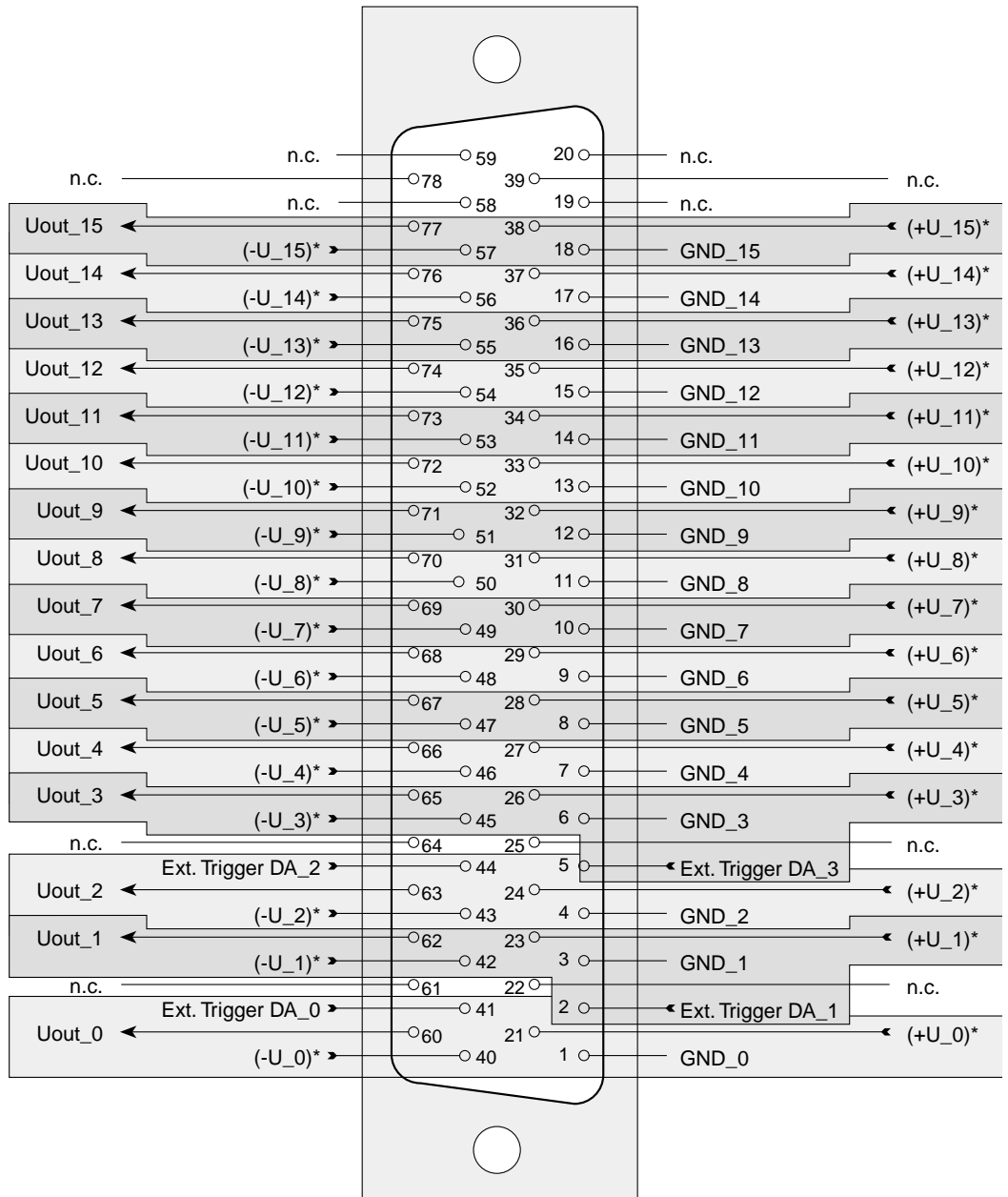


Diagram 11: Pinout of the 78pin D-Sub female connector



*** Attention:** With the options „High Current“ and „Island Channels“ the pins -U_x and +U_x are inputs for the external ±15V power supply. In all other cases these pins output ±15V and it is not permitted to connect them. **The hardware will be irreversible damaged!**

C Accessories

We recommend to use a high quality connector cable with single shielded lines per channel. There is the special connector cable ME-AK-D78/6000 (length: 1m) included with the package. It is also available as an accessories article.

ME-AK-D78/6000

Special connector cable from 78pin D-Sub male connector to 16 single shielded lines with open end. **Note the warning on p. 53!**

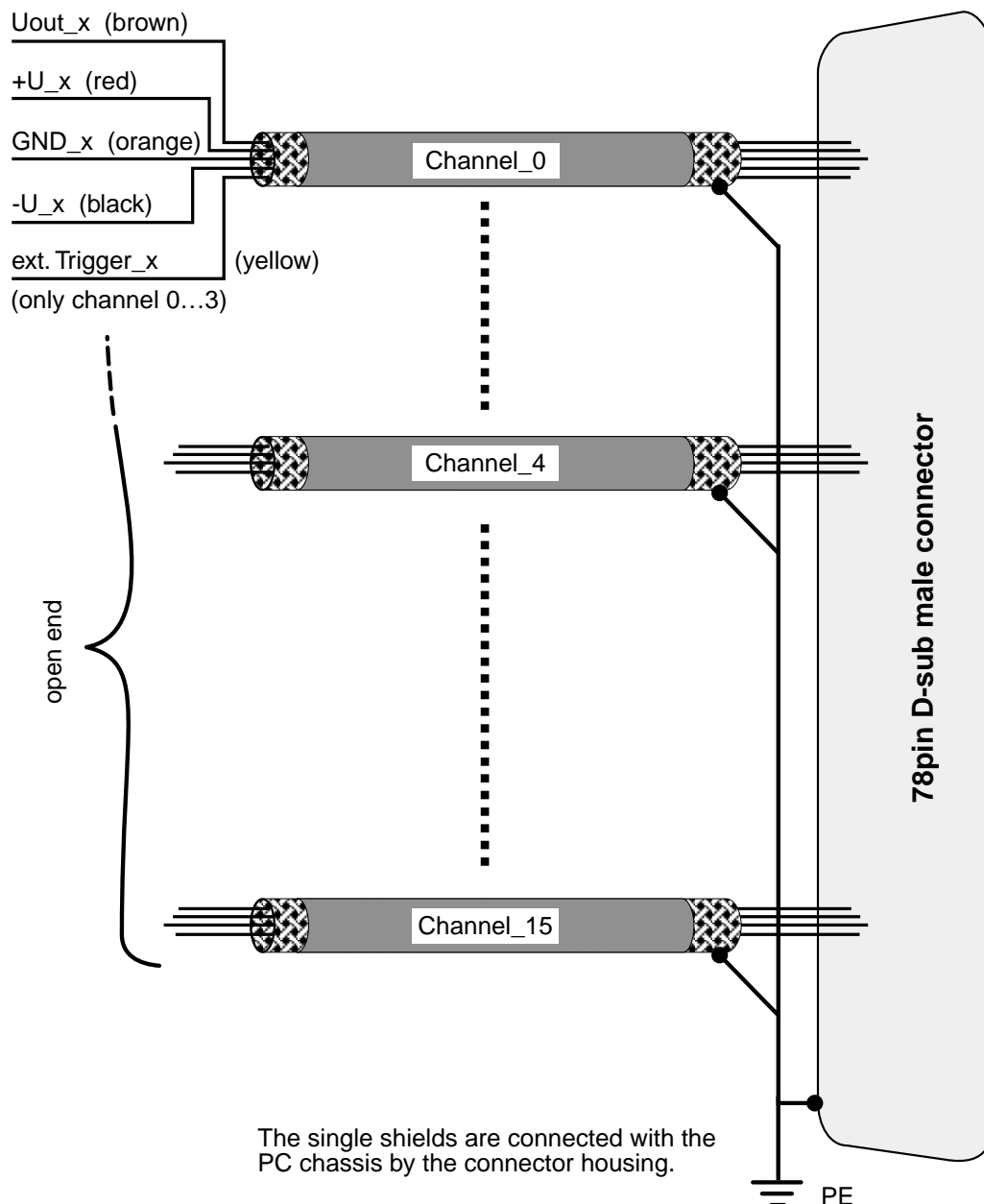


Diagram 12: Special cable ME-6000 (ME-AK-D78/6000)

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

Function Reference

me6x00AOContinuous 34
 me6x00AOContinuousEx 35
 me6x00AOReset 37
 me6x00AOResetAll 38
 me6x00AOSetTimer 39
 me6x00AOSetTrigger 40
 me6x00AOSingle 41
 me6x00AOSTart 42
 me6x00AOSTop 43
 me6x00AOSTopEx 44
 me6x00AOWaveGen 46
 me6x00AOWrapAround 48
 me6x00GetBoardVersion 31
 me6x00GetDLLVersion 32
 me6x00GetDriverVersion 32
 me6x00GetDrvErrMess 49

A

Accessories 54
 Analog Output
 me6x00AOContinuous 34
 me6x00AOContinuousEx 35
 me6x00AOReset 37
 me6x00AOResetAll 38
 me6x00AOSetTimer 39

me6x00AOSetTrigger 40
 me6x00AOSingle 41
 me6x00AOSTart 42
 me6x00AOSTop 43
 me6x00AOSTopEx 44
 me6x00AOWaveGen 46
 me6x00AOWrapAround 48

API-DLL 27

Appendix 51

B

Block Diagram 11

C

Continuous Mode 18

D

Description of the API Functions 29
 Driver overview 27
 Driver Update 55
 D-Sub connector 53

E

Electrical Isolation 14
 Error Handling
 me6x00GetDrvErrMess 49
 Example Programs 24

F

Features 6

-
- Function Reference 27
 - G**
 - General Functions
 - me6x00FrequencyToTimer 30
 - me6x00GetBoardVersion 31
 - me6x00GetDLLVersion 32
 - me6x00GetDriverVersion 32
 - H**
 - Hardware Description 11
 - Electrical Isolation 14
 - High Current Option 16
 - Island Channels 15
 - I**
 - Introduction 5
 - Island Channels 15
 - K**
 - Kernel driver 27
 - L**
 - LabVIEW™
 - Example programs 26
 - Programming 26
 - Virtual Instruments 26
 - M**
 - ME Board Menu 25
 - Model Overview 6
 - O**
 - Operation Modes
 - Continuous Mode 18
 - Single Value Output 17
 - WrapAround Mode 21
 - P**
 - Package contents 5
 - Pinout 53
 - Programming 17
 - Order of Operation 24
 - under High Level Languages 24
 - under LabVIEW 26
 - under VEE 24
 - S**
 - Service and Support 55
 - Software Support 8
 - Specifications 51
 - System driver 27
 - System Requirements 8
 - T**
 - Technical Questions 55
 - Test Program 9
 - Trigger external 13
 - V**
 - VEE
 - Example Programs 25
 - ME Board Menu 25
 - Programming 24
 - User Objects 25

W

WDM driver 27

WrapAround Mode 21