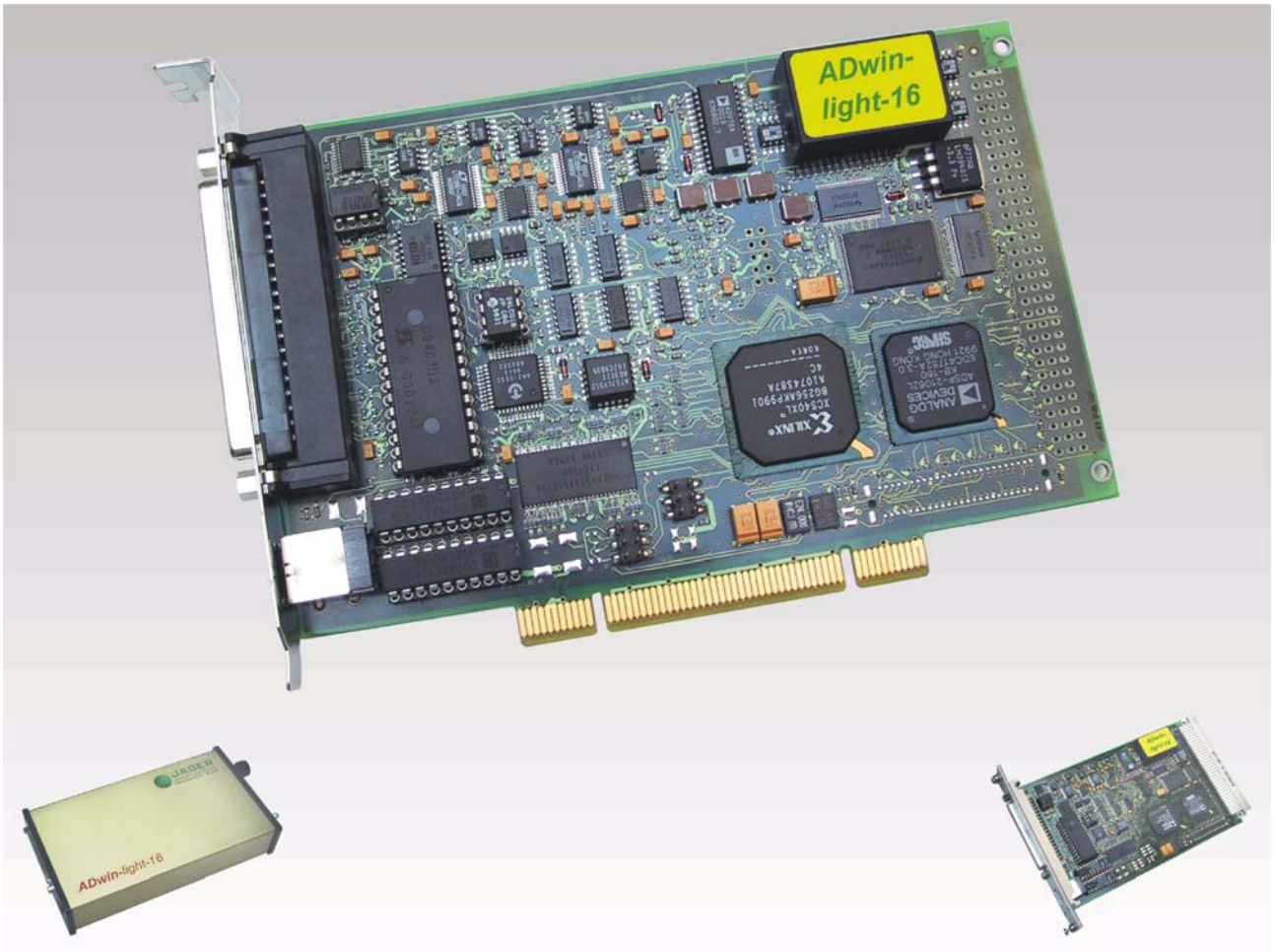
ADwin-light-16

Manual



ADwin-light-16 , manual version 2.3, October 2005

For any questions, please don't hesitate to contact us:

Hotline:	+49 6251 96320
Fax:	+49 6251 5 68 19
E-Mail:	info@ADwin.de
Internet	www.ADwin.de

JÄGER Computergesteuerte Messtechnik GmbH

Jäger Computergesteuerte
Messtechnik GmbH
Rheinstraße 2-4
D-64653 Lorsch
Germany

Table of contents

Typographical Conventions

"Warning" stands for information, which indicate damages of hardware or software, test setup or injury to persons caused by incorrect handling.

You find a "note" next to

– information, which have absolutely to be considered in order to guarantee an operation without any errors
– advice for efficient operation

"Information" refers to further information in this documentation or to other sources such as manuals, data sheets, literature, etc.

<C:\ADwin\ …>

File names and paths are placed in angle brackets and characterized in the font Courier New.

Program text

Program instructions and user inputs are characterized by the font Courier New.

Var_1

ADbasic source code elements such as `INSTRUCTIONS`, *`variables`*, `comments` and `other text` are characterized by the font `Courier New` and are printed in color (see also the editor of the ***ADbasic*** development environment).

Bits in data (here: 16 bit) are referred to as follows:

Bit No.	15	14	13	…	01	00
Bit value	2^{15}	2^{14}	2^{13}	…	2^1=2	2^0=1
Synonym	MSB	-	-	-	-	LSB

1 Information about this manual

This manual contains comprehensive information about the operation of the ***ADwin-light-16*** system. Additional information are available in

- the manual "***ADwin*** Driver Installation", which describes all interface installations for the ***ADwin*** systems.
  Start your installation with this manual.
- the description of the configuration program *ADconfig*. With it, you initialize the communication from the corresponding interface to your ***ADwin-light-16*** system.
- the manual ***ADbasic***, which contains all instructions for the compiler ***ADbasic*** and explains the principle of ***ADwin*** systems in particular.
- the description of the driver installation and command instructions for all well known development environments.

Please note:

To have your ***ADwin*** systems work properly, keep strictly to the information given in this documentation and in other mentioned manuals.

Qualified personnel

Programming, start-up and operation, as well as the modification of program parameters must be performed only by appropriately qualified personnel.

> *Qualified personnel are persons who, due to their education, experience and training as well as their knowledge of applicable technical standards, guidelines, accident prevention regulations and operating conditions, have been authorized by a quality assurance representative at the site to perform the necessary acivities, while recognizing and avoiding any possible dangers.*
> *(Definition of qualified personnel as per VDE 105 and ICE 364).*

Availability of the documents

This product documentation and all documents referred to, have always to be available and to be observed. For damages caused by disregarding the information in this documentation or in all other additional documentations, no liability is assumed by the company ***Jäger Computergesteuerte Messtechnik GmbH***, Lorsch, Germany.

Legal instructions

OEM products are mentioned without referring to possible patent rights, whose existence is not to be excluded.

Subject to change.

Hotline address: see inner side of cover page.

2 System description

2.1 *ADwin* system concept

ADwin systems guarantee fast and accurate operation of measurement data acquisition and automation tasks under real-time conditions. This offers an ideal basis for applications such as:

– very fast digital closed-loop control systems

– very fast open-loop control systems

– data acquisition with very fast online analysis of the measurement data

– monitoring of complex trigger conditions and many more

ADwin systems are optimized for processes which need **very short process cycle times** of one millisecond upto some microseconds.

System features

The ***ADwin*** system is equipped with analog and digital inputs and outputs, a fast processor (32-bit floating point signal processor) and local memory. The processor is responsible for the whole real-time processing in the system. The applications are running **independent** of the PC and its workload.

Processor

The processor of the ***ADwin*** system processes ***each*** **measurement value at once**.

In one cycle you can acquire the status of the inputs, process the status by the-help of any mathematical functions, and react to the results, even at very fast process cycle times of some microseconds. This results in a perfect and logical work sharing: The PC runs a program for visualizing of data, for input and operation of the processes, togeher with access to networks and data bases, while the processor of the ***ADwin*** system executes all tasks which require real-time in parallel.

Real-time operating system

The operating system for the DSP of the ***ADwin*** system has been optimized to reach the fastest response times possible. It manages parallel processes in a **multitasking** manner. Low priority processes are managed by time slicing. Requested high priority processes interrupt all low priority processes and are instantaneously and completely executed (preemtive multitasking). High priority processes are executed as time-controlled or event-controlled processes (external trigger).

Timing

The built-in **timer** is responsible for the precise calling of high priority processes. It has a resolution of 25 nanoseconds. The ***ADwin*** systems are characterized by an extremely short response time of only 300 nanoseconds during the change from a low to a high priority process. A continously running communication process enables a continous data exchange between the ***ADwin*** system and the PC even during applications in process. The communication has no influence on the real-time capability of the ***ADwin*** system, nevertheless, it is possible to exchange data at any time.

ADbasic

The real-time development tool ***ADbasic*** gives the opportunity to create time-critical programs for ***ADwin*** systems very easily and quickly. ***ADbasic*** is an **integrated development environment** under Windows with possibilities for online debugging. The habitual, easy-to-learn BASIC instruction syntax has been extended by many more functions, in order to get direct access to inputs and outputs as well as by functions for process control and communication with the PC.

Communication between *ADwin* system and PC

Interfaces

The ***ADwin*** system is connected to the PC via an **USB or Ethernet** interface. After power-up the ***ADwin*** system is booted from the PC via this interface. Afterwards the ***ADwin*** operating system is waiting for instructions from the PC which it will process.

Instruction processing

There are two kinds of instructions: On the one hand instructions, which transfer data from the PC to the ***ADwin*** system, for instance "load process", "start process" or "set parameter", on the other hand instructions which wait for a response from the ***ADwin*** system, for instance "read variables" or "read data sets". Both kinds of instructions are processed immediately by the ***ADwin*** system, which means immediate and full range responds. The ***ADwin*** system never sends data to the PC without request! The data transfer to the PC is always a response to an instruction coming from the PC. Thus, embedding the ***ADwin*** system into various programming languages and standard software packages for measurements is easily made, because they have only to be able to call functions and process the return value.

Software interfaces

Under Windows 95/98/NT/ME/2000/XP you can use a **DLL** and an **ActiveX** interface. On this basis the following drivers for **development environments** are available:
.NET, Visual Basic, Visual-C, C/C++, Delphi, VBA (Excel, Access, Word), TestPoint, LabVIEW / LabWINDOWS, Agilent VEE (HP-VEE), InTouch, DIAdem, MATLAB.

The easy, instruction-oriented communication with the ***ADwin*** system enables several Windows programs to access the same ***ADwin*** system in coordination at the same time. This is of course a great advantage when programs are developed and installed.

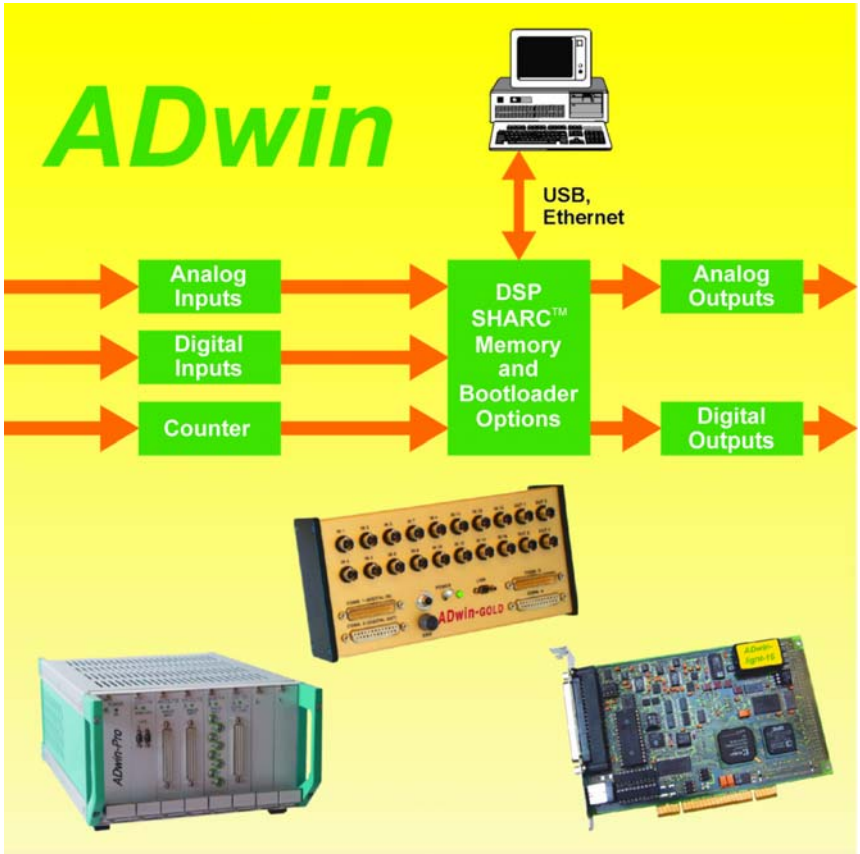


Fig. 1 – Concept of the ***ADwin*** systems

2.2 ADwin-light-16

Processor and memory

The ***ADwin-light-16*** system is equipped with the **32 bit signal processor** ADSP 21062 (SHARC) from Analog Devices with floating point and integer processing. It is responsible for the complete measurement data acquisition, online processing, and signal output and can instantaneously process - in combination with the A/D-converter - each measurement value with sample rates of up to 100 kHz.

The **on-chip memory with 256 kB** has a very short access time of 25 ns and is large enough to hold the complete ***ADwin*** operating system, the ***ADbasic*** process and all variables.

In order to get maximum access speed, all inputs and outputs are memory mapped in the external memory section of the DSP. For buffering larger quantities of data the DSP uses an **external memory** (SDRAM) of **8 MB**.

Analog inputs

In a 37-pin D-SUB socket there are **8 analog inputs** available, which are connected to a multiplexer, whose output signal is converted with a 16 bit analog-to-digital converter (ADC, see figure below).

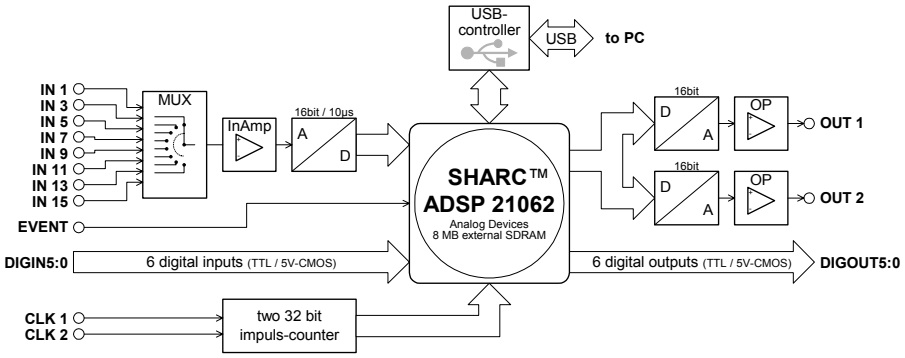


Fig. 2 – Functions diagram (with USB interface)

Analog outputs

ADwin-light-16 is equipped with **2 analog outputs** with 16 bit resolution and an output voltage range of -10 V ... +10 V. You can synchronize and calibrate the output of the voltage of all DACs per software. In order to smooth the output signal, it passes through a low pass filter with a cut-off frequency of f_c = 700 kHz.

Digital inputs and outputs

6 digital inputs and 6 digital outputs are available on the 37-pin D-SUB socket. The inputs and outputs are TTL-compatible.

Trigger input

ADwin-light-16 is equipped with a trigger input (EVENT, see also chapter 5.2 "Digital Inputs and Outputs"). Thus, processes can be triggered by a signal (trigger) and completely processed at once (see ***ADbasic*** manual, chapter "Processes in the ADwin Operating System").

Scope of delivery

The standard scope of delivery for ***ADwin-light-16***:

- ***ADwin-light-16*** system
- USB or Ethernet connecting cable, length 1.8 m
- ***ADwin***-CD-ROM
- Manual "Driver Installation"
- this hardware manual

The additional items for the type with external enclosure (*L16-EXT*) are:

- Power adapter: a slot metal sheet with power supply socket and PC-internal three-pole connecting cable
- power supply cable (between slot metal sheet and *L16-EXT*)

Designs

ADwin-light-16 is available as basic version with USB connection in different designs.

PC plug-in board (*L16-PCI*)

19" plug-in board (*L16-EURO*)

external enclosure (*L16-EXT*)

Compact-PCI plug-in board (*L16-cPCI*)

Fig. 3 – Designs

Types

The designs *L16-EURO* and *L16-EXT* can optionally be delivered with USB or 10/100 MBit Ethernet interface. The available types of the ***ADwin-light-16*** basic version are described in the following table.

Type	Interface	
	USB	Ethernet
PC plug-in board	*L16-PCI*	–
19" plug-in module (Euro)	*L16-EURO*	*L16-EURO-ENET*
Compact PCI plug-in module	*L16-cPCI*	–
External enclosure	*L16-EXT*	*L16-EXT-ENET*

Fig. 4 – Types of the ***ADwin-light-16*** basic version

Please take into account that the power supply for the different designs varies:

– +5 Volt for *L16-PCI*, *L16-EURO* and *L16-cPCI*

– +10 ... +18 Volt for *L16-EXT*

2.2.1 Ordering options (no later upgrading possible)

ADwin-light-16 can be equipped with the following options:

- ***L16-CO1***: **counter option** with a 32-bit up/down counter with four edge evaluation for incremental encoders.
- ***L16-DIO1***: add-on module with
  - **32 digital inputs/outputs** (programmable in groups of 8)
  - **CAN interface** (high speed)
  - **two 32 bit up/down counters** for impulse, period duration and duty cycle measurements as well as a four edge evaluation for the connection of incremental encoders.
- ***L16-Boot***: **Flash-EPROM bootloader** for standalone **operation without PC**. Can only be ordered in combination with an Ethernet interface.
- ***L16-Mount***: Kit for installation of the system on a DIN rail in a control cabinet.

Please take into account, that the counters of the add-on boards are not additionally available, but they replace the counters of the basic version. Therefore you cannot use the counters of different add-ons at the same time.

2.2.2 Accessories

ADbasic

- ***ADbasic***, real-time development tool for all ***ADwin*** systems
- cable connectors for an external power supply (for *L16-EXT* only)
- external power supply (necessary for notebook operation)

3 Operating Environment

The borad of the ***ADwin-light-16*** may only be operated in a closed casing (already given with *L16-EXT* design).

According to type and accessories (see chapter 2.2.1f, delivery options / accessories) the system can be operated in 19" enclosures, control cabinets or as a mobile system (e.g. in cars).

The ***ADwin-light-16*** **system must be earth-protected**, in order to
– build a ground reference point for the electronic
– conduct interferencing energy to earth.
Do connect the GND socket, which is internally connected with the ground reference point and the enclosure, via a short low-impedance solid-type cable to the central earth connection point of your installation.

Galvanic connection

The types with USB interface have a galvanic connection to the PC via USB and possibly also via power supply.

The types with Ethernet interface have their data lines galvanically isolated, but the ground potentials are connected, because the shielding of the Ethernet connector (RJ-45) is connected with GND.

Excluding transient currents

Transient currents, which are conducted via the enclosure or the shielding, have influence on the measured signal.
If you want to prevent transient currents, please make sure that the shielding is fully operative. Take measures for bleeding off interferences, such as earthing the shielding close to the entry into the control cabinet. The more frequently you earth the shielding on its way to the machine the better the shielding will operate.

Use cables with shielding on both ends for **signal lines.** Here too, you should reduce the bleed off of interferences via the enclosure by using screen clips.

Protection low voltage

The ***ADwin-light-16*** system is internally operated with a voltage of +5 V and ±15 V against GND and thus is not life-threatening. For operation with an external power supply, the instructions of the manufacturer apply.

Ambient temperature

ADwin-light-16 is designed for operation in dry rooms. The installation environment (PC or 19" rack) shall have an ambient temperature of +5 ... +50 °C, and a relative humidity of 0 ... 80 % (no condensation, see also Annex).

Chassis temperature

The temperature of the casing (surface temperature) of the type *L16-EXT* must not exceed +55 °C, even under extreme operating conditions - e.g. in a control cabinet or if the system is exposed to the sun for a longer period of time. Otherwise, you risk damages to the device or the output of not-defined data (values) which can cause damages to your measurement device under unfavorable circumstances.

4 Start-up of the Hardware

Do not connect any cables to the ***ADwin-light-16*** system **on start-up** before you have executed the **following steps**:

1. Software installation / hardware installation in the PC or 19“ rack.

   Follow the instructions in the manual: "***ADwin*** Driver Installation".
2. Read chapter 5 "Inputs and Outputs" in this manual
3. Only now connect the signal lines to the inputs and outputs.

Notes

Avoid direct contact with uninsulated parts in order to protect them against electrostatic discharges.

Reliable power supply

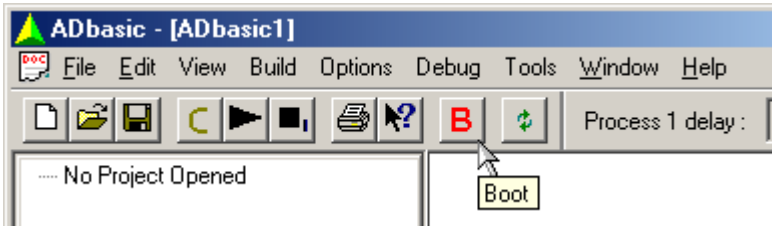
Please pay attention that a reliable power source is supplied.
This concerns the PC (standard scope of delivery), otherwise also the external power supply, if operated in a car, the battery voltage.

If using current-limiting power supplies, please pay attention the current demand on power-up which can be a multiple of the idle current. You find more detailed information in the Technical Data (Annex).

In case of power failure all unsaved data are lost. Not-defined data (values) can under unfavorable circumstances cause damages to your measurement device.

Check Data Connection

Start ***ADbasic*** and boot the ***ADwin*** system by clicking the boot button B.



The display in the status line: "`ADwin is booted`" shows that the operating system has been loaded appropriately and that ***ADbasic*** can establish a connection to the ***ADwin*** system.

***ADbasic* programs**

Programming the ***ADwin*** systems is described more detailed in the ***ADbasic*** manual.
Start with the programming examples in the ***ADbasic*** Tutorial.

5 Inputs and Outputs

Connections

The ***ADwin-light-16*** system is equipped with a connecting socket for USB or Ethernet as well as with a 37-pin D-SUB socket. You will find on the D-SUB socket (pin assignment next page) as follows:

– 8 analog inputs

– 2 analog outputs

– 6 digital inputs and outputs each

– 1 digital trigger input

– 2 impulse/event counters with 32 bit

– Output for power supply +5V; *L16-PCI* and *L16-cPCI* also ±12V

The design *L16-EXT* has an additional GND socket (see earth protection, page 7), a power input socket and a manual on/off switch.

All inputs and outputs may only be operated according to the specifications given (see Annex A-1: Technical Data). In case of doubt, ask the manufacturer of the device to which you want to connect the ***ADwin-light-16*** system.

Open inputs can cause errors – above all in an environment which is not free of any interferences. For your own safety, set unused inputs as close as possible to the D-SUB socket on a defined level (e.g. GND). Separate these inputs also from open-ended lines.
Exception to this is the event input, which already has an internal pull-up resistor (10 kΩ).

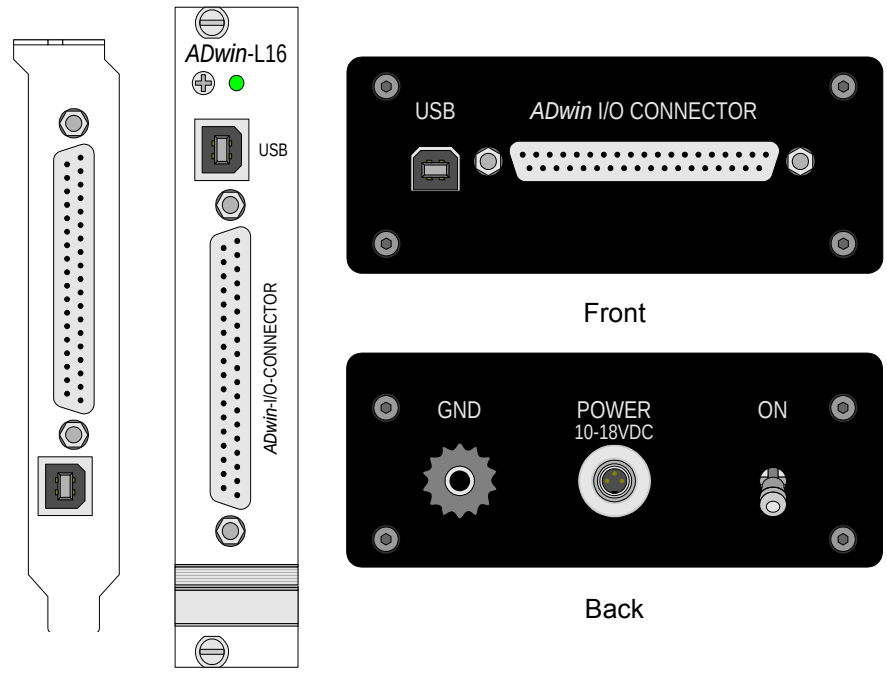


Fig. 5 – Connections at the ***ADwin-light-16*** system

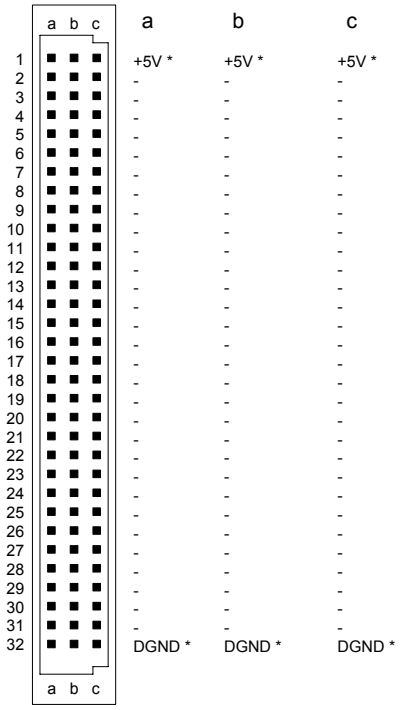


Fig. 6 – *L16-EURO* VGA pin socket female (power supply)

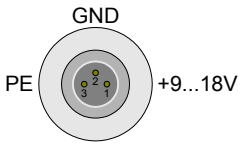
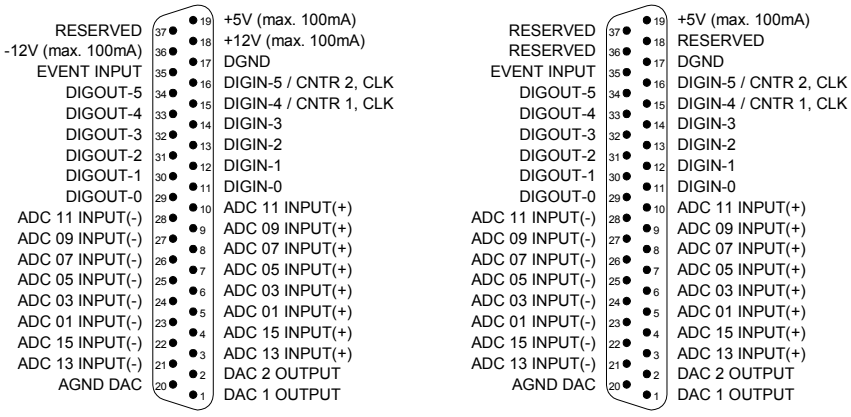


Fig. 7 – *L16-EXT* power connector (male)



L16-PCI and *L16-cPCI* *L16-EURO* and *L16-EXT*

Fig. 8 – Pin assignment inputs/outputs (female)

Standard instructions

For fast and easy programming there are standard instructions available in the compiler ***ADbasic***, which enable a user to easily measure or output data (see also ***ADbasic*** manual). Use other instructions (such as direct register access) only if extremely time-critical or special tasks require to do so. (see also chapter 11).

More detailed information about the analog as well as the digital inputs and outputs can be found in the following chapters.

5.1 Analog Inputs and Outputs

Earth protection

The design *L16-EXT* has to be earth-protected, in order to execute measurements without interferences. For this, connect the GND socket via a low impedance solid-type cable to the central earth connection point of your installation. When using the design PCI or EURO, the earth protection is made via PC or the 19" rack.

At the *L16-EXT* system, the enclosure is connected to the protective earth conductor of the PC via the GND-line of the power supply cable as well as via the GND-line of the USB cable.

5.1.1 Inputs

Multiplexer

The system has 8 analog measurement inputs, which are connected to the 16-bit analog-to-digital converter (ADC) via one multiplexer. The inputs are solely odd-numbered (ADC 01, ADC 03, ... ADC 15), which has to be considered upon programming.

Differential

The analog inputs are differential. For each of the measurement channels there is one plus and one minus input; between them the voltage difference is measured (pay attention to the potential of the input lines).

16-bit measurement

The signal at the multiplexer output is converted by a 16-bit analog-to-digital converter (ADC) (see Fig. 2 – Functions diagram (with USB interface)). The conversion time is 10 µs at a resolution of 305 µV.

Complete measurement

The instruction `ADC( )` executes a complete measurement with an ADC on one analog input. Thus, this instruction considers for instance the settling time of the multiplexer and assures perfect measurements (see also ***ADbasic*** manual).

5.1.2 Outputs

DAC instruction

The standard instruction `DAC(number,value)` checks each of the values exceeding or falling below of the 16-bit value range. If the value is in the 16-bit value range, the indicated value is output on the output *number*. If it is not, the maximum or minimum value is output (see also ***ADbasic*** manual).

5.1.3 Calculation Basis

Voltage range

The voltage range of the ***ADwin-light-16*** system at the analog inputs and outputs is –10 V to +10 V (bipolar 10 V).

Allocation of digits and voltage

The 65,536 (2^{16}) digits are allocated to the corresponding voltage ranges of the ADCs and DACs insofar that

– 0 (zero) digits correspond to the maximum negative voltage and

– 65,535 digits correspond to the maximum positive voltage

The value for 65,536 digits, exactly 10 Volt, is just outside the measurement range, so that you will get a maximum voltage value of 9.999695 Volt for the 16-bit conversion.

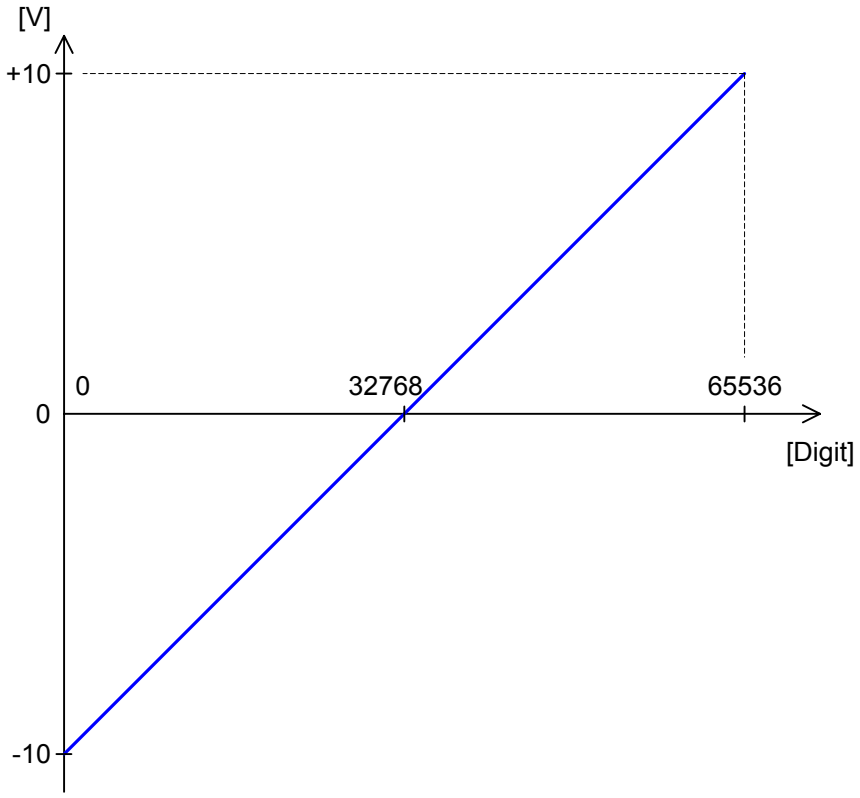


Fig. 9 – Zero offset in the standard setting of bipolar 10 Volt

In the bipolar settings you will get a zero offset.

For the voltage range of –10 V ... +10 V applies U_{OFF} = –10 V

Least Significant Bit

The quantization level U_{LSB} is the smallest digitally displayable voltage difference and is equivalent to the voltage of the least significant bit (LSB). The U_{LSB} is equivalent to the formula: 20 V / 2^{16} = 305.175 µV.

The measured 16-bit value of the ADC is returned to the lower word of the binary cell. Here you must also find the DAC value to be output.

Bit	31...16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
32-bit memory	upper word	16-bit value of the ADC/DAC in the lower word															

Conversion Digits ⇆ Voltage

DAC

For a DAC:

$$U_{OUT} = Digits \cdot U_{LSB} + U_{OFF}$$

$$Digits = \frac{U_{OUT} - U_{OFF}}{U_{LSB}}$$

ADC

For an ADC:

$$\text{Digits} = \frac{U_{IN} - U_{OFF}}{U_{LSB}}$$

$$U_{IN} = \text{Digits} \cdot U_{LSB} + U_{OFF}$$

Tolerance ranges

Slight deviations regarding the calculated values may be within the tolerance range of the individual component. Two kinds of deviations are possible, which are indicated in this manual (in LSB):

– The integral non-linearity (INL) defines the maximum deviation from the ideal straight line of the conversion characteristics curve, covering the whole input voltage range.

INL

– The differential non-linearity (DNL) defines the maximum deviation from the ideal quantization level.

DNL

5.2 Digital Inputs and Outputs

Digital inputs/outputs

6 digital inputs (DIGIN 00 … DIGIN 05) and 6 digital outputs (DIGOUT 00 … DIGOUT 05) are available on the 37-pin D-SUB socket. You will find the pin assignment in figure 8 on page 10.

The inputs DIGIN 04 and DIGIN 05 can be used as counter inputs at the same time (this is not possible when using a DIO1-add-on) and can optionally be used for both purposes.

The digital inputs and outputs are TTL-compatible and not protected against overvoltage.
Do not use connections marked as "RESERVED". They are reserved for upcoming changes or expansions and can cause damages to your system if you do not pay attention to this fact.

Trigger input (EVENT)

The ***ADwin-light-16*** system is equipped with an external trigger input (EVENT). An external signal (trigger) with rising edge can start processes, which are processed completely and immediately (see ***ADbasic*** manual, chapter "Structure of an ***ADbasic***-Program").

5.3 Impulse/Event Counter

The ***ADwin-light-16*** system is equipped with 2 impulse/event counters with 32-bit, which you can configure or read out both together or each individually.

With the options *L16-CO1* and *L16-DIO1* the counters described here are replaced by other counters. You will find the corresponding description in chapter 7 "CO1 Counter Add-On" or chapter 8 "DIO1 Add-On".

5.3.1 Hardware

The figure shows the design of a single counter.

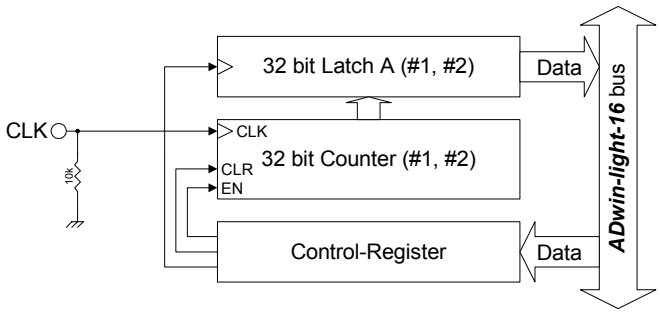


Fig. 10 – Block diagram of the impulse/event counter

The counters are externally clocked that means they increase their counter values by incrementing at each positive edge at the clock input (CLK). Both counters have a latch A, into which the counter value can be latched (by programm instruction) for read out.
The counters are controlled with special ***ADbasic*** instructions via the control register. The instructions are described in Fig. 11 – Counter instructions - short reference (below) and in the ***ADbasic*** manual (or online help).

Setting inputs

The clock inputs are on pins 15 and 16 (see Fig. 8 – Pin assignment inputs/outputs (female), page 10); for the correct function TTL compatible signals are necessary. More details and limit values can be found in the Technical Data in the Annex. Both inputs can optionally be used as digital signal input (see also chapter 5.2).

The inputs of the impulse/event counters have a pull-down resistor. Nevertheless open-ended inputs can cause errors in an environment which is not free of interferences. Therefore set the unused inputs to a defined level (e.g. GND).

5.3.2 Software

Include file

The counters are easily programmed by using ***ADbasic*** instructions. The instructions are part of an include file which must be included at the beginning of a program:

```
#INCLUDE ADWL16.INC
```

The instructions for both counters are shortly illustrated in the following table and more detailed in the ***ADbasic*** manual or online help. You can configure each counter individually or both counters together.

Counter nº Bit	2 1	1 0	Comment
CNT_CLEAR( )	0	0	no effect
	1	1	clear counter*
CNT_ENABLE( )	0	0	disable counter
	1	1	enable counter (pay attention to running counters)
CNT_LATCH( )	0	0	no effect
	1	1	copy counter value into latch A *
CNT_READLATCH(#)			read latch A (# = counter-nº 1, 2)
CNT_READ(#)			copy counter value into latch A and read it (# = counter nº1, 2)
* these functions are reset after being executed. All other functions are reset by the opposing function..			

Fig. 11 – Counter instructions - short reference

Please configure a counter according to the following order:

Sequence of instructions

1. disable specified counter (CNT_ENABLE)

   The instruction CNT_ENABLE always accesses all counters. Even if you want to change the status (disabled/enabled) of only one counter, you also have to configure the counters whose status shall remain unchanged.

2. clear counter (CNT_CLEAR)
3. enable counter(CNT_ENABLE)

For further processing of the counter value in the ***ADbasic*** program, transfer the value into latch A and read it from there.

5.3.3 Evaluation of the counter contents

The binary counters generate 32 bit values. Distinguish clearly between the evaluation of these binary values (e.g. differences) and the screen representation as decimal numbers.

To correctly evaluate the counter contents you need its original 32 bit values, especially with calculating differences. This is guaranteed only by use of ***ADbasic*** variables of type INTEGER or LONG.

unchanged bit pattern

The representation of 32 bit values in ***ADbasic*** often leads to confusion, because the signless counter value is shown as signed decimal number (see circle below). Consequently a transition between positive and negative range of numbers is shown on the screen, which yet is of no importance to the evaluation of the counter contents.

For completion the following describes the interpretation of a binary value:
The most significant bit (MSB) is interpreted as sign.
With a positive sign, the following 31 bits are directly interpreted as numerical value; binary and decimal value are similar.
With a negative sign, the 31 bits are first inverted, one added, and then interpreted as numerical value (2's complement); thus, negative decimal numbers have an absolute value different to the corresponding binary value.

Calculate differences only with integer numbers (INTEGER, LONG).

"Lap overflow"

For programming please remember that an "lap overflow" between the read out of two counts - i.e. the current counter value "laps" the last counter value which has been read - is not registered.

Such a lap overflow occurs after some 3½ minutes with an input frequency of 20 MHz or after more than 14 minutes with 5 MHz.

Circle

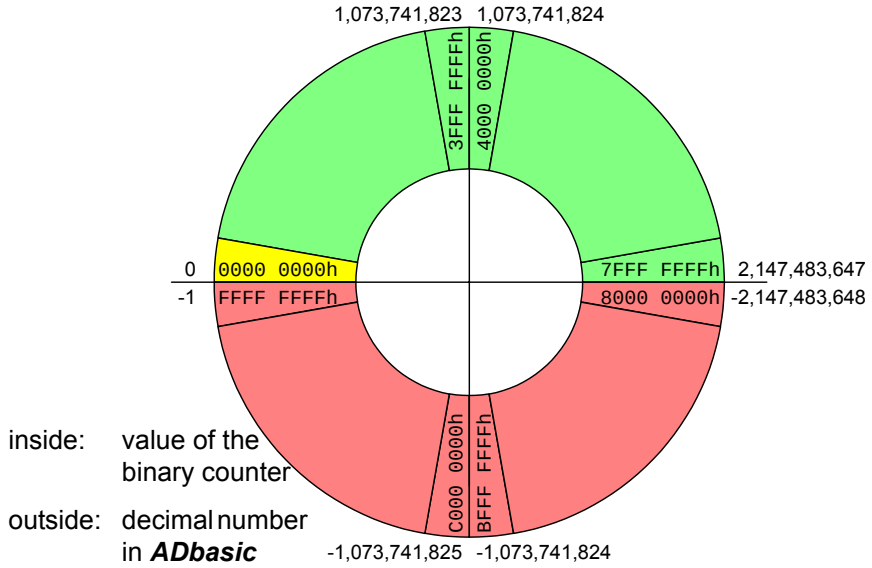


Fig. 12 – Circle model for interpretation of counter values

5.4 Time-critical tasks

Time-critical tasks

For extremely time-critical tasks you can use instructions with which you have direct access to control and data registers of the hardware (see ***ADbasic*** manual or online help). These registers can be found in the memory address area of the ADSP (memory mapped). These instructions also allow to optimize the program structure.

Contrary to the standard instructions `ADC()` and `DAC()` the instructions for direct access do not have any test routines. Before you use them you should have good knowlegde about programming and time and function sequences in an analog-to-digital converter, because you are now programming closely to the hardware.

Analog inputs and outputs

`ADC()`

Execute the following ***ADbasic*** instructions instead of the standard instruction `ADC()` according to the following order:

Program structure

```
SET_MUX()
...                         'wait for settling time
START_CONV()
WAIT_EOC()                  'wait for end of conversion
READADC()
```

It is important to set a sufficient time-delay using additional programming instructions between the instructions `START_CONV()` and `SET_MUX()`, in order to consider the multiplexer settling time (see also ***ADbasic*** manual: "Instruction Reference").

Use the waiting times shown below for instance for computing operations and thus, save computing time:

– Settling time of the multiplexer: At a maximum voltage jump of 20 V it is 6.5 µs.

– Conversion time of the 16 bit ADC: 10 µs.

Hardware addresses of the control and data registers

If you use the instructions PEEK and POKE ((see ***ADbasic*** manual or online help) you can directly access the control and data registers. This may accelerate the processing of the program, e.g.:

– a measurement can be executed very fast.

– you can write very quickly into one or more DAC registers, and you can synchronously start the output.

Please make sure that the calculated analog outputs values are within the range limits.

The hardware addresses of the registers can be found on the following page, grouped as analog inputs, analog outputs, digital inputs/outputs and counters.

Please take into account that some registers have an influence on several processes.

ADC

DAC

Address [HEX]	Function	Bit nº												Comment
		31-16	15-10	9	8	7	6	5	4	3	2	1	0	
20 40 00 00	set multiplexer to input channel (ADC 01... ADC 15)	-	-	-	-	-	-	0	0	0	n	n	n	"nnn" binary = 0...7 decimal; selected channel = nnn*2 + 1
20 40 00 10	start conversion: ADC#1	-	-	-	-	-	-	-	1	1	1	1	s	s = 0 : start conversion s = 1 : no effect
20 40 00 20	conversion status(EOC): ADC#1	-	-	-	-	-	-	-	-	-	-	-	e	e = 0 :end conversion e = 1 : conversion is running
20 40 00 30	read register: ADC#1	-	x	x	x	x	x	x	x	x	x	x	x	x : result of the conversion
20 40 01 00	read register and start conversion: ADC#1	-	x	x	x	x	x	x	x	x	x	x	x	

Fig. 13 – ADC hardware addresses of the control and data registers

Address [HEX]	Function	Bit nº												Comment
		31-16	15-10	9	8	7	6	5	4	3	2	1	0	
20 40 00 10	start conversion: all DAC synchronously	-	-	-	-	-	-	-	1	1	s	1	1	s = 0 : start conversion s = 1 : no effect
20 40 00 50	write only to the register: DAC #1	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted
20 40 00 60	write only to the register: DAC #2	-	x	x	x	x	x	x	x	x	x	x	x	
20 40 02 00	write to the register and start conversion immediately: DAC #1	-	x	x	x	x	x	x	x	x	x	x	x	
20 40 02 10	write to the register and start conversion immediately: DAC #2	-	x	x	x	x	x	x	x	x	x	x	x	

Fig. 14 – DAC hardware addresses of the control and data registers

Address [HEX]	Function	Bit nº								Comment
		31:16	15:6	5	4	3	2	1	0	
20 40 00 B0	input register DIGIN-05...DIGIN-00	-	-	x	x	x	x	x	x	x : digital value read in
20 40 00 C0	output register DIGOUT-05 ... DIGOUT-00	-	-	x	x	x	x	x	x	x : digital value to be output
20 40 00 C4	DIGOUT Bit-SET-Register	-	-	0	0	0	0	0	0	no effect
		-	-	1	1	1	1	1	1	set bit
20 40 00 C8	DIGOUT Bit-CLEAR-Register	-	-	0	0	0	0	0	0	no effect
		-	-	1	1	1	1	1	1	clear bit

Fig. 15 – DIO hardware addresses of the control and data registers

Address [HEX]	Function	Bit nº 31:16	15:6	5	4	3	2	1	0	Comment
20 40 02 04	contents of latch A, counter #1	x	x	x	x	x	x	x	x	x : latched counter value
20 40 02 14	contents of latch A, counter #2	x	x	x	x	x	x	x	x	x : latched counter value
20 40 03 00	enable/disable counter (= `CNT_ENABLE()`)	-	-	-	-	-	-	0	0	counter disabled
		-	-	-	-	-	-	1	1	counter enabled
20 40 03 10	clear counter(= `CNT_CLEAR()`)	-	-	-	-	-	-	0	0	no effect
		-	-	-	-	-	-	1	1	clear counter *
20 40 03 20	latch counter (= `CNT_LATCH()`)	-	-	-	-	-	-	0	0	no effect
		-	-	-	-	-	-	1	1	latch counter value into latch A *
* The bits are reset after the function has been executed. All other functions are reset by the opposing function.										

Fig. 16 – Counter hardware addresses of the control and data registers

6 Calibration

The two digital-to-analog (DAC) and the analog-to-digital (ADC) converter of the ***ADwin*** system have been calibrated in factory. In accordance with the regulations for keeping the measurement accuracy in your field of application, the systems must be calibrated in regular time intervals.

Calibration is made via software. You will find the program <L16Calib.exe> under the path <C:\ADwin\Tools\ADwin-L16> (at standard installations).

You need the following tools for the calibration:

– a digital multimeter (DMM) with a resolution of 30 µV.

– a reference voltage source with a resolution of 30 µV. Optionally you connect DAC 1 to ADC 01(+), DAC 2 to ADC 03(+) and AGND DAC with ADC 01(-) and ADC 03(-), for instance in form of a test connector. You need these connections also for the calibration diagram.

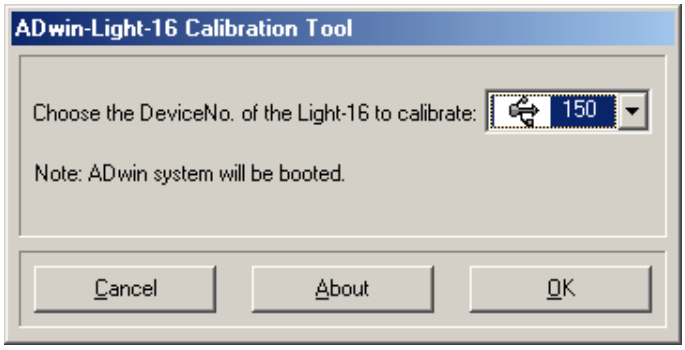
– connection cables from the inputs/outputs to the reference voltage source and to the measurement device.

Step 1

Connect your ***ADwin-light-16*** system with the PC and configure it with the program <ADconfig.exe>.

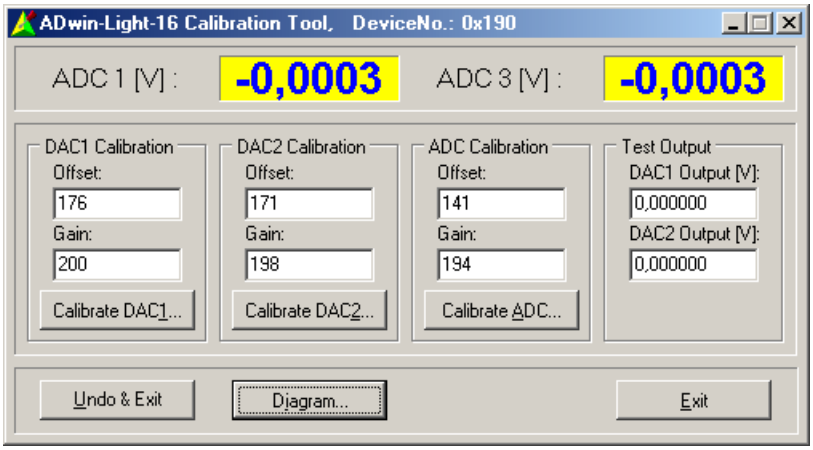
Step 2

Start the **calibration program** <L16Calib.exe>. The window "ADwin-light-16 Calibration Tool" appears.



Select the device number of the system to be calibrated and confirm by pressing "OK".
You will get a warning, if you haven't selected an ***ADwin-light-16*** system or if you have selected one with an older firm ware version. You can ignore the warning with "YES" or return to the previous windows with "NO".

An **overview window** appears. In the header the selected device number is displayed.



The upper field shows the current measurement values at the inputs ADC 01 and ADC 03. Below you will find the calibration settings for `Offset` and `Gain` of both DACs and the ADC; there you can directly enter values. You start calibration of the relevant converter with "`Calibrate …`"
At the right side ("Test Output") you can enter voltage values in the fields DAC1 and DAC2, which will automatically be output on the corresponding outputs.

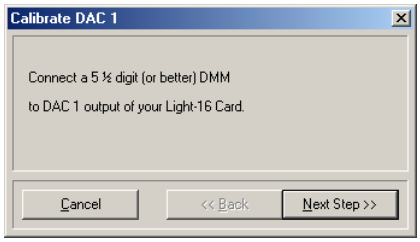
All settings you have made are automatically saved.
In the lower line you can undo all inputs with "`Undo & Exit`" and leave the calibration program. "`Diagram`" displays in a graph the accuracy of the current calibration setting. If you leave the program with "`Exit`" all settings remain.

Calibrate the converters in the order you like (only with reference voltage source). The calibration of a converter is effected in 3 steps; you can switch between the windows of the steps by using the forward/backward buttons. Calibration is also possible without reference voltage source, but it will not be so precise. Calibrate first the DAC1 and DAC2, and then the ADC.

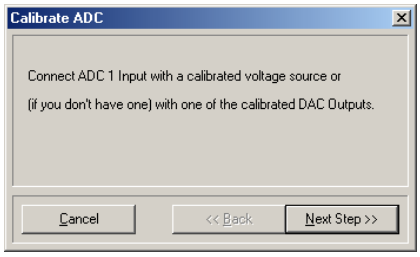
Step 3

The 3 levels for **calibrating a converter** are described below, for the DAC in the left column and for the ADC in the right column.

1. Connect the external device (DMM/voltage source)
   Select the corresponding key "`Calibrate ...`" for calibrating a converter; the first window appears.:



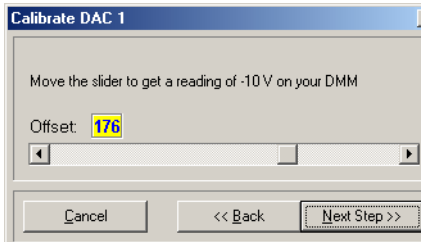
Connect a DMM with the pins AGND DAC and DAC1 or DAC2.



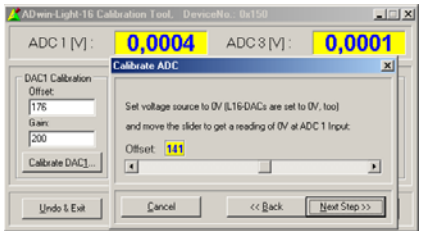
Connect the voltage source (or a DAC output) with the inputs ADC 01 or ADC 03.

Please note Fig. 8 – Pin assignment inputs/outputs (female).
Select "`Next Step >>`".

2. Setting th offset



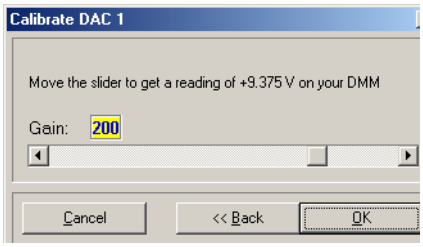
Adjust the offset value at the scrollbar in such a manner that your digital multimeter displays -10 V.



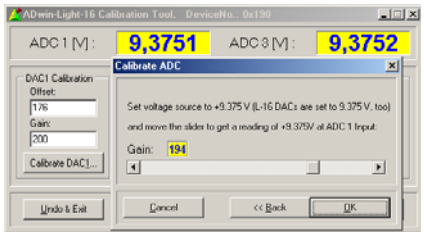
Set your voltage source to 0 V (setpoint). The setting of the ADC to this value is made automatically.
Adjust the offset value at the scrollbar in such a manner that the setpoint at the ADC 01 is displayed in the overview window.

Select "`Next Step` >>".

3. Set gain



Adjust the offset value at the scroll-bar in such a manner that your digital multimeter displays -10 V.

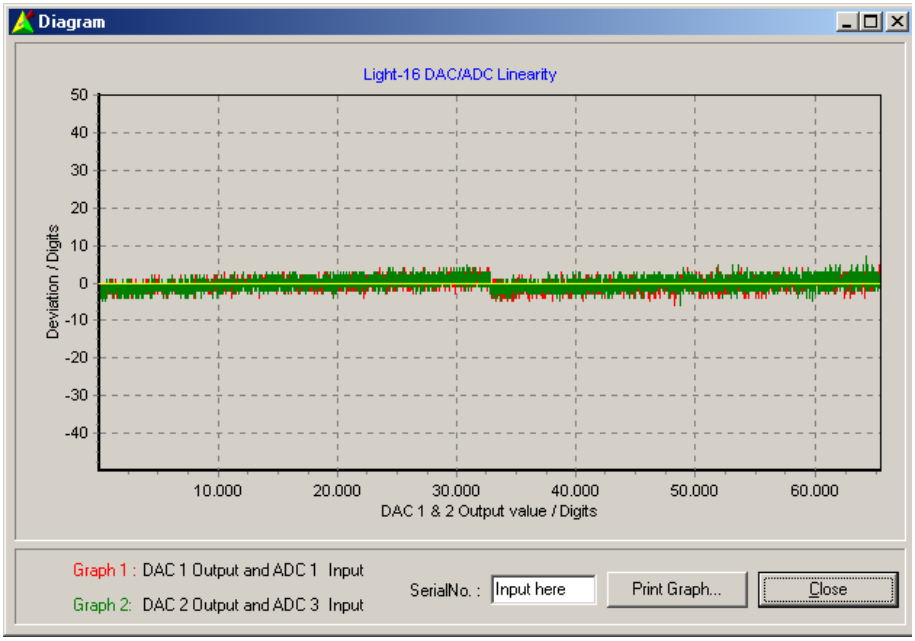


Set your voltage source to 9,375 V (setpoint). The setting of the ADC to this value is made automatically. Adjust the offset value at the scroll-bar in such a manner that the set-point at the ADC 01 is displayed in the overview window.

The calibration for this converter has finished. Select "0K". Repeat step 3 for the other converters if necessary.

Step 4

With a diagram (button `Diagram` in the overview window) you can check the **accuracy** of the calibration.
Connect first the output DAC 1 with the input ADC 01 as well as the output DAC 2 with the input ADC 03.



The program outputs the values 0...65,535 digits on both DACs, compares them to the measured input values and displays the deviation in graphs. Graph 1 (red) for DAC 1 / ADC 01 and graph 2 (green) for DAC 2 / ADC 03. The deviation should be smaller than 5 digits.

You can print the graph with `Print Graph` (a color printer is recommended). To do so, enter the serial number of your ***ADwin*** system, so that you can allocate the printout later. On the printout you will also find the calibration settings and the date of print.

With `Close` you return to the overview window.

Step 5

The calibration is finished.

7 CO1 Counter Add-On

The counter add-on CO1 (option *L16-CO1*) provides a 32-bit up/down counter with four edge evaluation and replaces the counter of the basic version (Fig. 17 – Block diagram of the L16-CO1 counter add-on shows the design of the counter).

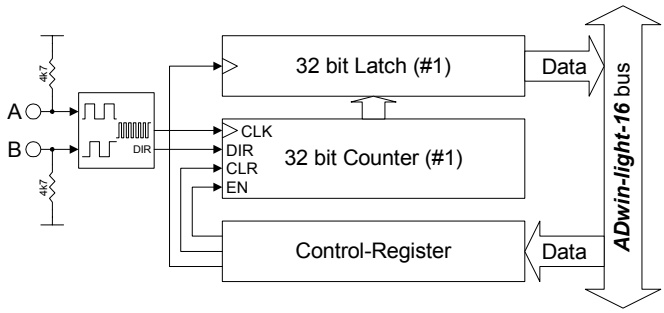


Fig. 17 – Block diagram of the *L16-CO1* counter add-on

7.1 Hardware

The counter is **externally clocked** and has a four edge evaluation for the connection of an encoder. The counter is read out via latch A, the control is made with ***ADbasic*** instructions via a control register (see "CO1 instructions, short reference" on page 23).

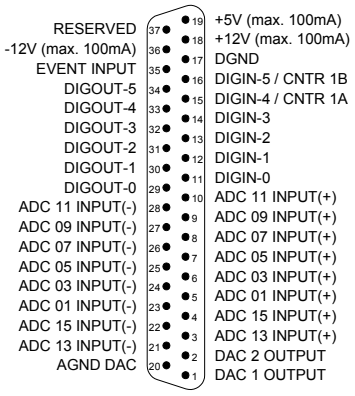
The four edge evaluation converts the digital input signals A and B (which are 90 degrees off-phase) into a clock (CLK) and direction signal (DIR) for the counter. Here a clock signal is generated from each edge of the A and B signals. The count direction (DIR) is determined by the order of the rising and falling edges of these signals.

Setting inputs

The clock inputs A and B are on the pins 15 and 16 of the D-SUB socket (see pin assignment below); TTL-compatible signals are necessary for the correct function. More details and limit values can be found in the Technical Data in the Annex.

Both inputs can be used optionally as digital signal input (see also chapter 5.2).

Although all inputs of the CO1 add-on have a pull-down resistor, open-ended inputs can cause errors in an environment which is not free of interferences. Therefore set unused inputs on a defined level (e.g. GND).



L16-PCI and *L16-cPCI* *L16-EURO* and *L16-EXT*

Fig. 18 – Pin assignment of *L16-CO1*

7.2 Programming

Include file

The CO1 add-on is easily programmed by using ***ADbasic*** instructions. The instructions are part of an include file which must be included at the beginning of a program:

```
#INCLUDE ADWL16.INC
```

The instructions for the CO1 add-on are shortly illustrated in the following table and more detailed in the ***ADbasic*** manual or online help.

Counter nº / Bit		Comment
CNT_CLEAR( )	0	no effect
	1	clear counter*
CNT_ENABLE( )	0	disable counter
	1	enable counter (pay attention to running counters)
CNT_LATCH( )	0	no effect
	1	copy counter value into latch A *
CNT_READLATCH(#)		read latch A (# = counter nº 1)
CNT_READ(#)		copy counter value into latch A and read it (# = counter nº 1)
* The bits are reset after the function has been executed. All other functions are reset by the opposing function.		

Fig. 19 – CO1 instructions, short reference

Please evaluate the counter contents only with variables of type INTEGER or LONG, above all when you want to evaluate differences or the count direction (see also page 15).

The count direction (up/down) can reliably be derived from the

sign of the difference: [new counter value] minus [old counter value]

and not from the *comparison* of the counter values.

For extremely time-critical tasks you can use instructions with which you have direct access to control and data registers of the counter. In the table the corresponding hardware addresses are illustrated.

The hardware addresses of the CO1 counters are identical with or replace those of the basic counter version.

Address [HEX]	Function	Bit number 31:01	Bit number 00	Comment
20 40 02 04	contents of latch A, counter #1	x	x	x : latched counter value
20 40 03 00	enable counter CNT_ENABLE( )	-	0	disable counter
		-	1	enable counter (pay attention to running counters)
20 40 03 10	clear counter CNT_CLEAR( )	-	0	no effecft
		-	1	clear counter*
20 40 03 20	latch counter CNT_LATCH( )	-	0	no effect
		-	1	latch counter*
* The bits are reset after the function has been executed. All other functions are reset by the opposing function.				

Fig. 20 – CO1 hardware addresses of the control and data registers

8 DIO1 Add-On

With the DIO1 add-on you are additionally provided with:

– 32 digital inputs/outputs (programmable in groups of 8)

– Two 32 bit up/down counters for impulse, period duration and duty cycle measurements as well as a four edge evaluation for connection of incremental encoders.
Inputs can be set to single-ended or differential.

– CAN interface (high-speed)

The block diagram shows the basic functions of an L16 system with the additional functions of the DIO1 add-on (as USB version).

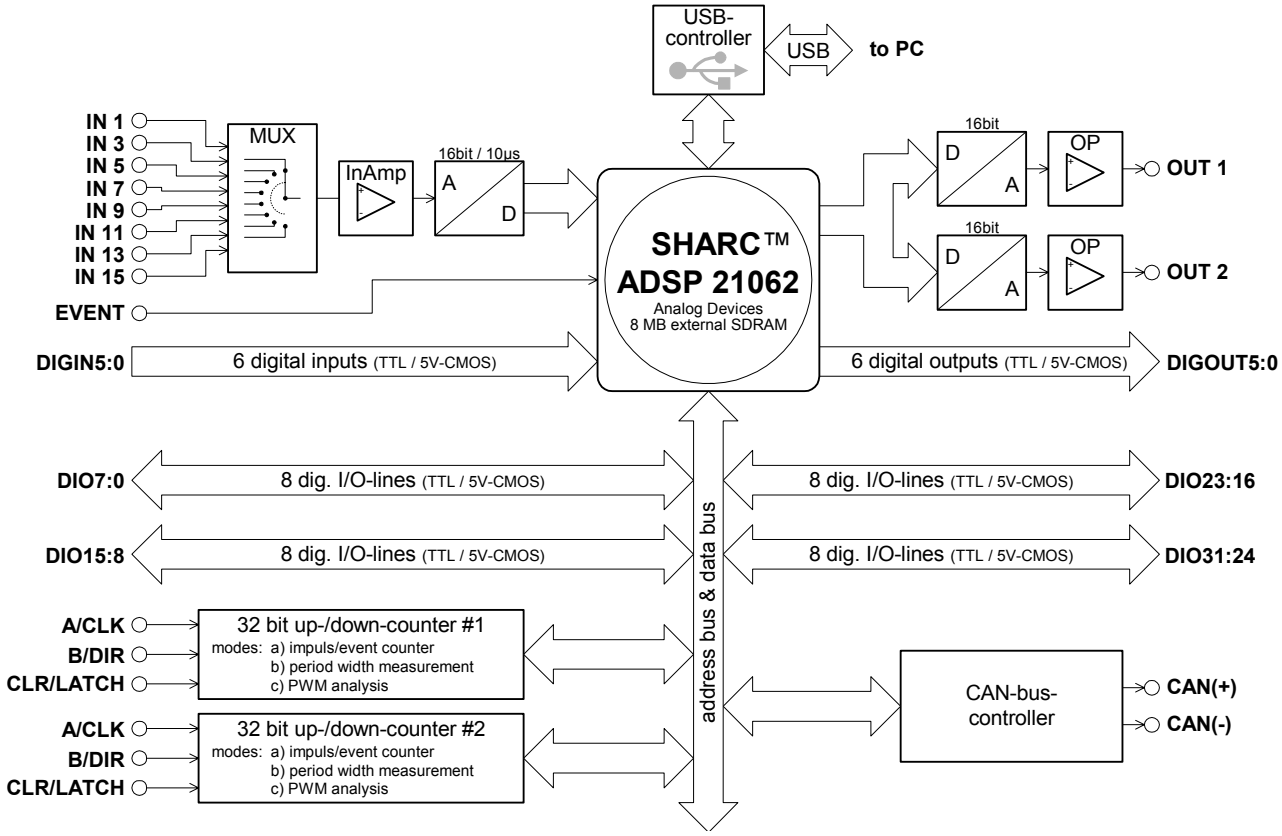


Fig. 21 – Block diagram of *L16-DIO1* (with USB interface)

The counters of the basic version are replaced by the DIO1 counters. The digital inputs and outputs remain.

The pin assignment at the connection "***ADwin*** I/O-CONNECTOR" is similar to the basic version, except one difference: The pins 15/16 - in the basic version each with double functions - are now solely used as DIGIN-04 and DIGIN-05.

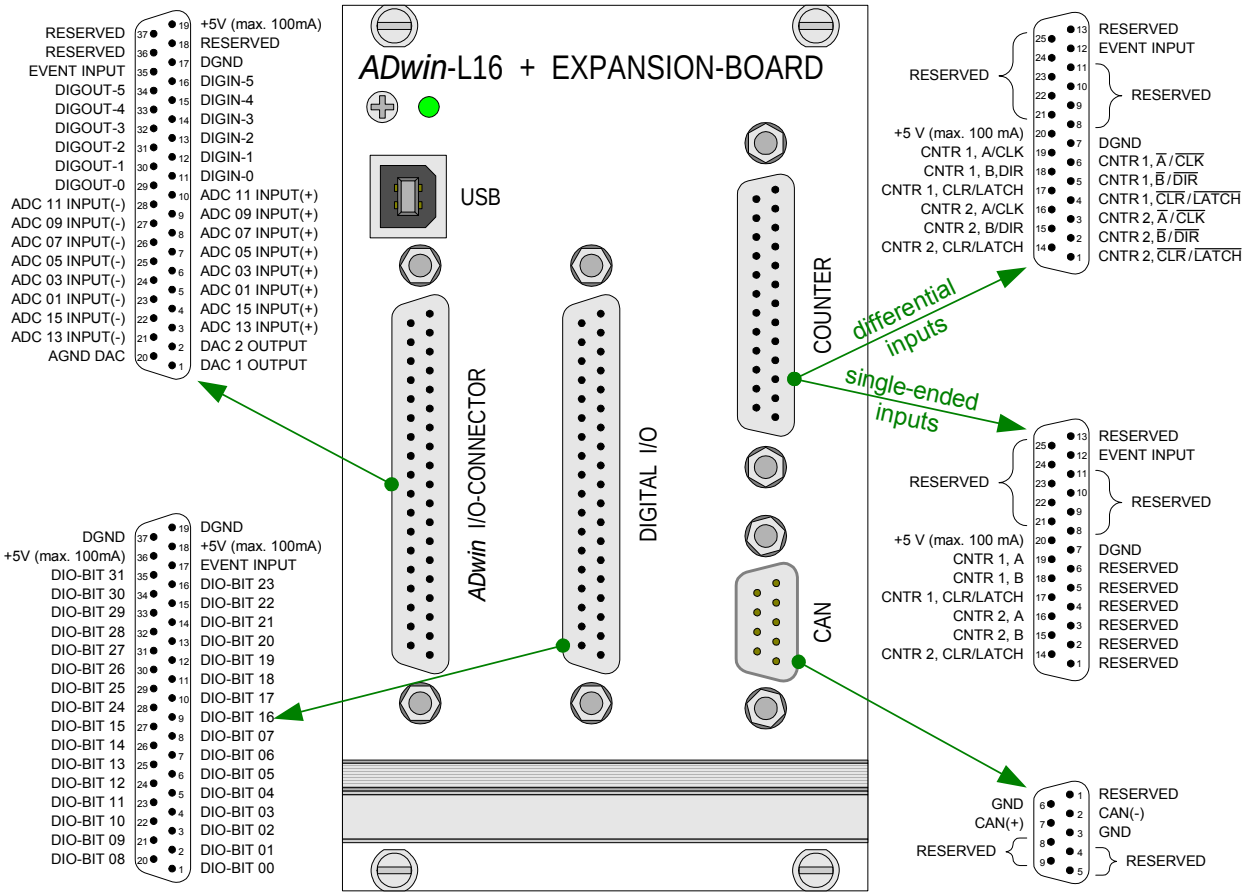


Fig. 22 – Overview of the *L16-EURO-DIO1* with pin assignments

Hardware configuration

There are several DIP switches on the DIO1 board, with which you can change the settings for the counters and the CAN interface.

The setting can only be made when the board is not installed or when the casing is opened. Please pay attention to the safety instructions at the beginning of this documentation.

Design	How to proceed
L16-EURO, L16-PCI, L16-cPCI	**Deinstalling the printed circuit board (PCB)** The deinstallation of the board is made in reverse order to the installation, which is described in the manual "***ADwin*** Driver Installation".
L16-EXT	**Opening the casing** You open the casing by unscrewing the upper Allen screws (2 mm) at both sides and loosening the lower screws. Slope the side plates and pull off the upper part of the casing. Note the orientation of the casing for the reinstallation. You are now looking directly on the DIO1 PCB.

Reassambling is made in reverse order.

Position of the DIP switches

The exact position of the DIP switches is shown in the following figure. The dotted frame shows the allocation of the DIP switches to the counters 1 and 2 (upper half, half at right) and to the CAN interface (lower left corner).

The setting of the DIP switches is described in chapter 8.2 "Counters" and chapter 8.3 "CAN-Bus".

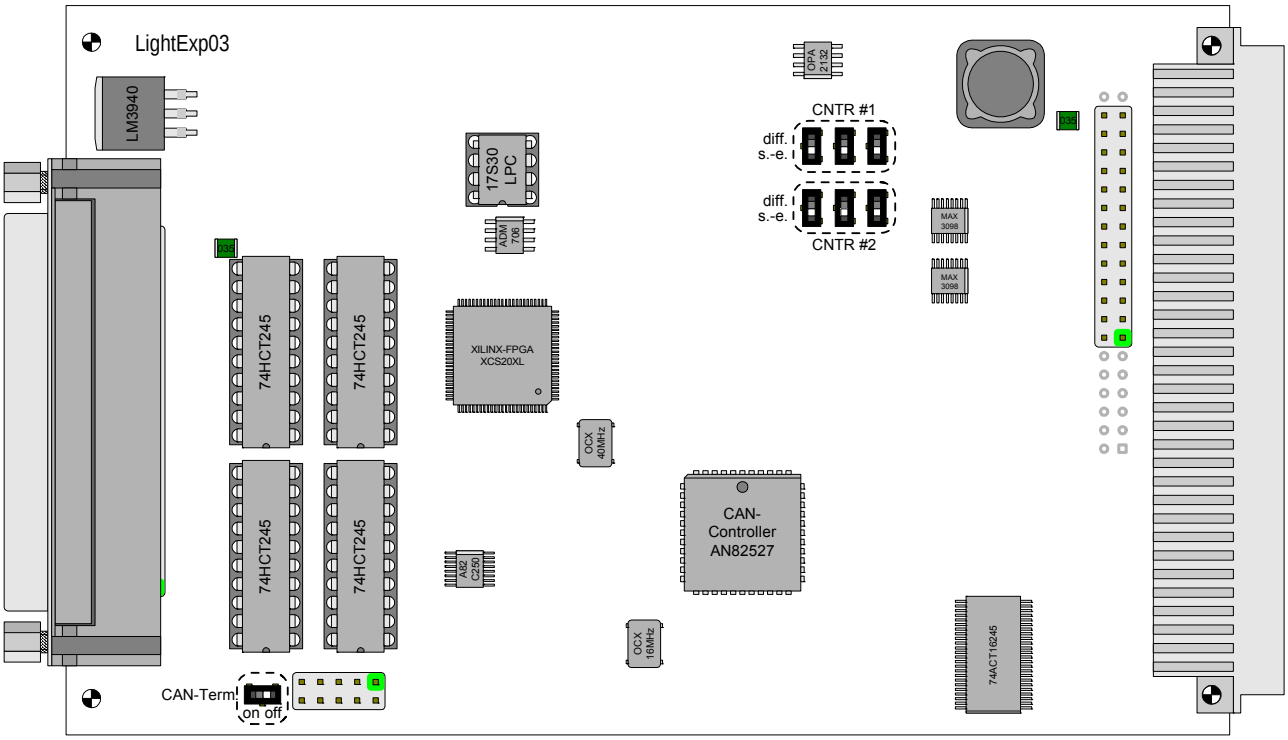


Fig. 23 – Position of the DIP switches on the DIO1 PCB

The technical data of the DIO1 add-on is shown in the Annex.

8.1 Digital Inputs and Outputs

In addition to the digital inputs/outputs of the basic version (DIGIN, DIGOUT, EVENT), you have 32 digital inputs or outputs (abbrev. DIO) on the 37-pin D-SUB socket "Digital I/O". They are programmable in groups of 8 each as inputs or outputs.

Trigger input

On each of the D-SUB sockets "Counter" and "Digital I/O" there is an external trigger input (EVENT). These inputs are galvanically connected and have the same pull-up resistor of 4.7 kΩ, so that you can select one of the inputs if necessary. Use only one of the three possible Event inputs.

With an external signal (trigger) at the event input a process can be triggered, and can be processed immediately and completely (see ***ADbasic*** manual, chapter: "Structure of an ***ADbasic*** program").

The digital inputs are TTL compatible and are not protected against overvoltage.

Power-up configuration

After power-up all connections are configured as inputs; this corresponds to the instruction `CONF_DIO_E(0)`. With the instruction

```
CONF_DIO_E(n)
```

You program the 32 DIO lines in 4 groups with 8 lines each as input or output (see online help). The following table shows the 16 possible configurations you will get with this instruction. In order to use this instruction you have to include the file `<ADWL16.INC>`.

CONF_DIO_E(n)	DIO 31 ... DIO 24	DIO 23 ... DIO 16	DIO 15 ... DIO 08	DIO 07... DIO 00
0	IN	IN	IN	IN
1	IN	IN	IN	OUT
2	IN	IN	OUT	IN
3	IN	IN	OUT	OUT
4	IN	OUT	IN	IN
5	IN	OUT	IN	OUT
6	IN	OUT	OUT	IN
7	IN	OUT	OUT	OUT
8	OUT	IN	IN	IN
9	OUT	IN	IN	OUT
10	OUT	IN	OUT	IN
11	OUT	IN	OUT	OUT
12	OUT	OUT	IN	IN
13	OUT	OUT	IN	OUT
14	OUT	OUT	OUT	IN
15	OUT	OUT	OUT	OUT
ADbasic instructions to use:	DIGIN_WORD2_E DIGOUT_WORD2_E DIGOUT_SET2_E DIGOUT_RESET2_E		DIGIN_WORD1_E DIGOUT_WORD1_E DIGOUT_SET1_E DIGOUT_RESET1_E	

Fig. 24 – Configurations with `CONF_DIO_E`

More information about programming of time-critical tasks can be found in chapter 5.4 on page 16.

8.2 Counters

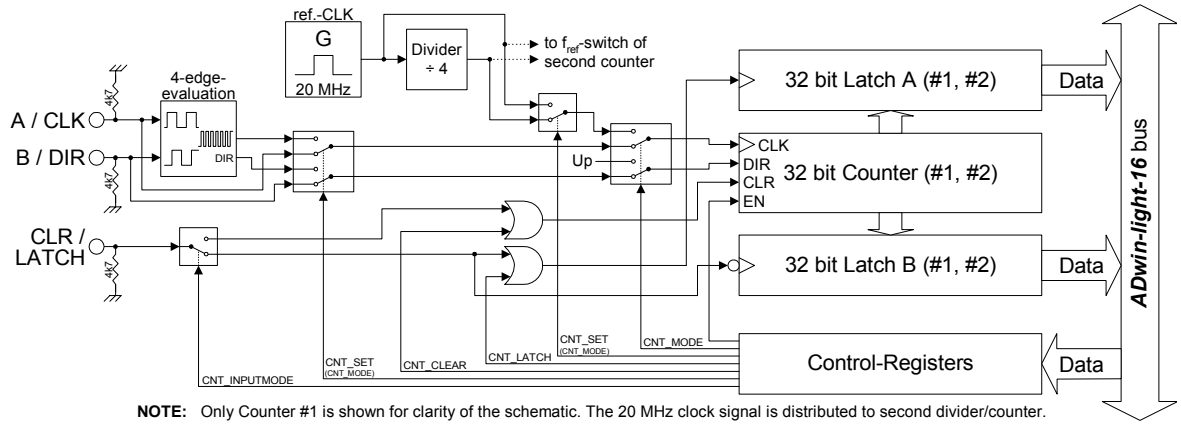
Counter

The add-on DIO1 provides **two 32-bit counters**, which you can configure and read out individually or all together. You can transfer single-ended or differential signals to the inputs.
The counters replace the incremental counters of the basic version.

Latch

The counters can be **internally or externally clocked** and are read out via accompanying latches. All counters have a latch A as well as a latch B (the figure shows the design of a single counter).

The counter values can be cleared or transferred into a latch by using programming instructions or (at special configurations) when there is an external signal at CLR/LATCH.



NOTE: Only Counter #1 is shown for clarity of the schematic. The 20 MHz clock signal is distributed to second divider/counter.

Fig. 25 – Block diagram of DIO1 counter

There are the following operating modes: event counting (external clock) and pulse width measurement (internal clock), see also chapter 8.2.2 / 8.2.3:

External clocking

4. **Event counting:** Incrementing/decrementing of the counter is caused by external square-wave signals at the inputs A/CLK and B/DIR. A signal at CLR/LATCH either sets the counter to zero (CLR) or has the counter value written into the latch (LATCH).

   There are the modes:
   - **Clock and direction**: Every positive edge at CLK increments or drecrements the counter value by one. The signal at DIR determines the counting direction (0 = down, 1 = up).
   - **Four edge evaluation**: Every edge of the signals (off-phase by 90 degrees) at A/CLK and B/DIR causes the counter to increment/decrement. The counting direction is determined by the sequence of the rising/falling edges of these signals. This mode is particularly used for incremental encoders.

Internal clock

5. **Pulse width measurement:** Incrementing/decrementing of the counter is caused by an internal reference clock with a signal frequency of 20 MHz (optionally 5 MHz after scaler). The square-wave signal at CLR/LATCH is evaluated: With every positive edge of the input signal the counter value is written to latch A, with a negative edge to latch B.

   You can calculate:
   - the period duration of the input signal at CLR/LATCH from the values in latch A and latch B.
   - the impulse width and pause time from the values in latch A and latch B.

Selecting the operating mode of the counter inputs

You can use the counter inputs in single-ended or differential mode. The setting is not fixed upon delivery. Therefore set the operating mode at the DIP switches of the DIO1 printed circuit board (position of DIP switches see fig. 23).

For differential mode set all 3 DIP switches of the corresponding counter into upward position (direction see also fig. 23); for single-ended operation set the switches into downward position.

8.2.1 Programming

The DIO1 counters are easily programmed by using ***ADbasic*** instructions. The instructions are part of an include file which must be included at the beginning of a program:

```
#INCLUDE ADWL16.INC
```

The instructions for the DIO1 counters are shortly illustrated in the following table and more detailed in the ***ADbasic*** manual or online help.

Counter nº	2	1	Comment
Bit	1	0	
CNT_CLEAR( )	0	0	no effect
	1	1	clear counter*
CNT_ENABLE( )	0	0	disable counter
	1	1	enable counter (pay attention to running counters)
CNT_INPUTMODE( )	0	0	set CLR/LATCH input to CLR mode
	1	1	set CLR/LATCH input to LATCH mode
CNT_LATCH( )	0	0	no effect
	1	1	copy counter value to latch register A*
CNT_MODE( )	0	0	external clock input
	1	1	internal reference clock (20MHz / 5MHz)
CNT_SET( )	0	0	CNT_MODE bit = 0 : 4 edge evaluation (A & B)
			CNT_MODE bit = 1 : internal reference clock of 20MHz
	1	1	CNT_MODE bit = 0 : clock and direction inputs (CLK & DIR)
			CNT_MODE bit = 1 : internal reference clock of 5MHz
CNT_CLEARENABLE( )	0	0	disable CLR input
	1	1	enable CLR input
CNT_GETSTATUS(#)			read status register (for the meaning of the bits see ADbasic manual or online help; # = counter nº 1 or 2)
CNT_READ(#)			copy counter value to latch A and read it (# = counter nº 1 or 2)
CNT_READLATCH(#)			read out latch A (triggered by pos. edge), (# = counter nº 1 or 2)
CNT_READFLATCH(#)			read out latch B (triggered by a neg. edge), (# = counter nº 1 or 2)
* The bits are reset after the function has been executed. All other functions are reset by the opposing function.			

Fig. 26 – DIO1 counter instructions - short reference

With these instructions of the table matrix you will be able to configure every counter individually or both counters together.

Please initialize the counters in the following order:

1. disable specified counter (CNT_ENABLE)

   The instruction CNT_ENABLE always accesses all counters. Even if you want to change the status (disabled/enabled) of only one counter, you

also have to configure the counters whose status shall remain unchanged.

2. set operating mode (CNT_MODE, CNT_SET, CNT_INPUTMODE)

Please take into account that the instruction CNT_SET is dependent on the instruction CNT_MODE.

3. clear counter (CNT_CLEAR)
4. enable counter (CNT_ENABLE)

Evaluate the counter value only with INTEGER or LONG variables. ***ADbasic*** then keeps internally the read bit patterns unmodified and considers automatically the transition from the positive to the negative numerical range (see also page 15). Then you get:

The count direction (up or down) can reliably be derived from the

sign of the difference: [new counter value] minus [old counter value]

and not from the *comparison* of the counter value.

Calculate the difference only with integer variables. (INTEGER, LONG).

A task can be very quickly processed if you access the control and data registers directly (see chapter 5.4 as well as ***ADbasic*** manual). The hardware addresses of the DIO1 add-on are illustrated in the following table (see also Fig. 26 – DIO1 counter instructions - short reference).

Address [hex]	Function	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Comment
20 40 02 04	contents of latch A, counter #1	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch
20 40 02 08	contents of latch B, counter #1	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch
20 40 02 14	contents of latch A, counter #2	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch
20 40 02 18	contents of latch B, counter #2	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch
20 40 03 00	enable/disable counter CNT_ENABLE( )	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : disable counter x = 1 : enable counter
20 40 03 10	clear counter CNT_CLEAR( )	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : no effect x = 1 : clear counter
20 40 03 20	latch counter CNT_LATCH( )	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : no effect x = 1 : latch counter
20 40 03 30	input:: CLR or LATCH	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : CLR input x = 1 : LATCH input
20 40 03 40	impulse/event counter or pulse width/period duration measurement	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : external clock input x =1: internal reference clock (20MHz/5MHz)
20 40 03 50	4 edge evaluation/CLK + DIR or 20MHz/5MHz refeference clock	-	-	-	-	-	-	-	-	x	x	x	x	CNT_MODE=0: x=0: 4-Fl.;x=1:CLK+DIR CNT_MODE=1: x=0: 20MHz; x=1: 5MHz
20 40 04 54	DIO1 add-on bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x: sets/clears output <bit n^{o}>
20 40 04 64	DIO1 add-on bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x: sets/clears output <bit n^{o}+16>
20 40 04 74	DIO1 add-on set_bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: no effect x = 1: set output <bit n^{o}> *
20 40 04 84	DIO1 add-on set_bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: no effect x = 1: set output <bit n^{o}+16>*
20 40 04 94	DIO1 add-on reset_bit 0...15	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: no effect x = 1: clear output <bit n^{o}> *
20 40 04 A4	DIO1 add-on reset_bit 16...31	-	x	x	x	x	x	x	x	x	x	x	x	x = 0: no effect x = 1: clear output <bit n^{o}+16> *
20 40 04 6C	configure inputs/outputs CONF_DIO() bit 0: DIO 00...07; bit 1: DIO 08...15 bit 2: DIO 16...23; bit 3: DIO 24...31	-	-	-	-	-	-	-	-	x	x	x	x	x = 0: configure channels as input x = 1: configure channels as output
* Function has no effect with inputs														

Fig. 27 – DIO1 hardware addresses of the control and data register

8.2.2 Operating mode impulse/event counting

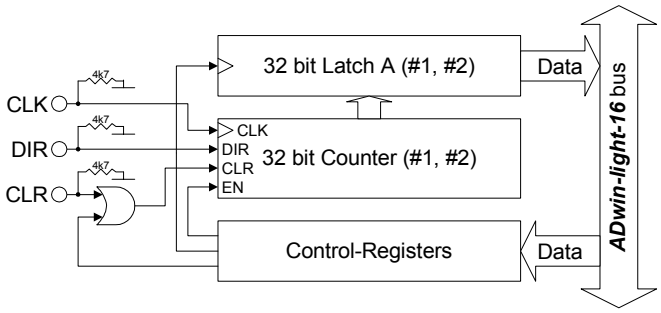
External square-wave signals at the inputs A/CLK and B/DIR clock the counters in this mode. With CNT_SET you either activate the mode for determining the clock frequency and direction or the four edge evaluation.

The input CLR/LATCH (at high-signal) can be used to

– clear the counter (CLR) **Clearing**

– latch the counter value into latch A (LATCH). **Latching**

Clock and direction



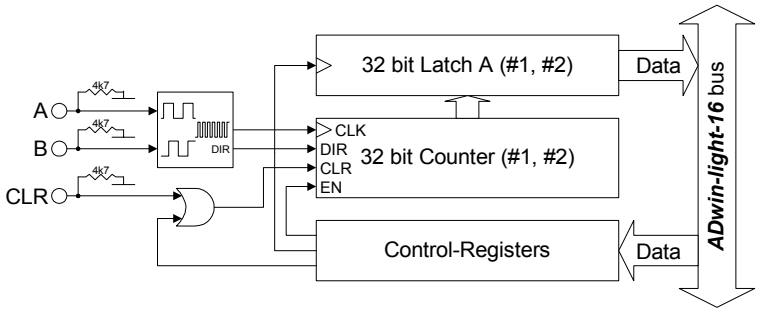
Every positive edge of a square-wave signal at the clock input (CLK) is counted (incremented or decremented) up to a maximum frequency of 20 MHz. The direction is derived from a high signal (increment) or low signal (decrement) at the direction input (DIR); this signal can be a fixed voltage or a dynamic signal (e.g. given by an external logic).

Programming example

```
INIT:                          initialize
  . . .
CNT_ENABLE(0)                  disable counter
CNT_CLEAR(1)                   clear counter
CNT_MODE(0)                    external clock input at the CLK input
CNT_SET(1)                     set mode clock and direction
CNT_INPUTMODE(0)               set input CLR/LATCH to CLR mode
CNT_ENABLE(1)                  enable counter
  . . .

EVENT:
CNT_LATCH(1)                   latch current counter value to latch A
CNT_READLATCH(1)               read latch A
  . . .                        evaluation in the program
```

Four edge evaluation

This mode determines clock and direction of two signals, which are input at A and B off-phase by 90 degrees (ideally). The count direction is determined by the temporal sequence of the rising and falling edges of the two input signals.



Please note:

– The counter counts 4 edges in each cycle.

– The maximum count frequency is 20 MHz. Together with the 4 edges per cycle it will result in a maximum input frequency of 5 MHz (at A or B).

– The time lap between an edge at A and an edge at B must not be shorter than 50 ns.
Impulse widths or pause durations shorter than 100 ns are not incremented.

– Changing the phase-shift (to ≠90 degrees) will have an effect on the maximum input frequency because of the minimum time lap of the edges. If it differs from 90 degrees, the maximum input freqeuency of 5 MHz decreases for instance to 45 degrees at 2.5 MHz.

Programming example

```
INIT:                          initialize
. . .
CNT_ENABLE(0)                  disable counter
CNT_CLEAR(1)                   clear counter
CNT_MODE(0)                    external clock input at CLK input
CNT_SET(0)                     enable mode four edge evaluation
CNT_INPUTMODE(0)               set input CLR/LATCH to CLR mode
CNT_ENABLE(1)                  enable counter
. . .

EVENT:
CNT_LATCH(1)                   latch current counter value to latch A
CNT_READLATCH(1)               read latch A
. . .                          evaluation in the program
```

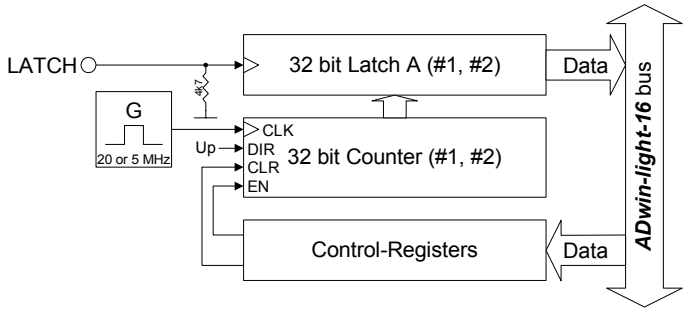
8.2.3 Operating mode pulse width and period duration measurement

In this operating mode an internal reference clock generator clocks the counter with a signal frequency of 20 MHz or (after a prescaler) 5 MHz. All counters have a switch in order to change the signal frequency. The period duration or pulse width of a square-wave signal at input CLR/LATCH can be measured.

in this mode you have to consider at high frequencies that your GLOBALDELAY remains smaller than a signal period, in order to acquire each cycle.

Period duration measurement

In this mode, a counter value is latched into latch A at every positive edge, and the previous data are overwritten. The pulse width will be derived from the counter value difference multiplied by the period duration of the reference clock.

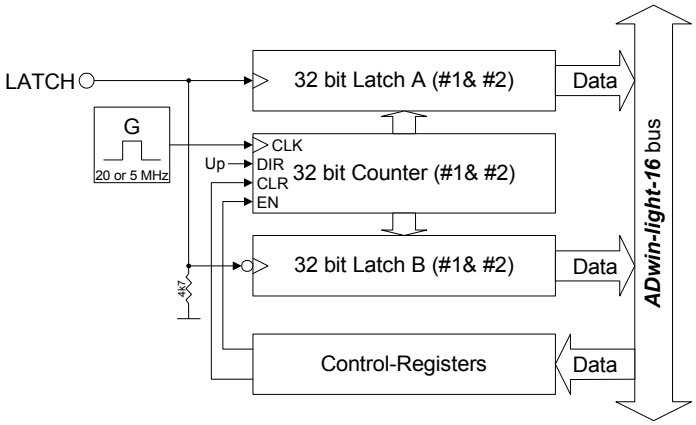


Programming example

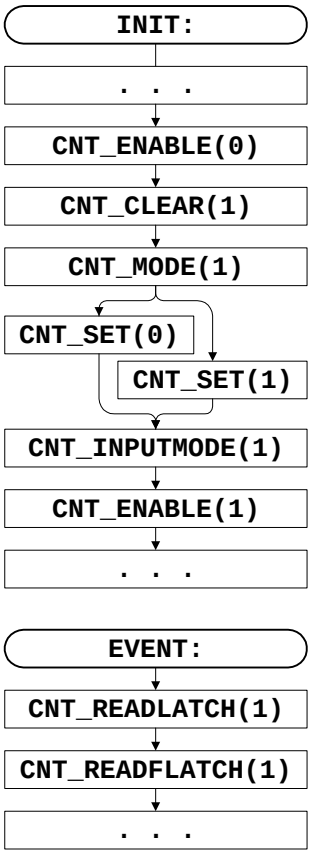
```
INIT:                          initialize
. . .
CNT_ENABLE(0)                  disable counter
CNT_CLEAR(1)                   clear counter
CNT_MODE(1)                    mode internal reference clock at internal CLK
                               input of the counter ...
CNT_SET(0)                     ... with 20 MHz  or
  CNT_SET(1)                   ... with 5 MHz
CNT_INPUTMODE(1)               set input CLR/LATCH to LATCH mode
CNT_ENABLE(1)                  enable counter
. . .

EVENT:
CNT_READLATCH(1)               read latch A
. . .                          evaluation in the program
```

Pulse width and pause duration measurement

The counters have each a latch A for positive and a latch B for negative edges. Thus, pulse and pause duration can be evaluated separately by calculating the differences of the latches.



Programming example



initialize

disable counter

clear counter

mode internal reference clock at internal CLK input of the counter ...

... with 20 MHz or

... with 5 MHz

set input CLR/LATCH to LATCH mode

enable counter

read latch A

read latch B

evaluation in the program

8.3 CAN-Bus

The add-on board DIO1 has a CAN bus interface for the high-speed CAN protocol. The connections are available on a 9-pin D-SUB connector (male); the pin assignment can be found on page 25.

Bus termination

The CAN bus has to be terminated at its physical end (and only there) by a resistor, that means only at the first and the last CAN node. This is made by setting a corresponding DIP switch on the DIO1 printed circuit board. (see Fig. 23 – Position of the DIP switches on the DIO1 PCB). If a termination is necessary, set the DIP switch toward the CAN connector.

8.3.1 Functions of the CAN controller

The CAN bus interface is equipped with the Intel® CAN controller AN82527 which works according to the specification CAN 2.0 parts A and B as well as to ISO 11898. You program the interface with ***ADbasic*** instructions, which are directly accessing the controller's registers.

Message

Messages sent via CAN bus are data telegrams with up to 8 bytes, which are characterized by so-called identifiers. The CAN controller of the DIO1 add-on supports identifiers with a length of 11 bit and 29 bit. The communication, that means the management of bus messages, is effected by 15 message objects.

The 255 registers are used for configuration and status display of the CAN controller. Here the bus speed and interrupt handling, etc. are set (see separate documentation "82527 - Serial Communications Controller, Architectural Overview" by Intel®)

The CAN bus can be set to frequencies of up to 1 MHz and is usually operated with 1 MHz; with low speed CAN the max. frequency is 125 kHz. The CAN bus is galvanically isolated by optocouplers from the ***ADwin*** system.

An arriving message can trigger an interrupt which instantaneously generates an event at the processor. Therefore an immediate processing of messages is guaranteed.

Message Management

Identifier

The CAN controller identifies messages by an identifier; these are parameters in a defined bit length. The parameters 0...2^{11}-1 or 0...2^{29}-1 result from the bit length.

Message objects

The controller stores each message (incoming or outgoing) in one out of 15 message objects. The message objects can either be configured to send or to receive messages. Message object 15 can only be used to receive messages. After initializing the CAN controller all message objects are not configured.

Each message object has an identifier, which enables the user to assign a message to a message object.

Transferring messages

In ***ADbasic*** a message is transferred to a message object using the array `can_msg[][]`, which can receive 8 data bytes plus the amount of data bytes (9 elements). When reading a message from the message object it can also be transferred to the array `can_msg[][]`.

Sending messages

Sending a message is made as follows:

– You configure a message object to send and define the identifier of the object (instruction **EN_TRANSMIT**).

– Save the message in `can_msg[]`.

– Send the message (instruction **TRANSMIT**). The message in the array `can_msg[]` is transferred to the message object. As soon as the bus is ready, the message is sent (with the identifier of the message object).

Receiving messages

Receiving a message is made as follows:

– You configure a message object to receive and define the identifier of the object (instruction **EN_RECEIVE**).

– The controller monitors the CAN bus if there are incoming messages and saves messages with the right identifier in the message object.

– Transfer the message from the message object into the array `can_msg[]` (instruction **READ_MSG**) and read out the corresponding identifier.

An arriving message overwrites the old data in the message object, which will be definitely lost. Therefore pay attention to reading out the data faster than you are receiving them. A data loss is indicated by a flag.

The message object 15 has an additional buffer, so that 2 messages can be stored there.

Assigning messages

The allocation of an arriving message to a message object is automatically controlled by comparing its identifiers. The global mask (CAN registers 6...7 or 6...9) controls this comparison as follows:

– The identifier of the message is bit by bit compared to the identifier of the message object. If the relevant bits are identical, the message is transferred to the message object. Not relevant bits are not compared to each other, that is, the message is transferred to the object (if it depends on this bit).

– Relevant bits are set in the global mask.

Global mask

With the global mask a message object is used for receiving messages with **different identifiers** (ID). The following example shows the assignment of the message IDs 1...4 to the message object IDs 1...4, when all bits of the global mask are set, except the two least-significant bits (if you have an 11-bit identifier it is 11111111100b).

Message ID	ID of the message object			
	1 …001b	2 …010b	3 …011b	4 …100b
1 (…001b)	x	x	x	0
2 (…010b)	x	x	x	0
3 (…011b)	x	x	x	0
4 (…100b)	0	0	0	x

x: Message is admitted
0: Message is not admitted

In this example the comparison of bit 2 is responsible for the assignment of the messages, because the bits 3...10 of the compared identifiers are identical (= 0) and the bits 0 and 1 are not compared, because they are set to zero in the global mask (= not relevant).

Setting the bus frequency

The **CAN bus frequency** depends on the configuration of the controller.

The initialization with INIT_CAN configures the controller automatically to a CAN bus frequency of 1 MHz. If the CAN bus is to operate with a different frequency, the values in the "Bit Timing Register 0" (BTR0, address 3Fh) and in the "Bit Timing Register 1" (BTR1, address 4Fh) have to be changed. Just use the instruction **SET_CAN_BAUDRATE** for setting a large quantity of bus frequencies.

With low speed CAN the maximum bus frequency is 125kBit/s.

Bus frequency for special cases

In some special cases it may be better to select configurations other than those set with the instruction mentioned above. For this purpose specified registers have to be set with the instruction POKE. The structure of the register is described below.

	Bit Timing Register 0 (BTR0)		Bit Timing Register 1 (BTR1)		
Bits	7…6	5…0	7	6…4	3…0
Sub-Reg.	SJW	BRP	SPL	TSEG2	TSEG1

The following table shows the admitted values and the meaning of the individual ranges:

Range	Admitted values	Meaning
SJW	0 … 3	Max. pulse elongation during bus synchronization
BRP	0 … 63	Pre-scaler
SPL	0 … 1	Sampling mode
TSEG1	2 … 15	Time segments before sampling
TSEG2	1 … 7	Time segments after sampling

The default setting of the ranges SJW and SPL is 0 and should only be changed if necessary. Select the sample point (specified by TSEG1 and TSEG2) in such a way that it is between 50% and 80% of the total bit length.

The CAN bus frequency is calculated as follows:

$$f_{CAN} = \frac{8MHz}{(BRP + 1)(TSEG1 + TSEG2 + 3)}$$

The following table illustrates all common settings for the Baud rates.

Baud rate [kBit/s]	125	250	500	1.000
BRP	3	1	0	0
TSEG1	6	6	6	2
TSEG2	7	7	7	3
BTR0	03h	01h	00h	00h
BTR1	76h	76h	76h	32h
Sample point[%]	54	54	54	60

Fig. 28 – CAN: Setting the Baud rates

Access to the two timing registers is only possible, when the access has been enabled before. This is done by the CCE-bit in the control register. The bit has to be reset afterwards.

Interrupt / Event

A message object can be enabled to trigger an interrupt when a message arrives. The interrupt output of the CAN controller is connected to the event input of the processor. The processor reacts immediately to incoming messages without having to control the message input (polling).

You can enable the interrupts of several message objects. Which object has caused the interrupt can be seen in the interrupt register (5Fh): It contains the number of the message object that caused the interrupt. If the interrupt flag (new message flag) is reset in the message object, the interrupt register will be updated. If there is no interrupt the register is set to 0. If another interrupt occurs during working with the first interrupt its source will be shown in the interrupt register. An additional interrupt does not occur in this case.

Programming

The DIO1 CAN interface is easily programmed by using ***ADbasic*** instructions. The instructions are part of an include file which must be included at the beginning of a program:

```
#INCLUDE ADWL16.INC
```

The instructions for the DIO1 CAN interface are shortly illustrated in the following table and more detailed in the ***ADbasic*** manual or online help.

Function	Instruction
Initializing the CAN controller	INIT_CAN
Setting and reading of registers	SET_REG, GET_REG
Initializing of message objects	EN_RECEIVE, EN_TRANSMIT
Transmitting and receiving data sets	TRANSMIT, READ_MSG
Enabling of interrupts	EN_INTERRUPT
Setting the Baud rate	SET_CAN_BAUDRATE

Fig. 29 – DIO1 Overview of CAN instructions

9 *ADwin-light-16*-Boot

This option is only available in conjunction with the Ethernet interface, that means with the *L16-EXT-ENET* or the *L16-EURO-ENET*.

After power-up ADwin-light-16-Boot automatically starts an application which has been programmed before. Thus after initializing the application an operation without PC is possible.

ADwin-light-16-Boot executes the following steps after power-up:

– Loading the operating system

– Loading the processes generated with the ***ADbasic*** compiler (max. 10)

– Automatic starting of the process nº 10. Here you have also to program the start of all other processes

If you do not wish to work with the bootloader option:

– Boot the system after power-up and the programmed processes are disabled

– After switching off and powering up anew, the bootloader option is enabled again

By programming the Flash-EEPROMs without processes the system will only be booted after restart with the file <ADwin9.btl>. A process will not be executed.

With the installation of the ***ADwin***-Developer software from the ***ADwin-***CD-ROM, the program for the bootloader option (ADethflash) is automatically copied. The version of the CD-ROM should be 3.00.2735 or higher.

Use the program ADethflash for an ***ADwin*** system with an Ethernet interface.

At standard installation you will find the program in the directory

<C:\ADwin\Tools\Ethernet Interface\...>.

Notes for the bootloader with Ethernet interface can be found in the documentation "***ADwin*** Driver Installation".

Together with with an Ethernet-Interface and bootloader you can save and read up to 2,000 Long or Float values with 32 bit via ***ADbasic*** process in the built-in Flash-EEPROM memory. You can find a more detailed description in the program <ADethflash.exe>, if you click on the button "Info about eeprom support".

10 Accessories

The following accessories are available for the ***ADwin-light-16*** system:

– ***ADwin-light-16***-pow: external 12V power supply
– various lengths of power supply and USB or Ethernet cables
– cable connector for an external power supply
– installation kit for enclosures

ADwin-light-16-pow

On the secondary side ADwin-light-16-pow provides 12 Volt at a maximum continuous load of 2 Ampere. The power supply is rated for the highest load and maximum expansions.

Cable connector

In case you want to use an external power supply, you need the cable connector for the correct connection to the ***ADwin-light-16*** system.

Please pay attention to a sufficient shielding of the USB and Ethernet cable, in order to avoid interferences in the data lines. Interferences have to be conducted before the chassis via ground (see also chapter 3).

11 Software

You are programming the ***ADwin-light-16*** - all expansions included - with simple ***ADbasic*** instructions.

All instructions are described in the ***ADbasic*** manual (which contains a complete instruction list in the appendix).

11.1 Example Program

CAN: Cyclic Read and Send of Messages

This programs describes the initialization of the CAN controller in the section `INIT:` and the cyclic read and send of messages in the section `EVENT:`

```
REM The program initializes the CAN controller,
REM configures one message object as sender
REM and one as receiver. The program exchanges all 10 ms data
REM between CAN controller and transputer.

#INCLUDE ADWL16.INC

DIM result AS INTEGER

INIT:
 INIT_CAN()                  'Initialize the CAN controller
                             'Set Baud rate to 125 kBit/s
 SET_CAN_BAUDRATE(125000)

 EN_RECEIVE(2,385,0) 'configure message object 2 for reading
                             'with 11 bit identifier 385
 EN_TRANSMIT(3,1,0) 'configure message object 3 for writing
                             'with 11 bit ident. 1

EVENT:
 REM read 1 data set and write 1 data set

 result = READ_MSG(2)'read data
 'If there are new data, they are written into the field
 'CAN_MSG.


 can_msg[1]=1                'data, which are to be sent,
 can_msg[2]=2                'are written into the field
 can_msg[3]=3                'CAN_MSG. You are getting the
 can_msg[4]=4                'data from this field to be
 can_msg[5]=5                'sent later
 can_msg[6]=6
 can_msg[7]=7
 can_msg[8]=8
 can_msg[9]=8                '8 data bytes

  TRANSMIT(3)                'send message in message object 3
```

CAN: Interrupt-Controlled Reading

The following program shows the initialization of the CAN controller and the interrupt-controlled reading of new messages:

```
REM The program initializes the CAN controller and
REM configures a message object as receiver.
REM The program reads interrupt-controlled messages
REM as soon as a new message arrives.

#INCLUDE adwl16.inc

DIM result,status,object AS INTEGER

INIT:
 INIT_CAN()                 'Initialize the CAN controller
                            'set Baud rate to 125 kBit/s
 SET_CAN_BAUDRATE(125000)

 EN_RECEIVE(1,385,0)        'message object 1 is configured
                            'for reading. Only messages with
                            '11 bit identifier 385 are saved.

 status = GET_CAN_REG(1)    'read status
 EN_INTERRUPT(1)            'When a message arrives in
                            'message object 1 an interrupt
                            'is triggered.

EVENT:
 object = GET_CAN_REG(5Fh)  'read interrupt register

 IF (objekt = 2) THEN       'Get the number of the message
  objekt = 15               'object, where the new message
 ELSE                       'can be found
  object = object - 2
 ENDIF

 result = READ_MSG(object)  'read out new data

 REM The data are available in the field CAN_MSG.
```

Annex

A.1 Technical Data

General Data / Limit Values						
	Symbol	Conditions	min.	typ.	max.	Unit
Supply Voltage						
voltage	U_b	*L16-PCI, -EURO, -cPCI*	4.75	5	5.25	V
		L16-EXT	10	12	18	
Supply Voltage with USB						
operating current						
L16, *L16-CO1*	I_{idle} at $U_{b, typ.}$	*L16-PCI, -EURO, -cPCI*	0.75	0.9	1.5	A
		L16-EXT	0.35	0.5	0.7	
L16-DIO1		*L16-PCI, -EURO, -cPCI*	0.85	1.0	1.6	
		L16-EXT	0.55	0.7	1.0	
inrush current						
L16, *L16-CO1*	$I_{power-on}$ bei $U_{b, typ.}$	*L16-PCI, -EURO, -cPCI*		6		A
		L16-EXT		3		
L16-DIO1		*L16-PCI, -EURO, -cPCI*		7		
		L16-EXT		3		
Supply Voltage with Ethernet						
operating current						
L16, *L16-CO1*	I_{idle} bei $U_{b, typ.}$	*L16-PCI, -EURO, -cPCI*	1.0	1.2	1.8	A
		L16-EXT	0.55	0.7	0.9	
L16-DIO1		*L16-PCI, -EURO, -cPCI*	1.15	1.3	1.9	
		L16-EXT	0.55	0.7	1.0	
inrush current						
L16, *L16-CO1*	$I_{power-on}$ bei $U_{b, typ.}$	*L16-PCI, -EURO, -cPCI*		6.4		A
		L16-EXT		3.3		
L16-DIO1		*L16-PCI, -EURO, -cPCI*		7.5		
		L16-EXT		3.3		
Operation						
temperature	$T_{environment}$	*L16-PCI, -EURO, -cPCI*	+5		+50	°C
	$T_{chassis}$	*L16-EXT*	+5		+55	
relative humidity	H_{rel}	no condensation	0		80	%
Storage						
temperature	T		-20		+70	°C
Dimensions						
width x height x depth	w x h x d	*L16-PCI-USB*	21.5 x 121.0 x 172.5			mm
		L16-EURO-USB	5 TE (1") x 3 HE (5") x 187			
		L16-EURO-ENET	10 TE (2") x 3 HE (5") x 187			
		L16-EXT-USB	226 x 109 x 44			
		L16-EXT-ENET	226 x 109 x 74			
		+ *CO1* add-on	see basic version			
		+ *DIO1* add-on	*L16-EURO*: width +10 TE *L16-EXT*: height +30			

General Data / Limit Values						
	Symbol	Conditions	min.	typ.	max.	Unit
Net Weight						
weight	m_{Netto}	*L16-PCI*-USB		135		g
		L16-EURO-USB		165		
		L16-EURO-ENET		275		
		L16-EXT-USB		1.100		
		L16-EXT-ENET		1.400		
		+ CO1 add-on		see basic version		
		+ DIO1 add-on		+140		
Mounting						
standard	*L16-PCI*: installation in the PC *L16-EURO*, *L16-cPCI*: installation in the 19"-enclosure *L16-EXT*: desktop unit					
optional	DNI rail mounting and wall monting for *L16-EXT*					

Digital Inputs/Outputs						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
I/O-lines						
line	DIGIN05:00 DIGOUT05:00	6 inputs and 6 outputs (TTL- / 5V-CMOS-level)				
	EVENT	1 ext. trigger input (positive TTL-logic)				
as inputs						
max. input voltage		TTL-level	-0.5		+5.5	V
logic-input voltage	V_{IH} (High)	V_{CC}= 5V	2			
	V_{IL} (Low)	V_{CC}= 5V			0.8	
logic-input current	I_I	V_{CC}= 5V		±0.1	±1,000	nA
as outputs						
logic-output voltage	V_{OH} (High)	I_{OH}= -6mA	3.84	4.3		V
	V_{OL} (Low)	I_{OL}= +6mA		0.17	0.33	
logic-output current	I_O	per DIO line			±35	mA
	I_{TOTAL}	all DIGIN or all DIGOUT via V_{CC} / GND			±70	
Counters						
Number and function	2 incremental counters.					
Counter and latch resolution				32		Bit
EVENT input						
Edge recognition, pos.	V_{T+} (Low)	V_{CC} = 5V	1.65	1.9	2.15	V
Edge recognition, neg.	V_{T-} (High)	V_{CC} = 5V	0.75	1.0	1.25	
Switching hysteresis	V_{T+} - V_{T-}		0.4	0.9		
input current	I_{IH}	V_I = 2.7V			20	µA
	I_{IL}	V_I = 0.4V			-50	

Analog Inputs/Outputs						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
Inputs						
number	8, differential via multiplexer					
input resistance	R_i		323.4	330	336.6	kΩ
overvoltage	$U_{in\ max.}$	ON & OFF			±35	V
MUX settling time	t_{MUX}	2 LSB 16 Bit		6.5 *		µs
*at an internal resistance of thevoltagesource of less than 10 Ω						
ADC 16-Bit						
conversion time	t_{conv}				10	µs
measurement range	U_{in}		-10		+9.999695	V
diff. common mode voltage					±2.0	
integral non-linearity	INL			±1	±3	LSB
differential non-linearity	DNL			±0.25	±0.5	
offset	drift			±2		ppm/°C
	error	adjustable				
gain	drift			±20		ppm/°C
	error	adjustable				
DAC 16-Bit						
number	2					
output voltage	U_{out}		-10		+9.999695	V
settling time	t_{settle}	2V step			3	µs
		FSR* (20V)			10	
maximum current					±5	mA
integral non-linearity	INL				±2	LSB
differential non-linearity	DNL				±1	
offset	error	adjustable				
gain	error	adjustable				
*FSR = Full Scale Range						

Processor						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
type	ADSP21062 (SHARC™)					
manufacturer	Analog Devices					
clock frequency	f_{CLK}			40		MHz
register width				32		Bit
internal memory(SRAM)		for the program		128		kByte
		for data		128		
external memory (SDRAM)				8		MByte

CO1 Add-on						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
Counters						
Number and function	1 up/down counter with four edge evaluation for connection of incremental encoders. The incremental counters of the basic version are replaced.					
Counter inputs	2 inputs (A, B), alternatively available as digital inputs					
Counter and latch resolution				32		Bit

DIO1 Add-On						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
Reference crystal oscillator						
reference frequency	f_{ref}			20		MHz
prescaler by 4	$f_{ref/4}$			5		
accuracy and drift					100	ppm
Counters						
Number and function	2 up/down counters for impulse, period duration and duty cycle measurements as well as a four edge evaluation for connection of incremental encoders. The incremental counters of the basic version are replaced.					
Counter inputs	3 differental inputs (A/CLK, B/DIR, CLR/LATCH) for each counter; counter programmable with differential or single ended inputs.					
Counter and latch resolution				32		Bit
count frequency	f_{CLK}	input CLK		20		MHz
		input A/B		5		
Digital Inputs/Outputs						
number	DIO31:00	32 (programmable in groups of 8 as inputs or outputs)				
	EVENT	ext. trigger input (pos. TTL-logic)				
as inputs*						
max. input voltage		TTL-level	-0.5		+5.5	V
logic-input voltage	V_{IH} (High)	V_{CC} = 5V	2			
	V_{IL} (Low)	V_{CC} = 5V			0.8	
logic-input current	I_I	V_{CC} = 5V		±0.1	±1,000	nA
as outputs*						
logic-output voltage	V_{OH} (High)	I_{OH}= -6mA	3.84	4.3		V
	V_{OL} (Low)	I_{OL}= +6mA		0.17	0.33	
logic-output current	I_O	each DIO line			±35	mA
	I_{TOTAL}	each DIO group (8) via V_{CC} / GND			±70	
* see also data sheet SN74 HCT 245 from Texas Instruments						

A.2 Hardware Addresses - General Overview

Address [HEX]	Function	Bit No.								Comments	Register available in the module		
		31...16	15...6	5	4	3	2	1	0		L16	L16+ CO1	L16+ DIO1
20 40 00 00	set multiplexer to input channel (ADC 01... ADC 15)	-	-	-	-	-	n	n	n	"nnn" binary = 0...7 decimal, selected channel = nnn*2 + 1	x	x	x
20 40 00 10	start conversion: ADC #1	-	-	-	-	-	-	-	s	s = 0 : start conversion s = 1 : no effect	x	x	x
	start conversion: all DACs synchronously	-	-	-	-	-	s	-	-				
20 40 00 20	conversion status (EOC) ADC #1	-	-	-	-	-	-	-	e	e = 0 : end of conversion e = 1 : conversion is running	x	x	x
20 40 00 30	read out register: ADC #1	-	x	x	x	x	x	x	x	x : result of conversion	x	x	x
20 40 00 50	only write into register: DAC #1	-	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20 40 00 60	only write into register: DAC #2	-	x	x	x	x	x	x	x				
20 40 00 B0	input register DIGIN-05:00	-	-	x	x	x	x	x	x	x : read digital value	x	x	x
20 40 00 C0	output register DIGOUT-05:00	-	-	x	x	x	x	x	x	x : digital value to be output	x	x	x
20 40 00 C4	set DIGOUT bits	-	-	x	x	x	x	x	x	x = 0 : no effect x = 1 : set bit	x	x	x
20 40 00 C8	clear DIGOUT bits	-	-	x	x	x	x	x	x	x = 0 : no effect x = 1 : clear bit	x	x	x
20 40 01 00	read out register and start conversion: ADC #1	-	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20 40 02 00	write into register and start conversion immediately: DAC #1	-	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20 40 02 04	contents of Latch A, counter #1	x	x	x	x	x	x	x	x	x : contents of latch register	x	x	x
20 40 02 08	contents of Latch B, counter #1 (DIO1 only)	x	x	x	x	x	x	x	x	x :contents of latch register	-	-	x
20 40 02 10	write into register and start conversion immediately: DAC #2	-	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20 40 02 14	contents of Latch A, counter #2	x	x	x	x	x	x	x	x	x : contents of latch register	x	-	x
20 40 02 18	contents of Latch B, counter #2	x	x	x	x	x	x	x	x	x : contents of latch register	-	-	x
20 40 03 00	enable/disable counter: `CNT_ENABLE()`	-	-	-	-	-	-	x	x	x = 0 : disable counter x = 1 : enable counter	x	Bit 0 only	x
20 40 03 10	clear counter: `CNT_CLEAR()` *	-	-	-	-	-	-	x	x	x = 0 : no effect x = 1 : clear counter	x	Bit 0 only	x
20 40 03 20	latch counter: `CNT_LATCH()` *	-	-	-	-	-	-	x	x	x = 0 : no effect x = 1 : latch counter	x	Bit 0 only	x
20 40 03 30	counter inputs CLR or LATCH	-	-	-	-	-	-	x	x	x = 0 : CLR input x = 1 : LATCH input	-	-	x
20 40 03 40	impulse/event counter or pulse width/period duration measurement	-	-	-	-	-	-	x	x	x = 0 :external clock input x = 1 : internal reference clock (20MHz / 5MHz)	-	-	x
20 40 03 50	4 edge evaluation/ CLK+DIR or 20MHz / 5MHz reference clock	-	-	-	-	-	-	x	x	`CNT_MODE=0`: x=0: 4 edge evaluation; x=1: CLK+DIR `CNT_MODE=1`: x=0: 20MHz; x=1: 5MHz	-	-	x
20 40 04 54	bits DIO-15:00	-	x	x	x	x	x	x	x	x = 0: clear output x = 1: set output	-	-	x
20 40 04 64	bits DIO-31:16	-	x	x	x	x	x	x	x	x = 0: clear output x = 1: set output	-	-	x
20 40 04 74	set bits DIO-15:00 **	-	x	x	x	x	x	x	x	x = 0: no effect x = 1: set output	-	-	x
20 40 04 84	set bits DIO-31:16 **	-	x	x	x	x	x	x	x	x = 0: no effect x = 1: set output	-	-	x
20 40 04 94	clear bits DIO-15:00 **	-	x	x	x	x	x	x	x	x = 0: no effect x = 1: clear output	-	-	x
20 40 04 A4	clear bits DIO-31:16 **	-	x	x	x	x	x	x	x	x = 0: no effect x = 1: clear output	-	-	x
20 40 04 6C	configure inputs/outputs CONF_DIO() Bit 0: DIO 07:00; Bit 1: DIO 15:08 Bit 2: DIO 23:16; Bit 3: DIO 31:24	-	-	-	-	x	x	x	x	x = 0: group as input x = 1: group as output	-	-	x

* after execution the register is automatically reset
** function without any effect at inputs

A.3 Table of figures

A.4 List of Abbreviations

A/D	Analog to Digital
ADC	Analog to Digital Converter
ADSP	Analog Devices Signal Processor
b	Binary number
CLK	CLocK
CLR	CLeaR
CMOS	Complementary Metal Oxide Semiconductor
CMRR	Common Mode Rejection Ratio
D/A	Digital to Analog
DAC	Digital to Analog Converter
DIL	Dual InLine
DIO	Digital Input / Output
DIR	DIRection
DMA	Direct Memory Access
DMM	Digital Multi-Meter
DNL	Differential Non-Linearity
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processor
EOC	End Of Conversion
EMC	Electro-Magnectic Compatibility
ESD	Electro-Static Discharge
FPGA	Field Programmable Gate Array
FSR	Full Scale Range
GND	GrouND

Manufacturer

AD	Analog Devices
BB	Burr-Brown
LT	Linear Technology
TI	Texas Instruments

h / Hex	Hexadecimal number
I/O	Input / Output
IC	Integrated Circuit
InAmp	Instrumentation Amplifier
INL	Integral Non-Linearity
IRQ	Interrupt ReQuest
kB	kilo Byte (= 1024 Byte)
kByte	see kB
LED	Light Emitting Diode
LSB	Least Significant bit
MB	Mega-Byte (= 1024 kB)
MByte	see MB
MSB	Most Significant bit
MUX	Multiplexer
OpAmp	Operational Amplifier
PC	Personal Computer
PGA	Programmable Gain Amplifier
S&H	Sample & Hold
SRAM	Static Random Access Memory
TTL	Transistor-Transistor Logic
V_{cc}	Voltage collector-collector
V_{ee}	Voltage emitter-emitter