ADwin-Gold-
USB / -ENET

Manual

ADwin-Gold USB / ENET, manual version 3.8, 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 complex information about the operation of the ***ADwin-Gold*** system. Additional information are available in

- the manual "ADwin Driver Installation", which describes all interface installations for the ***ADwin*** systems.
  With this manual you begin your installation!
- the description of the configuration program ADconfig, with which you initialize the communication from the corresponding interface to your ***ADwin-Gold*** system.
- the manual ***ADbasic***, which contains all instructions for the compiler ***ADbasic*** and which explains the functional layout of the ***ADwin*** system.
- the manual for the description of the installation and the instructions of all current 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.

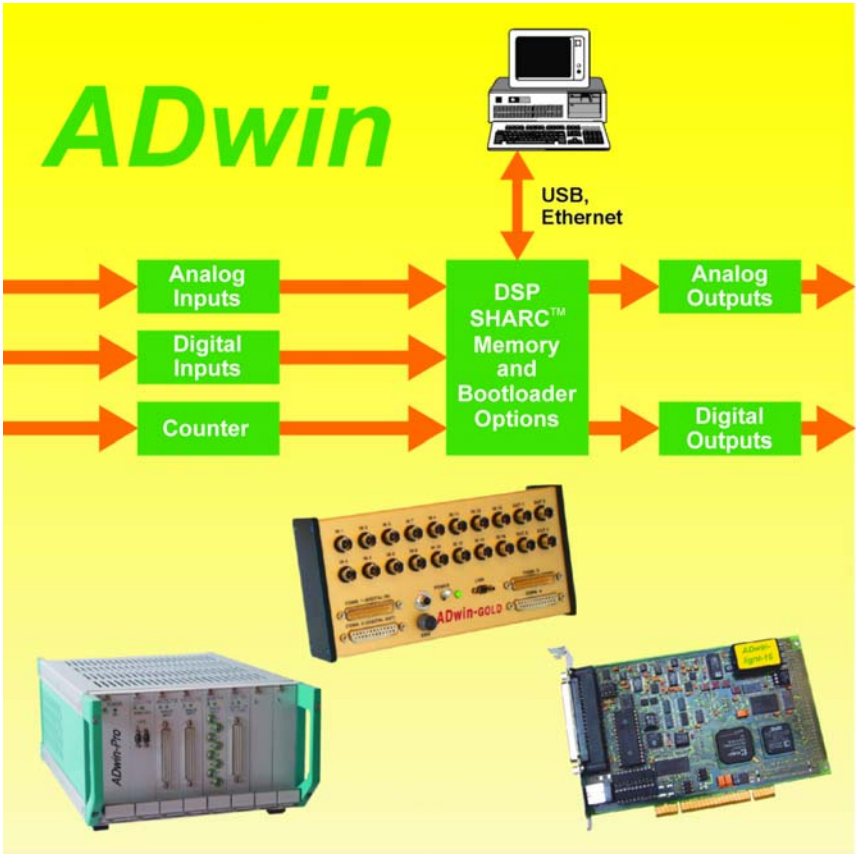


Fig. 1 – Concept of the ***ADwin*** systems

2.2 The *ADwin-Gold* System

Processor and memory

The ***ADwin-Gold*** system is equipped with the digital **32 bit signal processor T9** (SHARC ADSP 21062) from Analog Devices with floating point and integer processing. It is responsible for the complete measurement data acquisition, online processing, and signal output, and makes it possible to process instantaneously sample rates of up to several 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*** processes and all variables.

In order to get maximum access times, 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 of 16 MB (DRAM; optional 64 MB).

Analog inputs

The system has **16 analog inputs** with BNC plugs (alternatively: DSub connectors), which are divided into two groups each being connected to one multiplexer. These two outputs are optionally converted by a 14-bit or 16-bit analog-to-digital converter (ADC), (see Fig. 2 "Block diagram of the ADwin-Gold). With the 14-bit ADCs it is possible to sample **very fast,** with the 16-bit ADCs **highly accurately**.

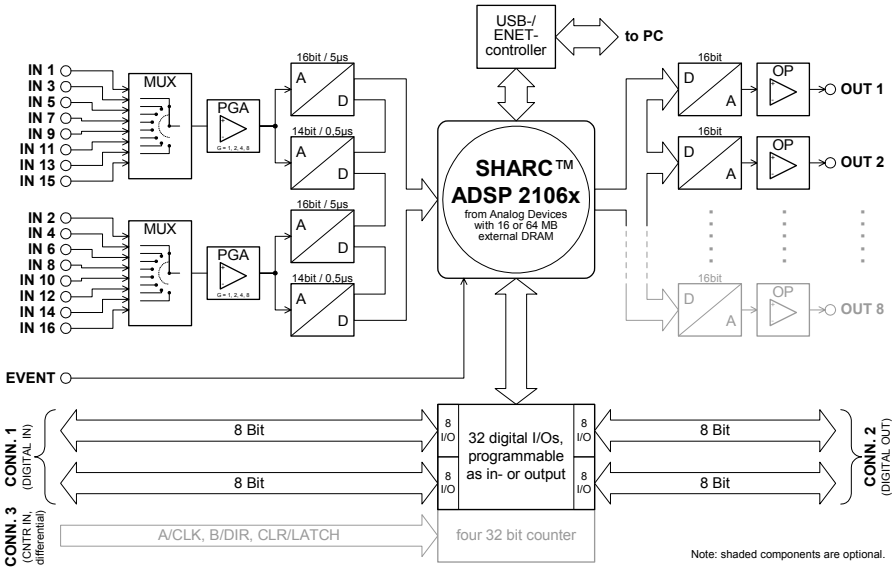


Fig. 2 – Block diagram of the ***ADwin-Gold***

Analog outputs

The standard version of the ***ADwin-Gold*** system is equipped with **2 analog outputs** (optional 8) with an output voltage range of -10V … +10V and a 16-bit resolution. You can synchronize the output of the voltage of all DACs per software.

Digital inputs and outputs

32 digital inputs or outputs are available on two 25-pin D-Sub connectors. They can be programmed in groups of 8 as inputs or outputs. The inputs or outputs are TTL-compatible.

Trigger input (EVENT)

The ***ADwin-Gold*** has a trigger input (EVENT, see also chapter 5.2 "Digital Inputs and Outputs"). Processes can be triggered by a signal and are completely processed afterwards. (see ***ADbasic*** manual, chapter "Structure of the ***ADbasic*** Program").

All analog data inputs and outputs of the system are differential.

The connection between ***ADwin-Gold*** system and computer is made via the USB or Ethernet interface (depending on the version you have purchased).

Standard delivery

The standard delivery items for the ***ADwin-Gold*** system:

– the ***ADwin-Gold*** system with USB or Ethernet interface,

– a USB cable or a cross-over Ethernet cable from the PC to the Gold device (length about 1.8 m).

– the power adapter: a three-pin power supply cable, which prevents the possibility of mismatch, at a slot metal sheet with socket connector,

– the power supply cable from the power adapter to the system,

– the ***ADwin*** CDROM,

– the manual "Driver Installation",

– this hardware manual.

2.2.1 Options (no upgrades possible)

Options

The following options are available:

– ***Gold-D***: All inputs and outputs have DSub-connectors, including the analoge inputs (instead of BNC plugs).

– ***Gold-DA***: 6 additional analog outputs (differential) with a 16-bit DAC each.

– ***Gold-CO1***: counter option with four 32 bit counters, which can optionally be used for period width measurement, as impulse counters or as up/down counters with clock/direction or four edge evaluation for quadrature encoders.

– ***Gold-CAN***: 4 decoders for use with incremental encoders with SSI interface, 2 CAN interfaces (both either high speed *or* low speed) and 2 RSxxx interfaces (RS232, RS485). This option is available in combination with the option Gold-D only.

– ***GOLD-MEM-64***: external memory with 64 MB instead of 16 MB and 512 kB internal CPU memory instead of 256 kB.

– ***Gold-Boot***: Flash-EPROM boot loader for stand-alone operation without PC (**only** in combination with the ***Gold-ENET***).

If not excluded above, all additional options can be combined with each other.

2.2.2 Accessories

Accessories

– ***ADbasic***, real-time development tool for all ***ADwin*** systems

– ***ADwin-Gold-pow***: external power supply (necessary for notebook operation)

– ***Gold-Mount***: kit for installation of the ***ADwin-Gold*** system on a DIN rail.

– Single cable-connector for a self-made external power supply cable.

3 Operating Environment

The ***ADwin-Gold*** electronic is installed in a closed aluminum enclosure and it is only allowed to operate it in this enclosure. With the necessary accessories the system can be operated in 19-inch-enclosures or as a mobile system (e.g. in cars). See also chapter 2.2.2 "Accessories").

Earth protection

The ***ADwin-Gold*** device **must be earth-protected**, in order to

– build a ground reference point for the electronic

– conduct interferences to earth.

Connect the GND plug, which is internally connected with the ground reference point and the aluminum enclosure, via a short low-impedance solid-type cable to the central earth connection point of your device.

Galvanic connection

The power supply cable is the galvanic connection between the computer and the ***ADwin-Gold***.

The version with USB interface has a galvanic connection to the computer or where appropriate also via the power supply.

The data lines at the version with Ethernet interface are optically isolated, but the ground potentials are connected, because the shielding of the Ethernet connector (RJ-45) is connected to GND.

Excluding transient currents

Transient currents, which are conducted via the aluminum enclosure or the shielding, have an influence on the measurement signal.
Please, make sure that the shielding is not reduced, for instance by taking measures for bleeding off interferences, such as connecting the shielding to the enclosure just before entering it. The more frequently you earth the shielding on its way to the machine the better the shielding will be.

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

BNC cables

The shielding of BNC cables is normally used as differential ground and looses therefore the shielding effect. So BNC cables are influenced by interferences when differential measurements are executed. For signal and data transfer outside of an enclosure it is necessary to use twisted pair data transfer cables, whose channels are shielded, too.

Protection low voltage

The ***ADwin-Gold*** is externally operated with a protection low voltage of 10 V to 35 V; internally it is operated with a voltage of +5 V and ±15 V against GND. It is not life-threatening. For operation with an external power supply, the instructions of the manufacturer applies.

Ambient temperature

The ***ADwin-Gold*** is designed for operation in dry rooms with a room temperature of +5°C … +50°C and a relative humidity of 0 … 80 % (no condensation, see Annex).

Chassis temperature

The temperature of the chassis (surface) must not exceed +60°C, even under extreme operating conditions – e.g. in an enclosure or if the system is exposed to the sun for a longer period of time. You risk damages at the device or not-defined data (values) are output which can cause damages at your measurement device under unfavorable circumstances.

4 Initialization of the Hardware

If you start **initializing** do not connect any cables to the ***ADwin-Gold*** before you have executed the **following steps**:

– Carry out completely the installation of the drivers and the power supply at the computer or notebook (see manual: ***"ADwin*** Driver Installation").

– connect the ***ADwin-Gold*** only with the computer or notebook (s.b.).

– Read chapter 5 "Inputs and Outputs" in this manual.

– Begin now with the connection of the inputs and outputs.

Please take into account that there is a galvanic isolation between the ***ADwin-Gold*** system and the computer via power supply cable, USB and Ethernet lines (see chapter 3, section "Galvanic connection").

Providing the power supply

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

Power supply

The power supply connection of the ***ADwin-Gold*** with 12V (see Annex, Technical Data) is made via the built-in connector, at left next to the power switch or above the GND plug (see Fig. 4). Connect the 3-pin subminiature connector there. For the pin assignment see the following picture:

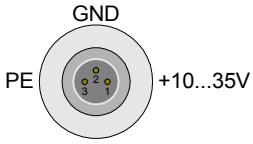


Fig. 3 – Power supply connector (male)

For using the system with an external power supply unit you need the subminiature connector described above. The connector is provided by the following manufacturer under the article number 712 299-0406-00-03 (Series 712):

Franz Binder GmbH + Co. elektrische Bauelemente KG
Rötelstrasse 27
74172 Neckarsulm,
Phone: ++49-7132 / 325-0
www.binder-connector.de
.Hypertext.message URL http://www.binder-connector.de;

When using the system with a notebook, power has to be supplied by a separate power supply, (see chapter 2.2.2 on page 5). Please pay attention to the fact that it is sufficiently dimensioned.

If using current-limiting power supplies, please pay attention to the fact, that after power-up the current demand can be a multiple of the idle current. More detailed information can be found in the Technical Data (Annex).

In case of a power failure all data which have not been saved are lost. Not-defined data (values) can under unfavorable circumstances cause damages to other equipment.

Connection

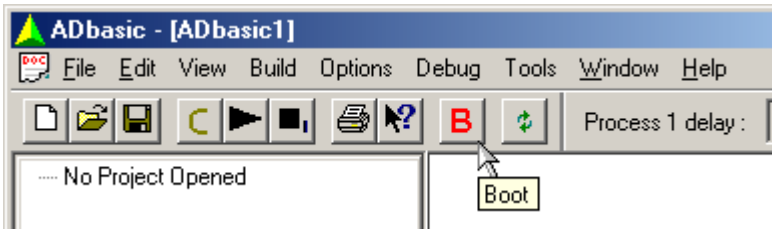
If you have completed the installation of the ***ADwin*** drivers and the configurations in the ***ADbasic*** menu "`Options\Compiler`", then connect the USB or Ethernet data transfer cables and the power supply cable. Then start the computer.

Power-up

In order to avoid switching off the system inadvertently, the switch is equipped with a blocking device. Pull the switch a little bit, then pull it into the direction "Power". Now the device is switched on and the LED lights up in red.

Booting

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



The display in the status line: "`ADwin is booted`" shows that the operating system has been loaded appropriately and that via ***ADbasic*** the ***ADwin*** system has been connected. At the same time the flashing of the LED (green colored now) at the ***ADwin-Gold*** shows that it is ready for operation.

Programs with *ADbasic*

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

Connectors

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-Gold*** system.

Open-ended inputs can cause errors - above all in an environment where interferences may occur. For your safety, set the inputs which you do not use to a specified level (for instance GND) and also connect them as close to the connector as possible. Don't connect open ended cables to the inputs; open ended cables may cause spikes at the inputs.
An exception is the event input, which has already an internal pull-up resistance (10 kΩ).

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 only if extremely time-critical or special tasks require to do so. (See also ***ADbasic manual***).

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

The pin assignment of CONN. 1 and CONN. 2 (Gold-D: DIO00-15 and DIO16-31) is illustrated on page 14.

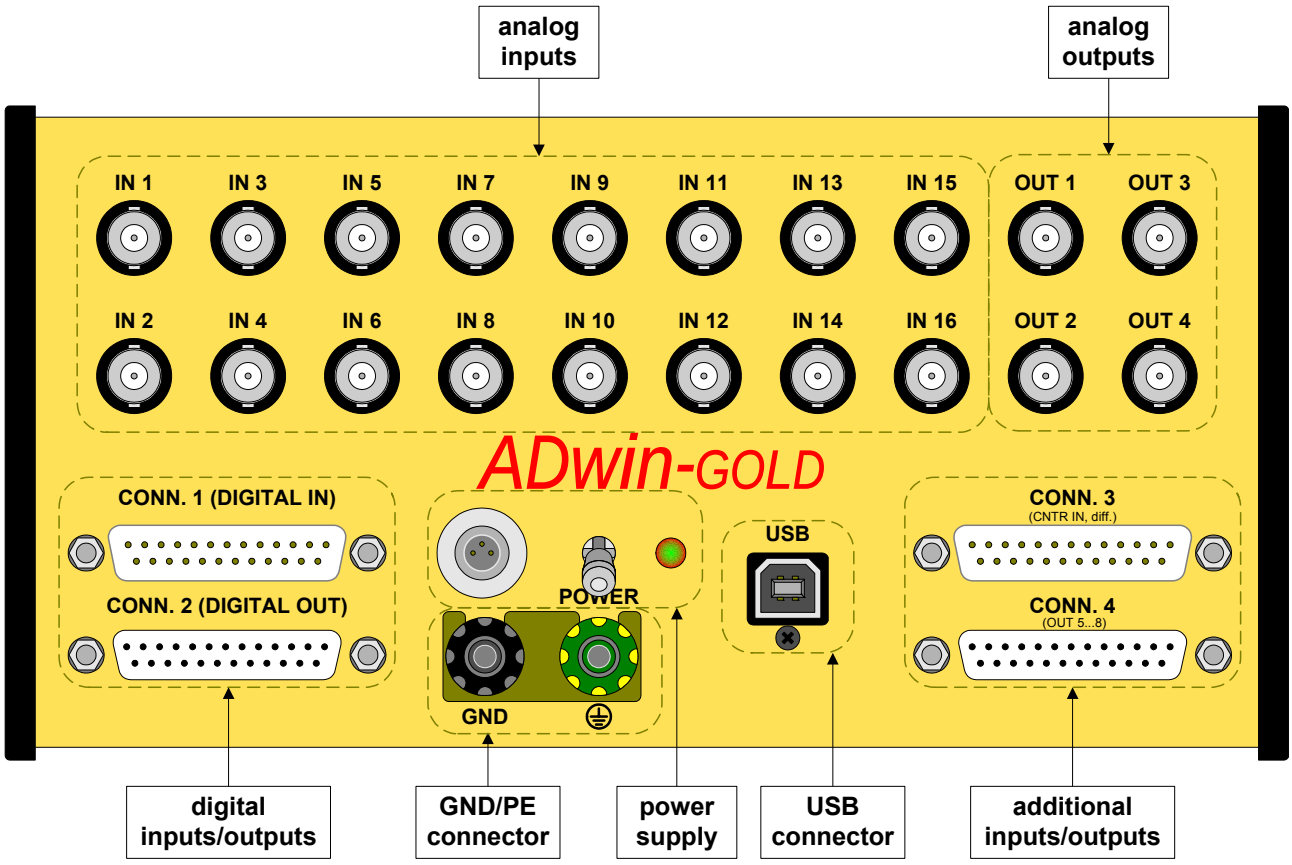


Fig. 4 – Schematic of ***ADwin-Gold*** (USB version)

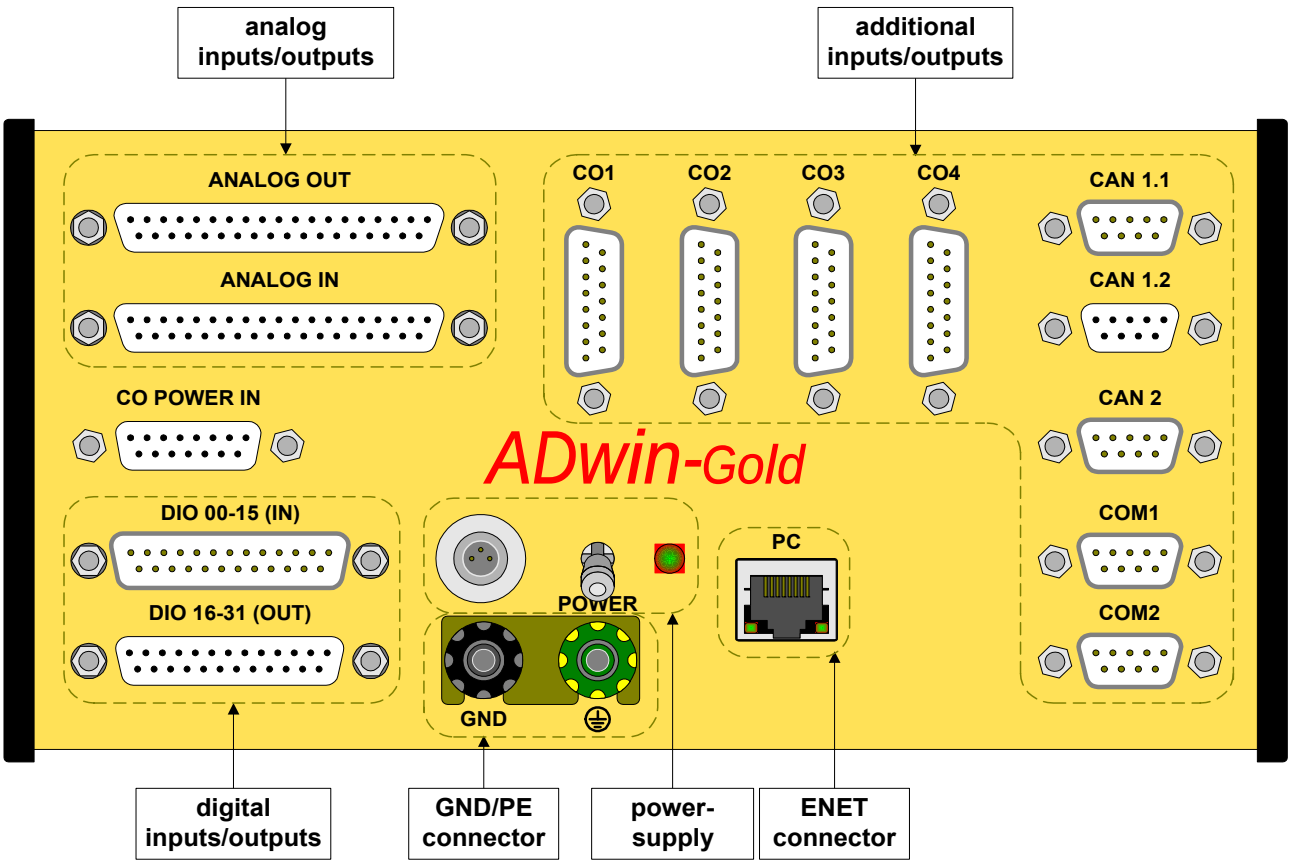


Fig. 5 – Schematic of ***ADwin-Gold-D*** (ENET version)

5.1 Analog Inputs and Outputs

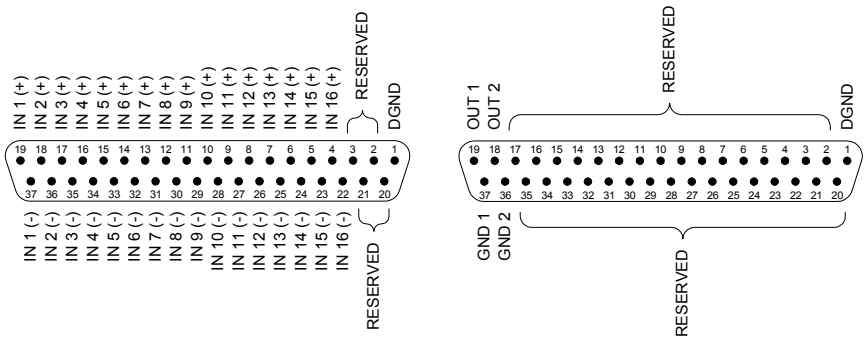
In order to operate the system without any interferences, **isolated BNC connectors** are necessary. Otherwise there will be the danger of damages caused by ESD or short circuits at the inputs. This will be the case when using not isolated BNC T-pieces.

The ***ADwin-Gold*** device has to be connected to earth, in order to execute measurement tasks without any interferences. Connect the GND plug via a low-impedance solid-type cable with the central earth connection point of your device.

The power supply from the power adapter at the computer also connects the earth of the ***ADwin-Gold*** system with the earth of the computer. If you do not operate the PC and the ***ADwin-Gold*** system in the same place, you should not use the power supplied by the PC but an external power supply unit which is earth-free, in order to avoid influences by different ground reference potentials.

In addition to the description of the inputs and outputs you will find notes below for the conversion of digits into voltage values and for the input settings of the analog inputs.

The pin assignment of the analog channels for ***ADwin-Gold-D*** is shown in Fig. 6.



ANALOG IN (female) **ANALOG OUT** (male)

Fig. 6 – Pin assignment of analog channels with Gold-D option

5.1.1 Analog Inputs

Multiplexer

The system has 16 analog inputs IN1 … IN16. The inputs with odd numbers (1, 3, … 15) are allocated to multiplexer 1, those with even numbers (2, 4, … 16) to multiplexer 2. The output of each multiplexer is connected to both a 14 bit-ADC and a 16 bit-ADC (see also Block diagram of the ADwin-Gold, page 4).

Differential

The analog inputs are differential. For each of the measurement channels there is a positive and a negative input, between them the voltage difference is measured (but not free of potential). Both, the positive and negative input have to be connected.

The inputs are equipped with male BNC-plugs, which are arranged in 2 rows; the Gold-D option has the inputs connected to the DSub-connector ANALOG IN. At the BNC-plugs, the positive input is the inner conductor, the negative input is the outer conductor.

Please note, that the inputs do need a mass connection between the system's GND-plug and the signal source. This is in addition to the connections to the positive and negative input.

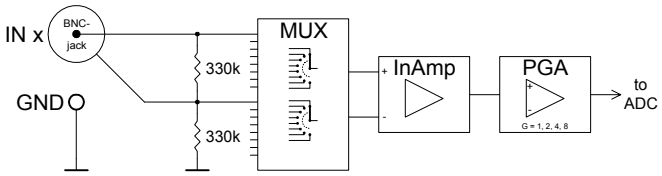


Fig. 7 – Input circuitry of an analog input

16-bit and 14-bit measurements

You can convert the signals at the multiplexer outputs optionally with a 14-bit or a 16-bit analog-to-digital-converter (ADC), (see Fig. 2 "Block diagram of the ADwin-Gold"). You are measuring with

– the 14-bit ADC very fast (max. 0.5µs, resolution 1.221mV)

– the 16-bit ADC very accurately (max. 5µs, resolution 305µV).

ADC instruction

The instructions `ADC()` for the 16-bit ADC and `ADC12()` for the 14-bit ADC execute a **complete measurement** with one of the ADCs on the analog input. The ADC instructions consider for instance the settling of the multiplexer and assure perfect measurements (see also ***ADbasic*** manual).

Please pay attention to a low internal resistance of the power supply unit (of the input signals), because it may have influence on the measuring accuracy. If this is not possible:

– Depending on the output resistance a linear error is caused.
  You can compensate this by multiplying the measurement value with a corresponding factor and get a sort of recalibration.

– From approx. 3kΩ upwards the multiplexer settling time extends.
  The waiting time defined in the standard instructions ADC and ADC12 is then too short, so that imprecise values are recalled. In this case please use the instructions described in chapter 5.3.1.

5.1.2 Analog Outputs

The system has 2 analog outputs (OUT1, OUT2) with BNC-plugs; with Gold-D option the outputs are located on the DSub connector ANALOG OUT (see Fig. 6). A digital-to-analog converter (DAC) is allocated to each of the outputs.

DAC instruction

The standard instruction DAC (*number, value*) checks each of the values if it exceeds or falls below of the 16-bit value range (0...65535). If the value is in the 16-bit value range, the indicated value is output on the output *number*. If it is not in the value range the maximum or minimum values are output, (see also ***ADbasic*** manual).

5.1.3 Calculation Basis

Voltage range

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

Allocation of digits to voltage

The 65536 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

– 65535 digits correspond to the maximum positive voltage

The value for 65536 digits, exactly 10 Volt, is just outside the measurement range, so that you will get a maximum voltage value of 9.999695V for the 16-bit conversion and a voltage value of 9.998779V for the 14-bit conversion.

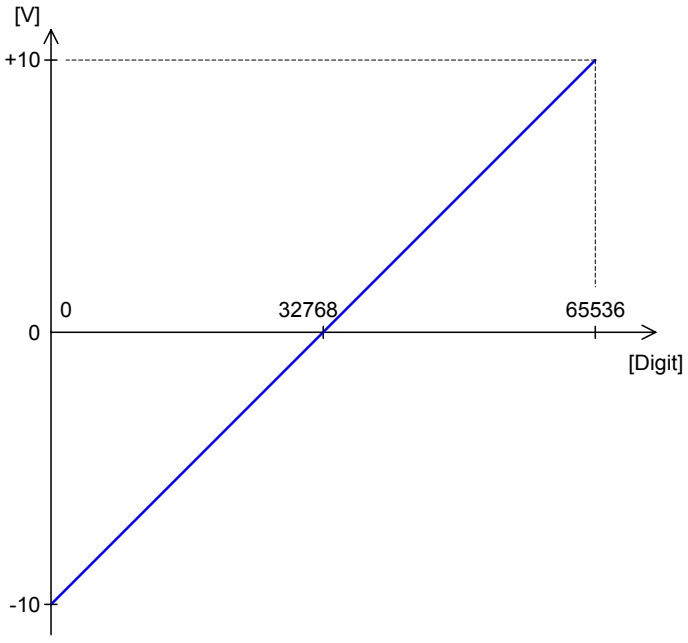


Fig. 8 – Zero offset in the standard setting of bipolar 10 Volt

Zero offset U_{OFF}

In the bipolar setting you will get a zero offset, also called offset U_{OFF} in the following text.
For the voltage range of −10V … +10V applies:

U_{OFF} = −10V

Gain factor k_v

The ***ADwin-Gold*** has a programmable gain (PGA), with which you can amplify the input voltage by the factors 1, 2, 4, and 8. At the same time the measurement range gets smaller by the corresponding gain factor k (see Annex "Technical Data").

Please note that upon applications with k_V >1 the interference signals are amplified respectively.

Quantization level U_{LSB}

The quantization level (U_{LSB}) is the smallest digitally displayable voltage difference and is equivalent to the voltage of the least significant bit (LSB). It is different for the two ADCs:

– 16-bit ADC: U_{LSB} = 20V / 2^{16} = 305.175µV

– 14-bit ADC: U_{LSB} = 20V / 2^{14} = 1220.7µV

The measured 16-bit value of the ADC is returned in the lower word of the register. A DAC value, which is to be output, has to be available there.

Bit No.	31…16	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
32-bit-memory	0	16-bit value of the 16-bit ADC / DAC in the lower word															
	0	14-bit value of the 14-bit ADC in the lower word														0	0

Fig. 9 – Storage of the ADC/DAC bits in the memory

In order to compare the measurement values of the 14-bit ADC with the values of the 16-bit ADC, the converted value is written left-aligned into the lower word of the register at the 14-bit ADC. Therefore the lower 2 bits are always 0 (zero).

The 16384 digits of the 14-bit ADC are mapped to the 65536 digits of the 16-bit ADC. Thus 4 digits of the 16-bit ADC are equivalent to one digit of the 14-bit ADC.

Therefore the following equations can be used for both ADC types:

Conversion Digit ⇆ 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 (14-bit and 16-bit):

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

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

Tolerance Ranges

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

INL

– The integral non-linearity (INL) defines the maximum deviation from the ideal straight line over the whole input voltage range.

DNL

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

5.2 Digital Inputs and Outputs

Digital inputs/outputs

On two 25-pin D-SUB sockets (DIO 00…DIO 31) there are 32 digital inputs or outputs. They are programmable in groups of 8 as inputs or outputs.

The digital inputs are TTL-compatible and not protected against over voltage. Do not use pins marked as "reserved". They are planned for changes and expansions and can cause damages to your system if you do not pay attention to this fact.

Trigger input (EVENT)

The ***ADwin-Gold*** is equipped with an external trigger input (EVENT). With this trigger input processes are triggered by an external signal (trigger) with rising edge and can completely and immediately be processed, (see also ***ADbasic*** manual, chapter: "Program Structure").

Power-up configuration

After power-up of the device, all connections are configured as inputs.

`CONF_DIO(12)`

The instruction `CONF_DIO(12)` configures DIO 15:00 as digital inputs and DIO 31:16 as digital outputs (see Fig. 10).

Only in this configuration will you be able to totally access the inputs and outputs with the instructions
`DIGIN`, `DIGIN_WORD`, `DIGOUT_WORD`, `SET_DIGOUT`, `CLEAR_DIGOUT`.

About programming under other configurations the following chapter will give you more detailed information: chapter 5.3 "Time-Critical Tasks" (see also ***ADbasic*** manual and tutorial).

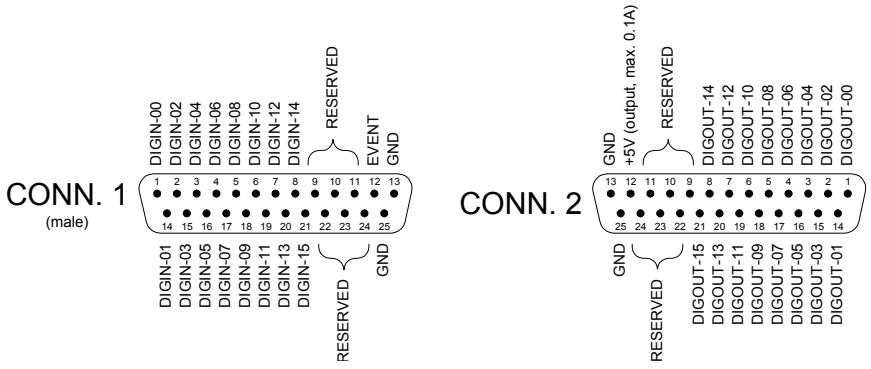


Fig. 10 – Pin assignment with the configuration `CONF_DIO(12)`

5.3 Time-Critical Tasks

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

Contrary to the standard instructions ADC(), ADC12() and DAC() the instructions for direct access **do not have any test routines.** Before you use them we recommend to learn more about time sequences, program structures and functions sequences in an ADC.

5.3.1 Analog Inputs and Outputs

ADC() and ADC12()

The standard instructions ADC() and ADC12() consist of a sequence of several instructions (see below). They need a certain time for execution. The execution time is mostly determined by the settling time of the multiplexer and the conversion time.

```
SET_MUX()
…                       'wait for settling of the
                        'multiplexer
START_CONV()
WAIT_EOC()              'wait for end of conversion
READ_ADC()              'or READ_ADC12() at ADC12()
```

Program structure

You can use (or extend) the waiting times caused by the standard instructions for other purposes. If you apply these instructions skillfully you may be able to execute faster measurements.

It is important to set the START_CONV() instruction in a sufficient time-delay from the SET_MUX() instruction, in order to consider the multiplexer settling time (see also ***ADbasic*** manual, Instruction Reference).

Use the waiting times for instance for arithmetic operations and save CPU time:

- Settling time of the multiplexer: At a maximum voltage jump of 20 Volt it is 6.5 µs (max.) for the 16-bit ADC and 2.5 µs for the 14-bit ADC.
- Conversion time of the ADC: Its is 0.5 µs for the 14-bit ADC and 5 µs for the 16-bit ADC.

Direct Register Access

ADC

A measurement can be executed very fast, when you directly access the control and data registers of the ADC.

DAC

If you have made sure that at the analog outputs the values are within the range limits, you can write very quickly into one or more DAC registers with direct access to the hardware registers, and you can synchronously start the output, (see ***ADbasic*** manual).

The hardware addresses for the direct access to the control and data registers are described on the following pages.

Address [HEX]	Function	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Commentary
20400000	Set MUX 1: channels 1, 3, 5, …, 15	-	-	-	-	-	-	-	-	-	n	n	n	""nnn"" binary = 0...7 decimal, selected ch. = nnn +
	Set MUX 2: channels 2, 4, 6, …, 16	-	-	-	-	-	-	n	n	n	-	-	-	""nnn"" binary = 0...7 decimal, selected ch. = 2(nnn + 1
	Gain PGA 1	-	-	-	-	g	g	-	-	-	-	-	-	""gg"" binary = 0...3 decimal, selected gain = 2^{gg}
	Gain PGA 2	-	-	g	g	-	-	-	-	-	-	-	-	
20400010	Start conversion: ADC 1 (16-bit)	-	-	-	-	-	-	-	-	-	1	-	s	s = 0 : start conversion s = 1 : no effect
	Start conversion: ADC 2 (16-bit)	-	-	-	-	-	-	-	-	-	1	s	-	
	Start conversion: ADC 1 (14-bit)	-	-	-	-	-	-	-	-	s	1	-	-	
	Start conversion: ADC 2 (14-bit)	-	-	-	-	-	-	-	s	-	1	-	-	
20400020	EOC status: ADC 1 (16-bit)	-	-	-	-	-	-	-	-	-	-	-	e	e = 0 : end of conversion e = 1 : conversion is running
	EOC status: ADC 2 (16-bit)	-	-	-	-	-	-	-	-	-	-	e	-	
	EOC status: ADC 1 (14-bit)	-	-	-	-	-	-	-	-	e	-	-	-	
	EOC status: ADC 2 (14-bit)	-	-	-	-	-	-	-	e	-	-	-	-	
20400030	Read out register: ADC 1 (16-bit)	-	x	x	x	x	x	x	x	x	x	x	x	x : result of conversion
20400040	Read out register: ADC 2 (16-bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400130	Read out register: ADC 1 (14-bit)	-	x	x	x	x	x	x	x	x	x	0	0	
20400140	Read out register: ADC 2 (14-bit)	-	x	x	x	x	x	x	x	x	x	0	0	
20400100	Read out register and start conversion: ADC 1 (16-bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400110	Read out register and start conversion: ADC 2 (16-bit)	-	x	x	x	x	x	x	x	x	x	x	x	
20400120	Read out register and start conversion: ADC 1 (14-bit)	-	x	x	x	x	x	x	x	x	x	x	x	
204001D0	Read out register and start conversion: ADC 2 (14-bit)	-	x	x	x	x	x	x	x	x	x	x	x	

Fig. 11 – Hardware addresses of the control and data registers for the **ADCs**

Address [HEX]	Function	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Commentary
20400010	Start conversion: All DACs synchronously	-	-	-	-	-	-	-	1	1	s	1	1	s = 0 : start conversion s = 1 : no effect
20400050	Write only to the register: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted
20400060	Write only to the register: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x	
20400070	Write only to the register: DAC 3 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400080	Write only to the register: DAC 4 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400090	Write only to the register: DAC 5 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
204000A0	Write only to the register: DAC 6 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400190	Write only to the register: DAC 7 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
204001A0	Write only to the register: DAC 8 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400200	Write to the register and start conversion immediately: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted
20400210	Write to the register and start conversion immediately: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x	
20400220	Write to the register and start conversion immediately: DAC 3 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400230	Write to the register and start conversion immediately: DAC 4 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400240	Write to the register and start conversion immediately: DAC 5 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400250	Write to the register and start conversion immediately: DAC 6 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400260	Write to the register and start conversion immediately: DAC 7 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	
20400270	Write to the register and start conversion immediately: DAC 8 (*Gold-DA*)	-	x	x	x	x	x	x	x	x	x	x	x	

Fig. 12 – Hardware addresses of the control and data register for the **DACs**

5.3.2 Digital Inputs and Outputs

After power-up of the device all 4 connection groups are configured as inputs; this corresponds to the instruction CONF_DIO(0). The following table shows how the inputs and outputs (IN, OUT) are configured when you use the value of the first column as instruction argument.

CONF_DIO()	DIO31:24	DIO23:16	DIO15:08	DIO07:00
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
19	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
Applicable instructions:	DIGOUT_WORD, CLEAR_DIGOUT, SET_DIGOUT		DIGIN_WORD, DIGIN	
Instruction is applicable for DIOnn, at	Configuration "OUT"		Configuration "IN" At configuration "OUT" the register contents of this byte is returned	

Fig. 13 – Overview of the configuration with CONF_DIO

Please pay attention to the following restriction:

Only if the inputs/outputs are configured with CONF_DIO(12) (see pin assignmenton page 14) will you be able to fully access the inputs/outputs with the instructions
DIGOUT_WORD, SET_DIGOUT, CLEAR_DIGOUT, DIGIN_WORD, DIGIN.
For any other configuration you have to read out or write into the corresponding hardware register (see instructions PEEK and POKE in the ***ADbasic*** manual).

Address [HEX]	Function	Bit 31:16	15:10	9	8	7	6	5	4	3	2	1	0	Commentary
204001E0	Configure DIO07:00	0	0	0	0	0	0	0	0	-	-	-	c	c = 0 : inputs c = 1 : outputs
	Configure DIO15:08	0	0	0	0	0	0	0	0	-	-	c	-	
	Configure DIO23:16	0	0	0	0	0	0	0	0	-	c	-	-	
	Configure DIO31:24	0	0	0	0	0	0	0	0	c	-	-	-	
204000B0	Input registers DIO15:00	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value read in
204001B0	Input registers DIO31:16	-	x	x	x	x	x	x	x	x	x	x	x	
204001C0	Output registers DIO15:00	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be output
204000C0	Output registers DIO31:16	-	x	x	x	x	x	x	x	x	x	x	x	

Fig. 14 – Hardware addresses of the control and data registers for the digital inputs / outputs

6 Calibration

6.1 General Information

The 2 digital-to-analog (DAC, optional 8) and the 4 analog-to-digital (ADC) converters of the ***ADwin-Gold*** systems 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.

You calibrate the system with the program <GoldCalib.exe>; at standard installation the path is <C:\ADwin\Tools\ADwin-Gold>.

Tools

The following tools are necessary for the calibration:

– A digital multimeter (DMM) with an accuracy of
  • 30µV when using 16-bit converters
  • 120µV when using 14-bit converters
– A reference voltage source with an accuracy of
  • 30µV when using 16-bit converters
  • 120µV when using 14-bit converters
– connecting cables from the inputs/outputs to the reference voltage source and to the measurement device (recommended: BNC cables).

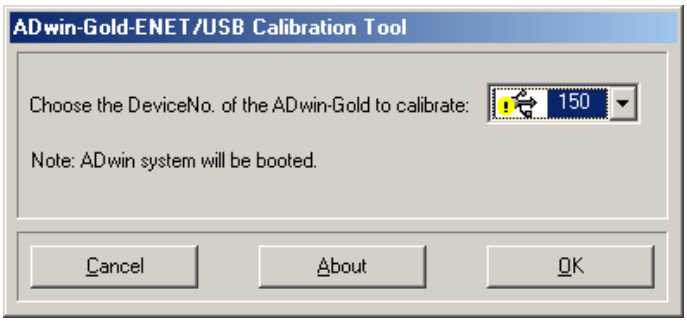
6.2 Calibrating

Connect your ***ADwin-Gold*** system with the computer and configure it according to the program <ADconfig.exe>.

Calibration has to be made when the ***ADwin-Gold*** system reaches its operating temperature. With a power-up temperature of the device of approx. 20 to 25 degrees Celsius (room temperature), the system reaches the operating temperature approx. 30 minutes after power-up.

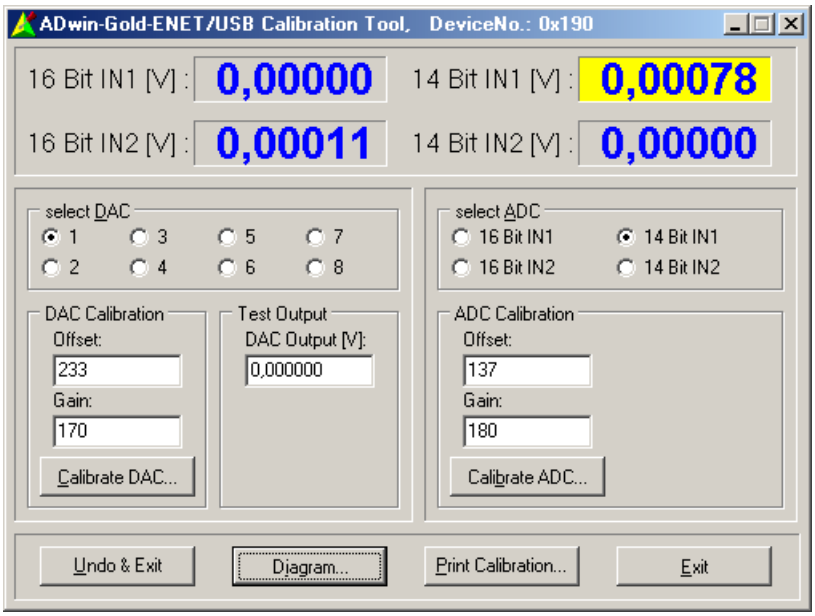
Step 1

Start the **calibration program** <GoldCalib.exe>. The window "ADwin-Gold-ENET/USB Calibration Tool" appears.



Choose the device number of the system you want to calibrate and confirm by clicking on "OK".
You will get a warning when you haven't chosen an ***ADwin-Gold*** system or one of an older firmware version. You can ignore the warning with "Yes" or return to the previous window with "No".

An **overview window** appears. In the header line the device number you have selected is shown.



The upper field shows the current measurement values at the inputs IN1 and IN2, measured each by the 16-bit and the 14-bit ADC.

Select in the lower field left the DAC which you wish to calibrate and at right the ADC. The measurement value at the selected ADC is highlighted. Below you will find the calibration settings for `Offset` and `Gain` of the DAC and the ADC. There you can directly enter values. With "`Calibrate DAC`" or "`Calibrate ADC`" you start the calibration of the selected converter.
In the dialog box "`Test Output`" you can enter a voltage value, which is automatically output at the converter/output you have selected earlier.

Every input you make is immediately transferred to the ***ADwin-Gold*** system. If you close the program with "`Exit`", the new settings remain. With "`Undo&Exit`" you undo all inputs and you exit the calibration program (that means the original settings are transferred to the ***ADwin-Gold*** system).
"`Diagram`" displays in a graph the accuracy of the current calibration setting.
You print a protocol of the settings with "`Print Calibration`".

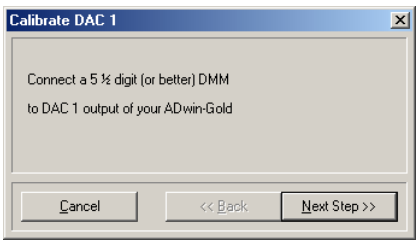
Calibrate the converters in the order you like (only with reference voltage source). The calibration of a converter is effected in 3 steps; you can switch between the windows of the steps by using the forward/backward buttons.

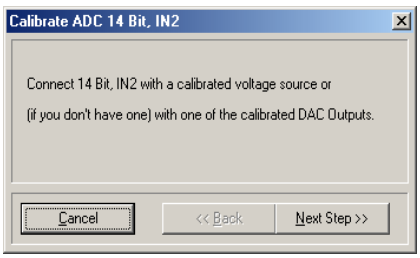
Calibration is also possible without reference voltage source, but it will not be so precise. Calibrate first the DACs and then the ADCs.

Step 2

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.:



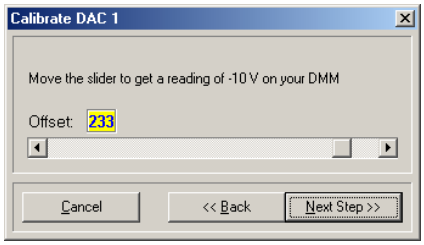


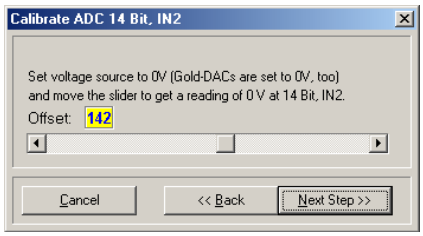
Connect a DMM to the selected output.

Connect the voltage source (or a calibrated DAC output) to the selected input.

Please note Fig. 4 "Schematic of ADwin-Gold (USB version)".
Select "Next Step >>".

2. Setting the offset





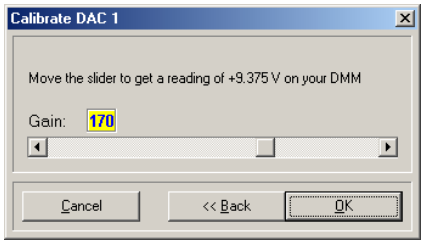
Adjust the offset value at the scrollbar in such a manner that your digital multimeter displays -10 V.

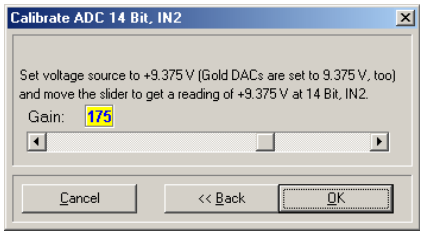
Set your voltage source to 0 V.The setting of the ADC to this value is made automatically.
Adjust the offset value at the scrollbar in such a manner that the setpoint at the ADC is displayed in the overview window.

Select "Next Step >>".

3. Setting the gain

:





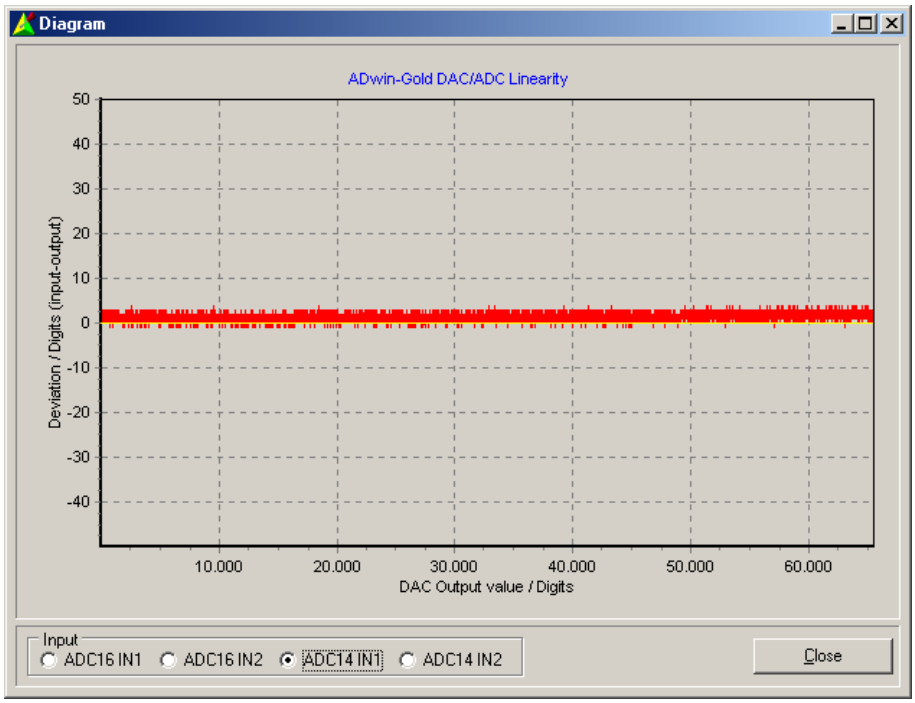
Adjust the offset value at the scrollbar in such a manner that your digital multimeter displays 9.375V.

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 scrollbar in such a manner that the setpoint at the ADC is displayed in the overview window.

The calibration for this converter has finished. Select "OK". Repeat step 2 for the other converters if necessary.

Step 3

With a diagram (button Diagram in the overview window) you can check the **accuracy** of the calibration. First connect any 2 outputs with the inputs IN1 and IN2. Select in the diagram one of the inputs and the corresponding converter.



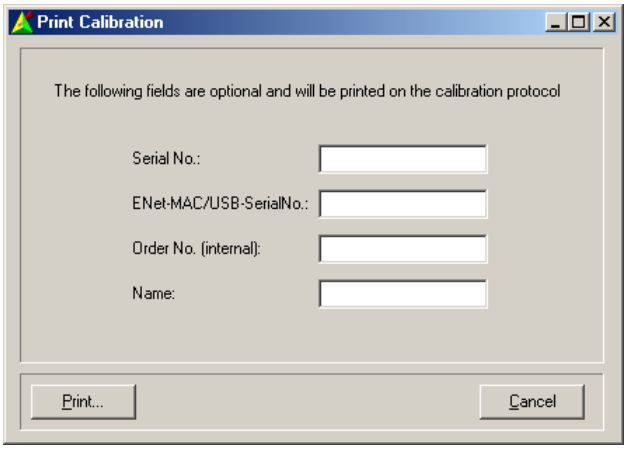
The program outputs the values 0...65535 digits on both DACs, compares them to the measured input values and displays the deviation in graphs.
The deviation should be less than 5 digits.

With `Close` you return to the overview window.

Step 4

With "`Print Calibration`" you can print a **protocol** of the specified calibration data.

In the open window you can enter different information which will be presented in your printout (for a later allocation to the protocol). With "`Print`" you start printing; the program automatically returns to the overview window.
In the protocol you will find the calibration settings of all inputs and outputs for gain and offset as well as the date of print.



The calibration is finished.

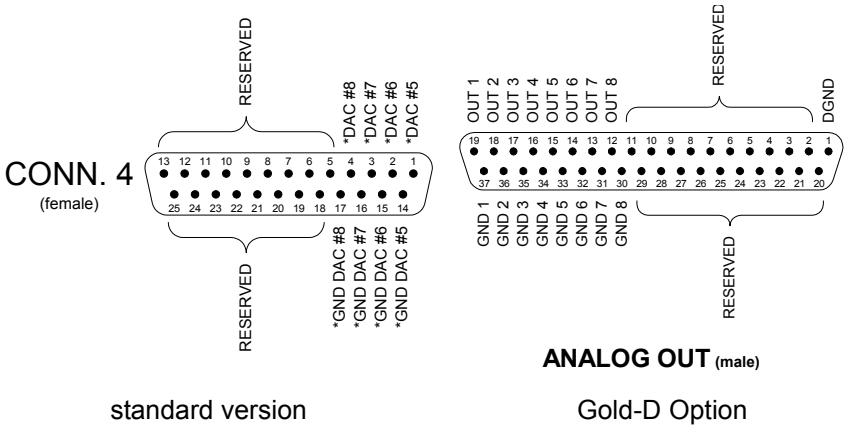
Step 5

7 DA Add-On

Connectors

With the DA add-on you will get **6 additional analog outputs** with a resolution of 16 bit (and a DAC each).

In the standard version two of these outputs go from DAC 3 and DAC 4 to the BNC plugs OUT 3 and OUT 4. The other 4 outputs connect DAC 5...DAC 8 to the pins 1...4 and 14...17 of the 25-pin D-SUB socket CONN4 (see figure). With the Gold-D option all additional outputs are connected to the pins of the DSub socket ANALOG OUT.



standard version Gold-D Option

Fig. 15 – Pin assignment of the DA add-on

Programming and calibration

You program and calibrate the additional DACs like the DAC 1 and DAC 2 (see chapter 5.1, chapter 6 and ***ADbasic*** manual).

8 CO1 Counter Add-On

The technical data of the counter add-on *CO1* is described in the annex A.1.

8.1 Hardware

Counter

The counter add-on *CO1* (ordering option *Gold-CO1*) has four 32-bit up/down counters with four-edge-evaluation. You can configure and read out the counters individually as well as all together. (The block diagram shows the design of a single counter).

Latch A and B

The counters can be internally or externally clocked and are read out via accompanying latches. All counters have each a Latch A and a Latch B. The counter values can be cleared or transferred in a latch by using programming commands or (if configured) when there is an external signal at CLR/LATCH.

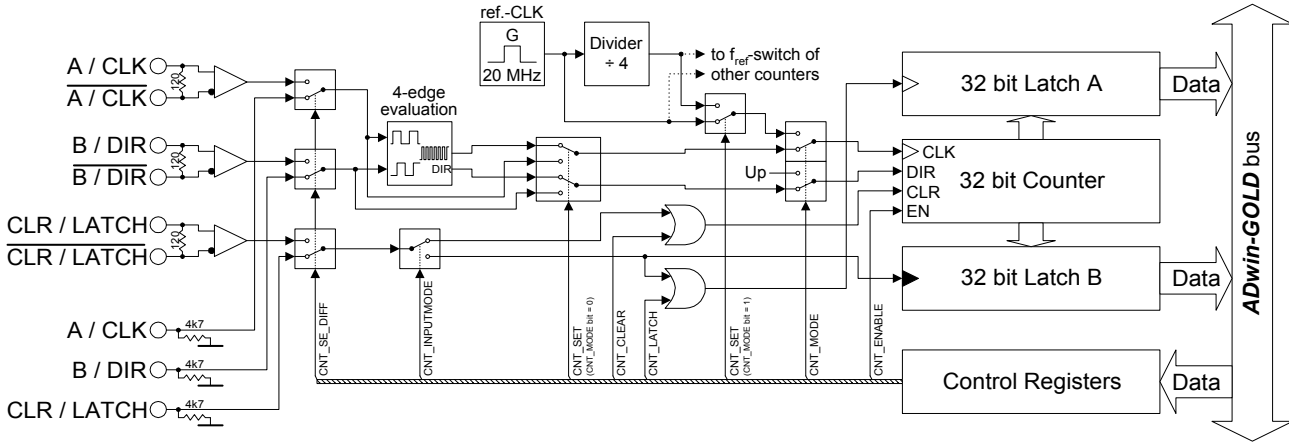


Fig. 16 – Block diagram of the *Gold-CO1* counter add-on

There are the following operating modes: event counting (external clock) and pulse width measurement (internal clock); see also chapter 8.3 / 8.4:

External clock input

a) **Event counting:** Incrementing/decrementing of the counter is caused by external square-wave signals at the inputs A/CLK and B/DIR. A positive edge at CLR/LATCH either sets the counter to zero (CLR) or copies the counter values into the latch (LATCH).

The following modes are possible:

1. **Clock and direction:** A positive edge at CLK increments or decrements the counter values by one. The signal at DIR determines the counting direction (0 = decrement; 1 = increment).
2. **Four edge evaluation:** Every edge of the signals (phase-shifted 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 quadrature encoders.

Internal clock input

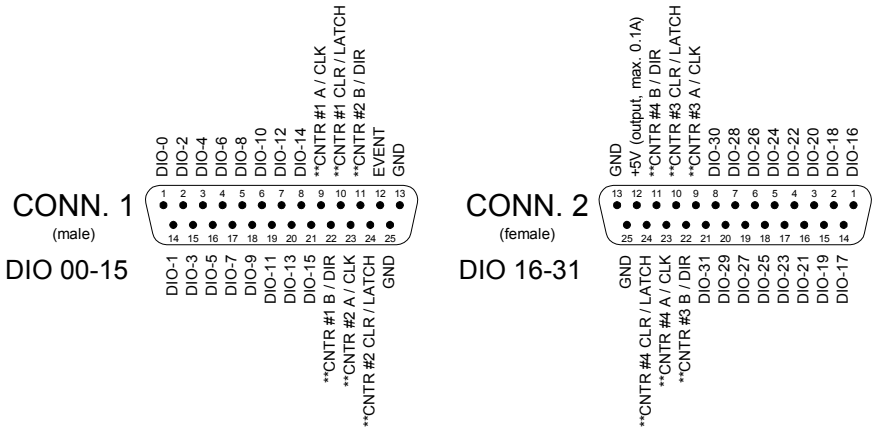
b) **Pulse width measurement:** Incrementing/decrementing of the counter is caused by an internal reference clock generator; a signal frequency of 5 MHz or 20 MHz can be used. The square-wave signal at CLR/LATCH is evaluated: With every positive edge the counter values are written to latch A, with a negative edge to latch B.

You can calculate:

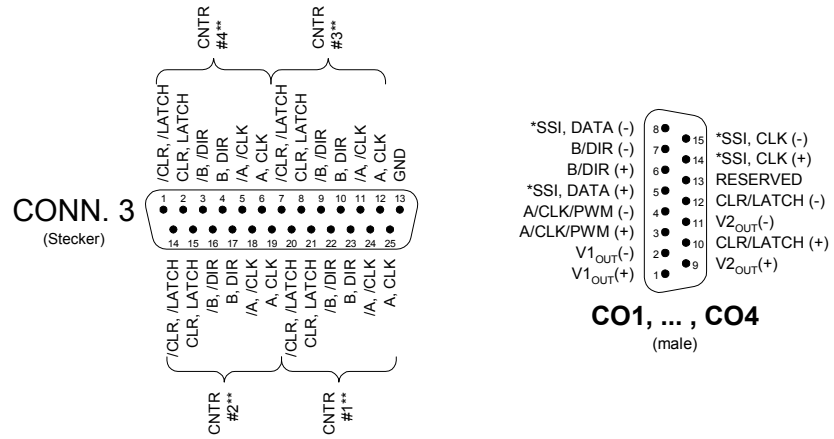
1. the **period duration** of the input signal at CLR/LATCH from the values in latch A or latch B.
2. the **pulse width** and **pause time** from the values in latch A and latch B

The counters are controlled by ***ADbasic*** instructions via a control register (instructions, see table in chapter 8.2).

At the inputs A/CLK, B/DIR and CLR/LATCH TTL-alike signals are necessary. More details and limit values can be found in the "Technical Data".



Counter inputs with TTL operation mode for Gold / Gold-D
(single-ended; mode not available with Rev. B2)



with Gold only

with Gold-D only

Counter inputs with differential operation mode

Fig. 17 – Pin assignment of the CO1 add-on

In any case you have to set the input operation mode with the instruction `CNT_SE_DIFF`. This is done in pairs, i.e. the counters 1 and 2 together and the counters 3 and 4 together (see also ***ADbasic*** manual).
With Rev. B2 differential operation mode can be set only, TTL operation mode (single ended) is available from Rev. B3.

Although all inputs for the *CO1* add-on have a pull-down resistor, not-connected inputs can cause errors in an environment which is not protected against interferences. If you do not use a counter input, connect for safety reasons both lines of the (differential) input to a specified potential: Connect the positive input to +5V and the negative input to GND.

On the option Gold-D you can – via the connector CO Power in – supply a voltage, which is then available at the connectors CO1…CO4, e.g. for external Inkrementalgeber.
Please note: All minus inputs $V1_{in}$ (-) are galvanically connected to GND via a common line; the minus inputs $V2_{in}$ (-) have such a common connection, too.

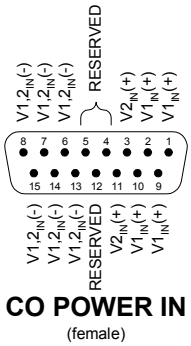


Fig. 18 – Pin assignment counter voltage supply (Gold-D)

8.2 Software

The functions necessary for accessing the counters can be found in the include file:

<ADWGCNT.INC>

Therefore programming has to start with the include file, so that you can use the instructions in the following table. The instructions are described in the ***ADbasic*** manual.

Instruction	Function
CNT_CLEAR( ) *	Clear counter
CNT_ENABLE( )	Disable or enable counter (please note the already running counters)
CNT_GETSTATUS(#)	Read out status register (# = counter no. 1…4)
CNT_INPUTMODE( )	Set CLR/LATCH input to CLR or LATCH mode
CNT_LATCH( ) *	Latch counter values into Latch A
CNT_MODE( )	Use external clock input or internal reference clock
CNT_SE_DIFF( )	Set clock inputs to differential or single-ended (as pairs)
CNT_SET( )	In combination with CNT_MODE( ): Set counter mode or length of the internal reference clock
CNT_READ(#)	Read out counter values and transfer them to Latch A (# = counter no. 1…4)
CNT_READLATCH(#)	Read out Latch A (triggered by positive edge), (# = counter no. 1…4)
CNT_READFLATCH(#)	Read out Latch B (triggered by negative edge), (# = counter no. 1…4)

* These functions are reset after they have been executed. All other functions are reset by opposing functions.

Fig. 19 – Instructions of the *Gold-CO1* counter add-on

With the instructions in the table matrix you are always effecting **all** counters (except CNT_READ…). Therefore pay attention to the fact which bits you are setting or deleting. You will be able to effect every counter individually or all together.

Sequence of instructions

Please configure the counters according to the following order:

1. Disable specified counter (CNT_ENABLE)
2. Set operating mode (CNT_MODE, CNT_SET, CNT_INPUTMODE, CNT_SE_DIFF)
3. Clear counter (CNT_CLEAR)
4. Enable counter (CNT_ENABLE)

For further processing of the values in the ***ADbasic*** program, transfer the values into the latch register and read them out there.

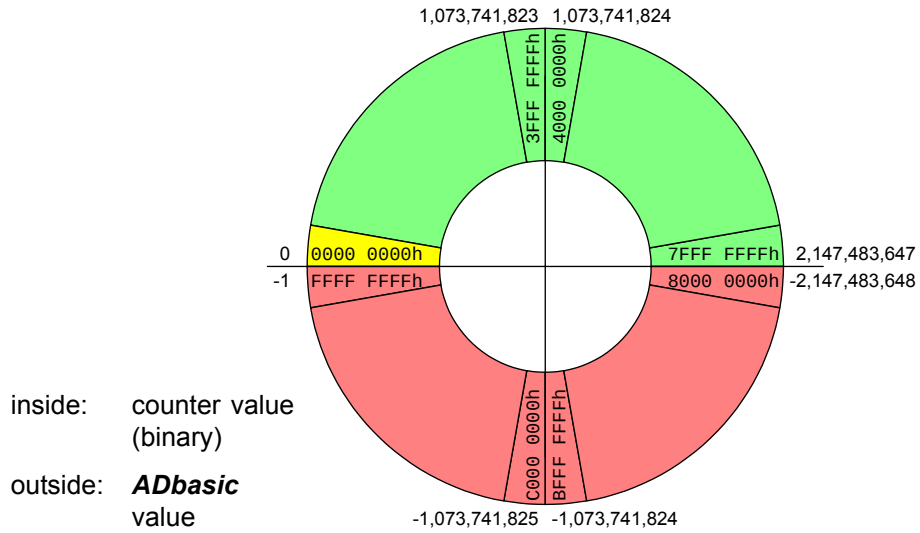
Please pay attention to the fact that the `CNT_SET` instruction depends on the `CNT_MODE` instruction.

If you disable or enable a specified counter, then you also enable the running counters (= set bits). If you do not set the bits of these counters (unintentionally), they will be disabled.

8.2.1 Evaluation of the Counter Contents

The binary counters of the CO1 add-on generate 32-bit values, which are interpreted by ***ADbasic*** as numerical values according to the model of the circle below: The most significant bit (MSB) is interpreted as a sign, the highest positive number (2^{31}-1) follows the highest negative number (-2^{31}) and the lowest positive number (0) follows the highest negative number (-1).

Circle



inside: counter value (binary)

outside: ***ADbasic*** value

Fig. 20 – Circle for the interpretation of counter values

Please pay attention to the following rules for programming:

a) Process the read 32-bit value only with variables of the type `INTEGER` or `LONG`. ***ADbasic*** then keeps internally the read bit pattern unmodified and automatically considers the transition from the positive to the negative range of numbers. Then you get:

Count direction

b) 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 values.

"Overflow"

For programming please remember that an "overflow" between the reading out of two counts - i.e. the current counter value "laps" the last counter value which has been read out - 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.

Example programs

You will find several example programs for the *CO1* add-on in the directory <`C:\ADwin\ADbasic3\samples_ADwin_Gold`> (standard installation).

8.3 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) **Clear**

– latch the counter values into latch register A (LATCH). **Latch**

8.3.1 Clock and Direction

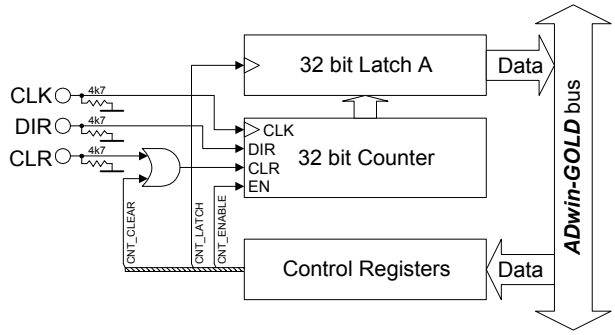


Fig. 21 – Block diagram of the CO1 add-on in the mode "clock and direction"

Every positive edge of a square-wave signal at the CLK input (clock) is counted (incremented or decremented) up to a maximum frequency of 20 MHz. The direction is derived from a high signal (count up) or low signal (count down) at the DIR input (direction); This signal can be static, for a fixed count direction, or dynamic, for changing directions.

Programming example

`INIT:`	initialization ...
`. . .`	
`CNT_ENABLE(0)`	disable counters
`CNT_CLEAR(1)`	clear counter(s)
`CNT_MODE(0)`	external clock input (CLK)
`CNT_SET(1)`	activate mode "clock and direction"
`CNT_INPUTMODE(0)`	input CLR/LATCH as CLR input
`CNT_SE_DIFF(0)`	select single-ended inputs
`CNT_ENABLE(1)`	enable counter(s)
`EVENT:`	event loop ...
`CNT_LATCH(1)`	current counter value to latch A
`CNT_READLATCH(1)`	read out latch A
`. . .`	evaluate counter value in process

8.3.2 Four Edge Evaluation

This mode determines clock and direction of two signals, which are phase-shifted by 90 degrees to the inputs A and B. The count direction is determined by the temporal sequence of the rising and falling edges of the two input signals.

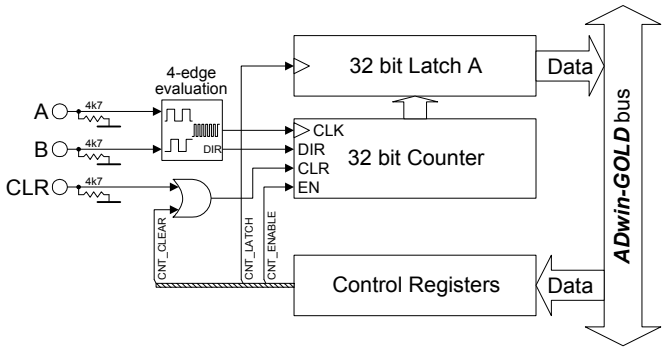


Fig. 22 – Block diagram of the *CO1* add-on in the mode "four edge evaluation"

Please note:

- The counter counts 4 edges in one cycle of the A/B signal.
- 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.
- The time 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 will have an effect on the maximum input frequency. If it differs from 90 degrees, the maximum input frequency of 5 MHz decreases for instance to 45 degrees at 2.5 MHz.

Programming example

`INIT:`	initialization ...
`. . .`	
`CNT_ENABLE(0)`	disable counters
`CNT_CLEAR(1)`	clear counter(s)
`CNT_MODE(0)`	external clock input (CLK)
`CNT_SET(0)`	activate mode "four edge evaluation"
`CNT_INPUTMODE(0)`	input CLR/LATCH as CLR input
`CNT_SE_DIFF(0)`	select single-ended inputs
`CNT_ENABLE(1)`	enable counter(s)
`EVENT:`	event loop ...
`CNT_LATCH(1)`	current counter value to latch A
`CNT_READLATCH(1)`	read out latch A
`. . .`	evaluate counter value in process

8.4 Operating Mode Impulse Width and Period Width 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.

Reference clock generator

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

8.4.1 Period Duration Measurement

All four counters can execute period duration measurements.

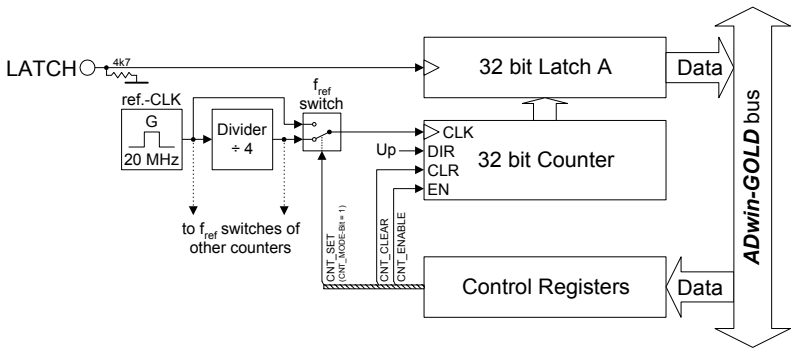


Fig. 23 – Block diagram of the *CO1* add-on in the mode "period duration measurement"

In this mode, the counter values are 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:                    initialization ...
. . .
CNT_ENABLE(0)            disable counters
CNT_CLEAR(1)             clear counter(s)
CNT_MODE(1)              mode "internal reference clock" with ...
CNT_SET(0)               ... 20 MHz or
CNT_SET(1)               ... 5 MHz reference frequency
CNT_INPUTMODE(1)         input CLR/LATCH as LATCH input
CNT_SE_DIFF(0)           select single-ended inputs
CNT_ENABLE(1)            enable counter(s)

EVENT:                   event loop ...
CNT_READLATCH(1)         read out latch A
. . .                    evaluate counter value in process
```

8.4.2 Impulse Width and Pause Duration Measurements

All 4 counters measure impulse width and pause duration.

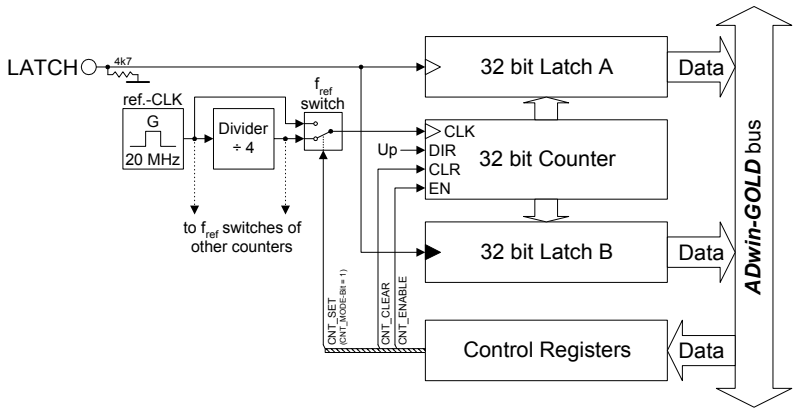


Fig. 24 – Block diagram of the *CO1* add-on mode
"impulse width/pause duration"

The counters 1 and 2 have two latches for positive (Latch A) and negative edges (Latch B). Thus, pulse and pause duration can be evaluated separately by calculating the differences of the latches.

Programming example

```
INIT:                                  initialization ...
. . .
CNT_ENABLE(0)                          disable counters
CNT_CLEAR(1)                           clear counter(s)
CNT_MODE(1)                            mode "internal reference clock" with ...
CNT_SET(0)                             ... 20 MHz or
CNT_SET(1)                             ... 5 MHz reference frequency
CNT_INPUTMODE(1)                       input CLR/LATCH as LATCH input
CNT_SE_DIFF(0)                         select single-ended inputs
CNT_ENABLE(1)                          enable counter(s)

EVENT:                                 event loop ...
CNT_READLATCH(1)                       read out latch A
CNT_READFLATCH(1)                      read out latch B
. . .                                  evaluate counter value in process
```

8.4.3 Hardware addresses (*CO1*-add-on)

A process can be executed very quickly if you access directly the control and data register (see chapter 5.3 and ***ADbasic*** manual). The hardware addresses of the CO1 add-on can be found in the following table (see command index table in chapter 8.2).

Address [HEX]	Function	Bit					Commentary
		31:04	3	2	1	0	
20400204	Read out Latch A: Counter 1	x	x	x	x	x	x : Contents of the latch
20400208	Read out Latch B: Counter 1	x	x	x	x	x	
20400214	Read out Latch A: Counter 2	x	x	x	x	x	
20400218	Read out Latch B: Counter 2	x	x	x	x	x	
20400224	Read out Latch A: Counter 3	x	x	x	x	x	
20400238	Read out Latch B: Counter 3	x	x	x	x	x	
20400234	Read out Latch A: Counter 4	x	x	x	x	x	
20400238	Read out Latch B: Counter 4	x	x	x	x	x	
20400300	Enable counter	-	x	x	x	x	x = 0 : Disable counter x = 1 : Enable counter
20400304	Set counter inputs to TTL or differential mode (in pairs only)	-	-	-	y	x	x: counter inputs 1+2 y: counter inputs 3+4 x,y = 0: TTL (single-ended) x,y = 1: differential
20400310	Clear counter	-	x	x	x	x	x = 0 : No influence x = 1 : Clear counter
20400320	Latch counter	-	x	x	x	x	x = 0 : No influence x = 1 : Latch counter
20400330	Input: CLR or LATCH	-	x	x	x	x	x = 0 : CLR input x = 1 : LATCH input
20400340	Impulse/event counter or impulse/pause duration measurement	-	x	x	x	x	x = 0 : External clock input x = 1 : Int. ref. clock(20/5MHz)
20400350	4 edge evaluation/CLK+DIR or 20/5MHz reference 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
20400370	Counter: Error register[a]	several bits					Error bits, see CNT_GETSTATUS

a. You have to reset this register manually!

Fig. 25 – Hardware addresses of the *CO1* counter add-on

9 CAN add-on

The add-on *Gold-CAN* is equipped with several additional interfaces that are configured and operated individually:

– 4 SSI decoders (page 33)

  The decoders can be used for the connection of incremental encoders with SSI interface. All inputs are differential and designed for RS422/485 level (5V).

  The decoder inputs are located on the connectors CO1...CO4, where the inputs of the CO1 add-on can also be found.

– 2 CAN interfaces (page 35)

  Depending on your requirements, you can order both interfaces either as high-speed or low-speed version. Switching in operation is not possible.

  The inputs of the CAN 1 interface are located on the connectors CAN 1.1 and CAN 1.2, those of the CAN 2 interface on the connector CAN 2.

– 2 RSxxx interfaces (page 39)

  Both interfaces can be configured separately per software to be operated as RS232 or RS485.

  The interface inputs are located on the connectors COM1 and COM2.

The add-on *Gold-CAN* is only available in combination with the Gold-D option.

9.1 SSI Decoder

An incremental encoder with SSI interface can be connected to the decoders. The signals are differential and have RS422/485 levels.

The decoders either read out an individual value (on request) or they continously provide the current value.

The connections of the 4 decoders are on the connectors CO1...CO4 (15-pin, DSUB), on the pins 5, 8, 14 and 15 (see fig. 26). If the device is equipped with the CO1 add-on the remaining pins are reserved for the counter connections.

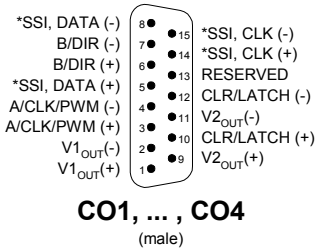
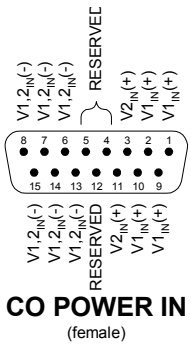


Fig. 26 – Pin assignment SSI decoder

A voltage input to the connection `CO Power in` is supplied at the connectors CO1...CO4, for example for an external incremental encoder.
Please note: The negative inputs $U1_{in}$ (-) are galvanically connected with GND via a common line, the negative inputs $U2_{in}$ (-) have such a connection, too.



Setting the properties

The following properties of the decoders can be set via software:

– Clock rates: With **`SSI_SET_CLOCK`** clock rates of approx. 40 kHz up to 1 MHz are possible with a pre-scaler.

– Resolution: Can be set with **`SSI_SET_BITS`** up to 32 bit.

Example: Conversion of Gray code

A conversion from Gray code into binary code is made with the routine below, which you have programmed in the ***ADbasic*** process.

```
REM PAR_1 = Gray value to be converted
REM PAR_2 = Flag indicating a new Gray value
REM PAR_9 = Result of the Gray-to-binary conversion

DIM m, n AS LONG

EVENT:
 IF(PAR_2=1) THEN            'Start of conversion
  m=0                        'initialize value
  PAR_9=0                    ' -"-
  FOR n=1 TO 32              'Go through all possible 32 bits
   m=(SHIFT_RIGHT(PAR_1,(32-n)) AND 1) XOR m
   PAR_9=(SHIFT_LEFT(m,(32-n))) OR PAR_9
  NEXT n
 PAR_2=0                     'Enable next conversion
 ENDIF
```

Fig. 27 – Listing: Conversion of Gray code into binary code

Programming

The functionality of the decoders is easily programmed with ***ADbasic*** instructions:

Range	Instructions
Initialization	SSI_MODE SSI_SET_BITS SSI_SET_CLOCK
Receiving of data	SSI_READ SSI_START SSI_STATUS

The instructions are in the include file <ADWGCAN.INC>. More information can be found in the ***ADbasic*** manual and the online help.

9.2 CAN Interface

The CAN interfaces 1 and 2 can be operated individually. Depending on your requirements, you can order both interfaces either as high-speed or low-speed version. Switching in operation is not possible.

9.2.1 Hardware Description

The connections of the interfaces 1 and 2 are located on the 9-pin DSUB connector:

– Interface 1: Connector (male) CAN 1.1 and connector (female) CAN 1.2. The pins of the connectors are internally connected with each other.

– Interface 2: Connector CAN 2.

The pinouts for CAN "High speed" and "Low speed" are different.

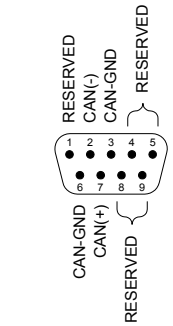
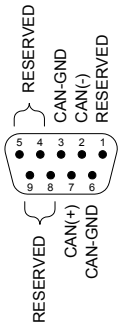
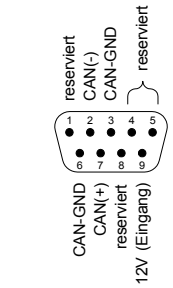
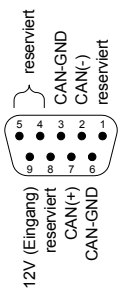
CAN Bus	Male connectors CAN 1.1, CAN 2	Female connector CAN 1.2
High speed	 **CAN 1.1 & CAN 2** (male)	 **CAN 1.2** (female)
Low speed		

Abb. 28 – CAN: Pinbelegungen

Both interfaces have their individual CAN-GND potential; the potentials are both galvanically isolated from each other as well as from the mass potential (GND) of the enclosure.

Power supply (Low speed only)

The low speed version requires an external power supply of 12V DC to run the CAN controller. The power must be supplied for each interface separately.

Bus Termination (High speed only)

If the CAN interface functions as the physical termination of a high-speed CAN bus, it must be terminated with a 120Ω resistor (only the first or the last CAN node). CAN nodes, which are not positioned in an end-location, must not be terminated.

If termination is required for one (or both) interfaces, the pins `CAN(+)` and `CAN(-)` must be connected by a resistor of 120Ω.

9.2.2 Description of the CAN interface

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. 29 – 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 interface is easily programmed using ***ADbasic*** instructions:

Range	Instructions
Initialization	INIT_CAN EN_INTERRUPT SET_CAN_BAUDRATE

Range	Instructions
Receiving and sending of data	CAN_MSG EN_RECEIVE EN_TRANSMIT READ_MSG TRANSMIT
Write / read access to the controller register	SET_CAN_REG GET_CAN_REG

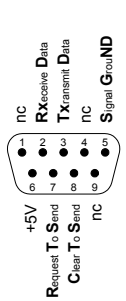
The instructions are in the include file <ADWGCAN.INC>. More information can also be found in the ***ADbasic*** manual and the online help.

9.3 RSxxx Interfaces

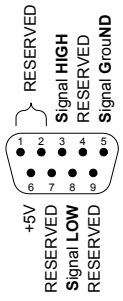
Each of the 2 RSxxx interfaces is equipped with the "Quad Universal Asynchronous Receiver/Transmitter" controller (UART), type TL16C754 from Texas Instruments®. Functionality and programming of the interfaces are based on this controller.

Both interfaces can be operated individually with the RS232 or RS485 protocol. The physical difference between the protocols is the level of the signals, which are generated by special driver components on the bus.

Pin assignment



COM1, COM2
(RS232) *(male)*



COM1, COM2
(RS485) *(male)*

Bus termination (RS485 only)

If an RS485 interface functions as the physical bus termination, the terminator must be a resistor (only the first or last RS485 participant). RS485 participants, which are not positioned in an end-location, must not be terminated.

For the termination there is – if required for the chosen circuit type – a voltage of +5V provided at pin 6.

9.3.1 Setting the interface parameters

Each interface has an input and an output FIFO with a length of 64 bytes each.

The settings of the interface parameters are made separately for each channel, using the controller register. Below the settings are described more detailed:

Handshake

– Handshake: The interface is operated in 4 modes:
  1. RS232 without handshake
  2. RS232 with software handshake (Xon/Xoff)
  3. RS232 hardware handshake (RTS/CTS). The signals RTS and CTS must be connected.
  4. RS485

Parity

– Parity: In order to recognize an error or incorrect data during the transfer, a parity bit can be transferred at the same time. The parity can be even or odd or you can have no parity bit at all.

Data bits

– Data bits: the active data to be transferred may be 5...8 bits long.

Stop bits

– Stop bits: The number of stop bits can be set to 1, 1½ or 2. Here the number of stop bits depends on the number of data bits:
  • 5 data bits: 1 or 1½ stop bits.
  • 6…8 data bits: 1 or 2 stop bits.

Baud rate

– Baud rate: The physical data are between 35 Baud and 2.304 MBaud; when using an RS-232 interface the maximum Baud rate is 115.2 kBaud.

  The Baud Rates are derived by the clock rate of the module; the basic clock rate has a frequency of 2.304 MHz. Based on this fact, every Baud rate is possible that can be derived by an integer division of the basic frequency. The divisor can have values between `1...0FFFFh`. The following table shows some common Baud rates and their divisors:

Baud rate	Divisor	
	dez.	hex.
2304000	1	0001h
1152000	2	0002h
460800	5	0005h
230400	10	000Ah
115200	20	0014h
57600	40	0028h
38400	60	003Ch

Baud rate	Divisor	
	dez.	hex.
19200	120	0078h
9600	240	00F0h
4800	480	01E0h
2400	960	03C0h
1200	1920	0780h
600	3840	0F00h
300	7680	1E00h

Fig. 30 – RS-xxx: Baud rates

Special features of RS485

Via a RS485 interface more than 2 participants can communicate with each other. (Contrary to the RS232 interface). With RS485 interfaces a bus can be set up.

Consider the following:

– There is no handshake, because a handshake is only possible between 2 participants.

– The interface must know if it should write to the bus or get data from the bus (**RS485_SEND**).

9.3.2 Programming

Functionality and programming of the interface depend on this controller. The controller is easily programmed with ***ADbasic*** instructions:

Range	Instructions
Initialization	**RS_INIT**, **RS_RESET**
Receiving and transmitting of data	**RS485_SEND**, **READ_FIFO**, **WRITE_FIFO**
Write and read access to the controller register	**GET_RS**, **SET_RS**

The instructions are in the include file <ADWGCAN.INC>. More information can also be found in the ***ADbasic*** manual and the online help.

Example programs

RS232

The following program illustrates the initialization of the serial RS232 interface in the **INIT:** section and the cyclic reading and writing of data in the **EVENT:**. section. The process is timer-controlled:

```
REM The program initializes the serial interface
REM in the Init: section.
REM In the Event: section data is exchanged between
REM the interfaces 1 & 2 of the RS module.
REM The interfaces are tested with this program.
REM For this connect the interfaces with each other
REM befor starting the program.

#INCLUDE adwpext.inc
DIM DATA_1[1000] AS LONG'Transmitted data
DIM DATA_2[1000] AS LONG'Received data
DIM lauf AS LONG        'Control variable

INIT:
 FOR run = 1 TO 1000    'Initialization of the transmit-
                        'ted data
  DATA_1[run] = run AND 0FFh
 NEXT run
 REM Initialization of the interfaces:
 REM 9600 Baud, not parity bit, 8 data bits,
 REM 2 stop bits, RS232 witout handshake
 RS_INIT(19600.0.8,1,0)
 RS_INIT(2,9600,0,8,1,0)
 PAR_1 = 1
 PAR_4 = 1

EVENT:
 REM Read and write a data set
 IF (PAR_1 <= 1000) THEN'Send data
  PAR_2 = WRITE_FIFO(1,DATA_1[PAR_1])
  IF (PAR_2 = 0) THEN INC PAR_1
 ENDIF

 PAR_3 = READ_FIFO(2)   'Read data
 IF (PAR_3 <> -1) THEN
  DATA_2[PAR_4] = PAR_3
  INC PAR_4
 ENDIF
 IF (PAR_4 > 1000) THEN END'All data are transmitted
```

RS485

In this example the RS485 interface is a passive participant, which reads data coming from the input. If a specified value (55) is received, the interface starts to send. It sends continuously the value 44.

```
REM Interface 2 reads all data coming from the bus
REM until it receives the value 55. Now the interface
REM becomes active and sends the value 44.

#INCLUDE adwgcan.inc
DIM ret_val, val AS LONG

INIT:
 RS_RESET()
 REM Initialization of the interfaces:
 REM 38400 Baud, no parity bit, 8 data bits,
 REM 1 stop bit, RS485 software handshake
 RS_INIT(1,38400,0,8,0,3)
 RS_INIT(2,38400,0,8,0,3)
 RS485_SEND(1,1)        'Send interface 1
 RS485_SEND(2,0)        'Receive interface 2

EVENT:
  val = READ_FIFO(2)    'Read data from interface 2

  IF (val = 55) THEN
   RS485_SEND(2,1)      'Send interface 2
   ret_val = WRITE_FIFO(2,44) 'Write data
  ENDIF
```

10 *ADwin-Gold*-Boot

This option is only available in an ***ADwin-Gold-ENET***.

ADwin-Gold-Boot starts a previously programmed application automatically after power-up. After installation of this application an operation without computer is possible.

With ***ADwin-Gold-Boot*** the following steps are executed after power-up:

– Loading the operating system

– Loading of the compiled processes, compiled by ***ADbasic*** (max. 10).

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

Disable boot loader

If you do not wish to work with the boot loader option:

– Boot the system after power-up and the previously saved processes are disabled.

– After switching off and powering up anew, the boot loader option is enabled again.

By programming the Flash-EEPROM without processes and only with the file <ADwin9.btl> the system will only be booted after power-up, but no processes can be executed.

With the installation of the ***ADwin*** Developer-Software from the supplied ***ADwin*** CD-ROM, the utility program for the boot loader (ADethfl[1]ash) is automatically copied. You should have a CDROM version 3.00.2735 or a later version.

Use the program <ADethflash.exe> for an ***ADwin-Gold*** system with Ethernet interface.

At standard installation you will find the program in the directory

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

Help for Ethernet interface

You will find information about the boot loader with Ethernet interface in the ***ADwin*** Driver Installation manual.

2000 values you can freely dispose of

In combination with the Ethernet interface and boot loader you can write or read out 2000 long or float values à 32-bit via ***ADbasic*** processes into/from the Flash- EEPROM. A more detailed description can be found in the program <ADethflash.exe> by clicking on "Info about eeprom support".

1. After execution this register will automatically be reset.

11 Accessories

The following accessories are available for the ***ADwin-Gold***:

– ***ADwin-Gold***-pow: external 12V power supply unit

– various lengths of power supply and USB or Ethernet cable

– cable connector for an external power supply

– mounting kits for enclosures

***ADwin-Gold*-pow**

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

Please pay attention to the fact that the USB, Ethernet cables are sufficiently shielded, in order to avoid interferences in the data lines. Interferences have to be passed before entering the chassis via GND (ground). (See also chapter 3 "Operating Environment").

12 Software

You are programming the ***ADwin-Gold*** system - all add-ons included - with simple ***ADbasic*** instructions. The instructions are described in the ***ADbasic*** manual.

The manual's annex contains a list of all instructions, which can be used for ***ADwin-Gold*** and its add-ons.

Annex

A.1 Technical Data

All technical data refer to a powered-up ***ADwin-Gold*** system.

General Data/Limit Values						
	Symbol	Conditions	min.	typ.	max.	Unit
Supply Voltage/Supply Current						
Voltage	U_b		10	12	35	V
Idle current, USB Interface	$I_{idle, USB}$	U_b=10V		1.1		A
		U_b=12V		0.9		
		U_b=35V		0.3		
		U_b=12V; *Gold-DA*		1.4		
		U_b=12V; *Gold-CO1*		0.9		
Power-up current, USB Interface	$I_{power-on, USB}$	U_b=12V	1.7			
		U_b=12V; *Gold-DA*	2.9			
		U_b=12V; *Gold-CO1*	1.7			
Idle current, Ethernet Interface	$I_{idle, USB}$	U_b=10V		1.3		A
		U_b=12V		1.1		
		U_b=35V		0.4		
		U_b=12V; *Gold-DA*		1.5		
		U_b=12V; *Gold-CO1*		1.1		
Power-up current, Ethernet Interface	$I_{power-on, Enet}$	U_b=12V	2.1			
		U_b=12V; *Gold-DA*	3.1			
		U_b=12V; *Gold-CO1*	2.1			
Valid operation ranges						
Temperature	$T_{chassis}$		+5		+60	°C
Relative humidity	F_{rel}	no condensation	0		80	%
Storage						
Temperature	T		-20		+70	°C
Connectors						
DSUB connectors	Metric ISO threads; UNC threads available as ordering option					
Dimensions						
Width x height x depth	W x H x D	*Gold-USB*, *Gold-ENET*	214 × 67 × 109			mm
		+ *CO1* Add-On	Height: +30			
Net weight						
Weight	m_{Net}	*Gold-USB*, *Gold-ENET*	1,320			g
		with *CO1* Add-On	1,760			
		Clips[a]	32			

a. Accessories for DIN rail mounting: *Gold-Mount*

Digital Inputs/Outputs						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
I/O-lines						
Number	DIO00:DIO31	32 (programmable in groups of 8 as inputs or outputs)				
	EVENT	ext. trigger input (positive TTL logic)				
Inputs						
Max. input voltage		V_{CC} = 5V	-0.5		+5.5	V
Logic input voltage	V_{IH} (High)	V_{CC} = 5V	2.4			
	V_{IL} (Low)	V_{CC} = 5V			0.8	
Logic input current	I_I	V_{CC} = 5V		±0.01	±2	µA
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	
EVENT Input						
Edge recognition, pos.	V_{T+} (Low)	V_{CC} = 5V	1.65	1.9	2.15	V
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	2 × 8 via multiplexer, differential					
Input resistance	R_i		323.4	330	336.6	kΩ
Overvoltage	$U_{in\ max.}$	ON & OFF			±35	V
Multiplexer settling time	t_{MUX}	1 LSB 14-bit		2.5		µs
		1 LSB 16-bit		6.5		µs
ADC 14-bit						
Conversion time	t_{conv}				0.5	µs
Measurement range	U_{in}	F_V=1	-10		+9.999695	V
		F_V=2	-5		+4.999847	
		F_V=4	-2.5		+2.499924	
		F_V=8	-1.25		+1.249962	
Diff. common mode voltage.					±2.5	
Integral non-linearity	INL			±1	±3	LSB
Differential non-linearity	DNL			±0.25	±0.5	

Analog Inputs/Outputs						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
Offset	Drift			±2		ppm/K
	Error	adjustable				
Gain	Drift			±5		ppm/K
	Error	adjustable				
ADC 16-bit						
Conversion time	t_{conv}				5	µs
Measurement range	U_{in}	F_V=1	-10		+9.999695	V
		F_V=2	-5		+4.999847	
		F_V=4	-2.5		+2.499924	
		F_V=8	-1.25		+1.249962	
Diff. common mode voltage					±2.5	
Integral non-linearity	INL			±1	±3	LSB
Differential non-linearity	DNL			±0.25	±0.5	
Offset	Drift			±2		ppm/K
	Error	Adjustable				
Gain	Drift			±5		ppm/K
	Error	Adjustable				
Outputs: DAC 16-bit						
Number	2 (with DA add-on: 8)					
Output voltage	U_{out}		-10		+9.999695	V
Settling time	t_{settle}	2V jump		3		µs
		FSR [a] (20V)		10		
Permissible current					±25	mA
Integral non-linearity	INL				±2	LSB
Differential non-linearity	DNL				±1	
Offset	Drift			±1		ppm/K
	Error	Adjustable				
Gain	Drift			±3		ppm/K
	Error	Adjustable				

a. 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 programs		128	256 [a]	kByte
		for data		128	256 [a]	
External memory	SDRAM			16	64 [a]	MByte

a. combined memory expansion G-MEM-64

CO1 Add-On						
Parameters	Symbol	Conditions	min.	typ.	max.	Unit
Counter						
Number	4 counters (CNTR1 … CNTR4)					
Inputs	For each counter 3 differential inputs (A/CLK, B/DIR, CLR/LATCH); counter inputs programmable in pairs for differential or TTL mode (single-ended)					
Counter resolution				32		Bit
Count frequency	f_{CLK}	Input CLK		20		MHz
		Input A/B		5		
Latch width	LATCH			32		Bit
Reference quartz oscillator						
Reference frequency	f_{ref}			20		MHz
Prescaler by 4	f_{ref} / 4			5		
Accuracy and Drift					100	ppm
Counter inputs differential[a]						
Differential input threshold voltage	V_{TH}	-10V ≤ V_{CM} ≤ 13.2V	-200		+200	mV
Input hysteresis	ΔV_{TH}	-10V ≤ V_{CM} ≤ 13.2V		40		mV
Range of common mode voltage	V_{CM}		-10		+13.2	V
Differential slew rate			0.33			V/µs
Permissible differential input voltage		for each input			±3.9	V
Counter inputs single ended[b] (with Schmitt trigger)						
Edge recognition, pos.	V_{T+} (Low)	V_{CC} = 5V	1,65	1,9	2,15	V
Edge recognition, neg.	V_{T-} (Low)		0,75	1,0	1,25	
Switching hysteresis	V_{T+} - V_{T-}		0,4	0,9		
Input current	I_H	V_I = 2,7V			20	µA
	I_L	V_I = 0,4V			-50	

a. see also data sheet MAX3098 from MAXIM
b. see also data sheet 74LS19 from Texas Instruments

A.2 Hardware Addresses - General Overview

Address [HEX]	Function	Bit No..												Comment	Register available in the module		
		31:16	15:10	9	8	7	6	5	4	3	2	1	0		Gold	Gold-DA	Gold-CO1
20400000	Set MUX 1: IN1, IN3, …, IN15	-	-	-	-	-	-	-	-	-	n	n	n	"nnn"binary = 0…7 decimal, selected channel = nnn + 1	x	x	x
	Set MUX 2: IN2, IN4, …, IN16	-	-	-	-	-	-	n	n	n	-	-	-	"nnn" binary = 0…7 decimal, selected channel = 2×(nnn + 1)	x	x	x
	Gain PGA 1	-	-	-	-	g	g	-	-	-	-	-	-	"gg" binary = 0…3 decimal, selected channel = 2^{gg}	x	x	x
	Gain PGA 2	-	-	g	g	-	-	-	-	-	-	-	-		x	x	x
20400010	Start conversion: ADC 1, 16-bit	-	-	-	-	-	-	-	1	1	1	1	s	s = 0 : start conversion s = 1 : no effect	x	x	x
	Start conversion: ADC 2, 16-bit	-	-	-	-	-	-	-	1	1	1	s	1				
	Start conversion: all DACs synchronously	-	-	-	-	-	-	-	1	1	s	1	1				
	Start conversion: ADC 1, 14-bit	-	-	-	-	-	-	-	1	s	1	1	1				
	Start conversion: ADC 2, 14-bit	-	-	-	-	-	-	-	s	1	1	1	1				
20400020	Start conversion: ADC 1, 16-bit	-	-	-	-	-	-	-	-	-	-	-	e	e = 0 : end of conversion e = 1 : conversion is running	x	x	x
	Start conversion: ADC 2, 16-bit	-	-	-	-	-	-	-	-	-	-	e	-				
	Start conversion: ADC 1, 14-bit	-	-	-	-	-	-	-	-	-	e	-	-				
	Start conversion:: ADC 2, 14-bit	-	-	-	-	-	-	-	-	e	-	-	-				
20400030	Read out register: ADC 1, 16-bit	-	x	x	x	x	x	x	x	x	x	x	x	x : result of the conversion	x	x	x
20400040	Read out register: ADC 2, 16-bit	-	x	x	x	x	x	x	x	x	x	x	x		x	x	x
20400050	Write only into register: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20400060	Write only into register: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x		x	x	x
20400070	Write only into register: DAC 3	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
20400080	Write only into register: DAC 4	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
20400090	Write only into register: DAC 5	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
204000A0	Write only into register: DAC 6	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
204000B0	Input regiser DIO00:DIO15	-	x	x	x	x	x	x	x	x	x	x	x	x : read digital value	x	x	x
204000C0	Output registerDIO16:DIO31	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be output	x	x	x
20400100	Read out register and start conversion: ADC 1, 16-bit	-	x	x	x	x	x	x	x	x	x	x	x	x : result of the conversion	x	x	x
20400110	Read out register and start conversion: ADC 2, 16-bit	-	x	x	x	x	x	x	x	x	x	x	x		x	x	x
20400120	Read out register and start conversion: ADC 1, 14-bit	-	x	x	x	x	x	x	x	x	x	0	0		x	x	x
20400130	Read out register: ADC 1, 14-bit	-	x	x	x	x	x	x	x	x	x	0	0		x	x	x
20400140	Read out register: ADC 2, 14-bit	-	x	x	x	x	x	x	x	x	x	0	0		x	x	x
20400190	Write only into register: DAC 7	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	–	x	–
204001A0	Write only into register: DAC 8	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
204001B0	Input register DIO16:DIO31	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be read in	x	x	x
204001C0	Output register DIO00:DIO15	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be output	x	x	x
204001D0	Read out register and start conversion: ADC 2, 14-bit	-	x	x	x	x	x	x	x	x	x	0	0	x : digital value to be converted	x	x	x
204001E0	Configure DIO00:DIO07	0	0	0	0	0	0	0	0	-	-	-	c	c = 0 : inputs c = 1 : outputs	x	x	x
	Configure DIO08:DIO15	0	0	0	0	0	0	0	0	-	-	c	-		x	x	x
	Configure DIO16:DIO23	0	0	0	0	0	0	0	0	-	c	-	-		x	x	x
	Configure DIO24:DIO31	0	0	0	0	0	0	0	0	c	-	-	-		x	x	x
20400200	Write into register and start conversion immediately: DAC 1	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20400204	Contents of Latch A, counter 1	x	x	x	x	x	x	x	x	x	x	x	x	x :contents of the latch	x	x	x
20400208	Contents of Latch B, counter 1	x	x	x	x	x	x	x	x	x	x	x	x		–	–	x
20400210	Write into register and start conversion immediately: DAC 2	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	x	x	x
20400214	Contents of Latch A, counter 2	x	x	x	x	x	x	x	x	x	x	x	x	x :contents of the latch	x	x	x
20400218	Contents of Latch B, counter 2	x	x	x	x	x	x	x	x	x	x	x	x		–	–	x
20400220	Write into register and start conversion immediately DAC 3	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	–	x	–
20400224	Contents of Latch A, counter 3	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch	–	–	x
20400228	Contents of Latch B, counter 3	x	x	x	x	x	x	x	x	x	x	x	x		–	–	x
20400230	Write into register and start conversion immediately DAC 4	-	x	x	x	x	x	x	x	x	x	x	x	x :digital value to be converted	–	x	–
20400234	Contents of Latch A, counter 4	x	x	x	x	x	x	x	x	x	x	x	x	x : contents of the latch	–	–	x
20400238	Contents of Latch B, counter 4	x	x	x	x	x	x	x	x	x	x	x	x		–	–	x

Address [HEX]	Function	Bit No..												Comment	Register available in the module		
		31:16	15:10	9	8	7	6	5	4	3	2	1	0		Gold	Gold-DA	Gold-CO1
20400240	Write into register and start conversion immediately: DAC 5	-	x	x	x	x	x	x	x	x	x	x	x	x : digital value to be converted	–	x	–
20400250	Write into register and start conversion immediately: DAC 6	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
20400260	Write into register and start conversion immediately: DAC 7	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
20400270	Write into register and start conversion immediately: DAC 8	-	x	x	x	x	x	x	x	x	x	x	x		–	x	–
20400300	Enable/disable counter: CNT_ENABLE()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : disable counter x = 1 : enable counter	x	x	x
20400310	Delete counter: CNT_CLEAR()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : no effect x = 1 : clear counter	x	x	x
20400320	Latch counter: CNT_LATCH()	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : no effect x = 1 : latch counter	x	x	x
20400330	Select counter input: CLR or LATCH	-	-	-	-	-	-	-	-	x	x	x	x	x = 0 : CLR input x = 1 : LATCH input	–	–	x
20400340	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)	–	–	x
20400350	4-edge evaluation/ CLK+DIR or 20MHz / 5MHz reference clock	-	-	0	0	-	-	-	-	x	x	x	x	CNT_MODE=0: x=0: 4-edge evaluation; x=1: CLK+DIR CNT_MODE=1: x=0: 20MHz; x=1: 5MHz	–	–	x
20400370	Counter: Error register[a]	several bits												Error bits (CNT_GETSTATUS)	–	–	x

a. After execution this register will automatically be reset.

A.3 Hardware revisions

The revision of a Gold system is marked on the bottom of the casing. The differences of the revision status' are shown below.

Revision	First release	Changes to previous revision status
A	1998	First release with link data connection.
B1	Nov. 2002	Protoype (internal use only, not delivered to customers)
B2	Apr. 2003	Data connection to PC no longer via link, but via Ethernet or USB. All analog inputs and counter inputs are only available with differential operation mode.
B3	Nov. 2003	Additional TTL counter inputs for single-ended operation mode (for use as alternative to counter inputs for differential operation mode). New option Gold-D with DSUB connectors instead of BNC sockets.
B4	Dec. 2003	Several enhancements
B5	Mar. 2004	Several enhancements Layout change of printed circuit board
B6	Aug. 2004	Enhanced Ethernet interface (ENET-2) with increased data throughput. New option Gold-CAN with several communication interfaces.

A.4 Table of figures

A.5 Index