

ADwin-Pro II

System and hardware description



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

Hotline: +49 6251 96320
Fax: +49 6251 568 19
E-Mail: info@ADwin.de
Internet: www.ADwin.de

 **JÄGER**
Computergesteuerte
Messtechnik GmbH
Jäger Com-
putergesteuerte
Messtechnik GmbH
Rheinstraße 2-4
D-64653 Lorsch
Germany

Table of contents

Table of contents	III
Typographical Conventions	IV
1 The ADwin-Pro II system	1
2 How to Install an ADwin-Pro II System	2
3 Operating Environment	3
4 Enclosures for the ADwin-Pro II System	4
4.1 ADwin-Pro II	5
4.2 ADwin-Pro II-DC	6
4.3 ADwin-Pro II-BM	7
4.4 ADwin-Pro II-light	8
4.5 ADwin-Pro II-light-DC	9
4.6 ADwin-Pro II-mini	10
5 ADwin-Pro Modules	11
5.1 Unsupported Pro I modules	11
5.2 Setting the module's addresses (Pro I modules, too)	11
5.3 Processor modules	13
5.4 Pro II: Multi-IO Modules	23
5.5 Pro II: Analog Input Modules	52
5.6 Pro II: Analog Output Modules	92
5.7 Pro II: Digital-I/O Modules	104
5.8 Pro II: Extension and Interface Modules	146
6 Calibration	223
6.1 General information	223
6.2 Calculation basis	224
6.3 Calibrating a module	225
7 Accessories	227
7.1 LEMO Cable Sets for ADwin-Pro Systems	227
7.2 LEMO Adapter sets	227
7.3 Cables / Terminal blocks for OPT-16 and TRA-16	228
7.4 Enclosure Mounting	228
7.5 Reference addresses	228
Annex	A-1
A.1 RoHS Declaration of Conformity	A-1
A.2 List of Modules	A-1

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 absolutely have to be considered in order to guarantee an error free operation.
- 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 commands and user inputs are characterized by the font *Courier New*.

Var_1

Source code elements such as commands, variables, comments and other text are characterized by the font *Courier New* and are printed in color.

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 The ADwin-Pro II system

The *ADwin-Pro II* system is an external processing system with modular expansion options. Depending on applications, the different enclosures can be equipped with classic *ADwin-Pro I* and new *ADwin-Pro II* modules.

When the *ADwin-Pro II* system was developed great attention was paid to the electromagnetic compatibility. The *ADwin-Pro II* system and all available input and output modules have the CE sign and can therefore be configured differently later if necessary.

Each *ADwin-Pro II* system needs a processor module. It communicates via Ethernet with the PC or notebook.

In order to meet the various requirements for measurement and control tasks, the system can be equipped with the following modules:

- analog input modules and analog output modules
- digital input modules and digital output modules
- counters
- filters, isolation amplifiers
- amplifiers for thermocouples and resistance thermometers
- serial communication interfaces (CAN, RSxxx, Fieldbus)
- storage / read module for PCMCIA storage media

All modules have a revision identifier written on the module front, e.g. Rev. A2, Rev. B3, Rev. C3. Earlier delivered modules have no identifier; they are to be considered as revision "Rev. A". *ADwin-Pro II* modules have the revision identifier Rev. E1 or higher.

Different revision characters mean different module properties and are described separately.

The revision identifier is followed by a minor counting number, which is mainly used for internal purposes of Jaeger Computergesteuerte Messtechnik GmbH.

Applicable modules

Revision Identifier

2 How to Install an ADwin-Pro II System

Please keep strictly to the following order:

1. Start with the manual "ADwin installation":
 - Install software and interface drivers from the ADwin-CDROM.
 - Initialize the data connection from PC to ADwin system and do an operational test.
The power connectors are described in [chapter 4 "Enclosures for the ADwin-Pro II System"](#).
 - Follow the notes in [chapter 3 "Operating Environment"](#).
2. Set module addresses, see [chapter 5.2 on page 11](#).
3. Do your first steps with the ADbasic Tutorial.
4. Programming in ADbasic:

The ADbasic manual describes the real-time development environment, the structure of an ADbasic program and gives hints for optimizations.

The ADbasic instructions are described in the online help of the development environment or in these documents:

- ADbasic manual: Basic instructions for calculation, program structure and process control.
- ADwin-Pro II software manual: Instructions and hints for accessing the Pro modules.

For operation, please pay attention to the notes in this manual concerning the respective modules.

Please note:

For ADwin systems to function correctly, follow strictly the information provided in this documentation and in other mentioned manuals.

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 activities, while recognizing and avoiding any possible dangers.

(Definition of qualified personnel as per VDE 105 and ICE 364).

This product documentation and all documents referred to, have always to be available and to be strictly 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.

This documentation, including all pictures is protected by copyright. Reproduction, translation as well as electrical and photographic archiving and modification require a written permission by the company *Jäger Computergesteuerte Messtechnik GmbH*, Lorsch, Germany.

OEM products are mentioned without referring to possible patent rights, the existence of which may not be excluded.

Hotline address: see inner side of cover page.

Qualified personnel

Availability of the documents



Legal information

Subject to change.

3 Operating Environment

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

- build a ground reference point for the electronic
- conduct interferences to earth.

Connect the GND clamp / plug via a short low-impedance solid-type cable to the central earth connection point of the controlled system. The GND plug is internally connected with ground and the enclosure.

In the Ethernet cable, the data lines are galvanically isolated, but the ground potentials are connected, because the shielding of the Ethernet connector (RJ-45) is connected to GND.

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 controlled system 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 enclosure by using screen clips.

Operate the device with the defined and fitting supply voltage. For operation with an external power supply, the instructions of the manufacturer apply. Close the device for operation, use cover plates to cover gaps between built-in modules.

ADwin-Pro II is designed for operation in dry rooms with an ambient temperature of +5°C ... +50°C and a relative humidity of 0 ... 80% (no condensation). The device may be operated in a control cabinet or mobile (e.g. in a car).

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

For use in a control cabinet, please note:

- The device shall not be placed above strong heat sources, e.g. a high power transformer.
- Ventilation inside the control cabinet towards and from the *ADwin-Pro II* device must be provided. Especially, the ventilation slots of the device must be kept free, so that the device can lead off its generated heat completely.
- If the device is mounted at the front side with mounting angles Pro II-Mount (see [chapter 7.4](#)), i.e. in a 19" rack, we recommend to additionally support the enclosure at the rear side on a bearing for safety reasons.

Earth protection



Galvanic connection

Excluding transient currents



Supply voltage

Ambient atmosphere

Chassis temperature



4 Enclosures for the ADwin-Pro II System

The different sizes for the enclosures depend on the number of slots and the kind of power supply.

Enclosure	Number of Slots	Power supply	
		Power supply	
ADwin-Pro II	16	100V...240V	AC
ADwin-Pro II-DC	16	10V...35V	DC
ADwin-Pro II-BM	15	100V...240V	AC
ADwin-Pro II-light	7	100V...240V	AC
ADwin-Pro II-light-DC	7	10V...35V	DC
ADwin-Pro II-mini	5	10V...36V	DC

The number of slots is given for Pro II modules. If Pro I modules be used—in combination with Pro II modules or not—less modules fit into the enclosure.

For the slot area (including power supply slot) the following dimensions apply:

$$1 \text{ HP} = 1/5 \text{ inch} = 5.08 \text{ mm}$$

$$1 \text{ U} = 1\frac{3}{4} \text{ inch} = 44.45 \text{ mm}$$

The slots mostly have a width of 5 HP = 1 inch.

Plug-in a module



You plug-in a module into the enclosure like this:

- Switch off the *ADwin* device! A module may sustain damage if you plug it in or out with the power supply switched on.
- Remove one or more cover plates at the wanted position, until the bearings be seen at the left edge: one upper and one lower bearing.
 - Pay attention to the color of the bearings. There are different, offset bearings for Pro I and Pro II modules:
 White bearings: Pro I modules.
 Black bearings: Pro II modules.
 - The processor module has a fixed position, no other position can be used.
- Insert the board carefully into both bearings, plug ahead. If positioned correctly the module cannot be skewed.
- Push the module into the enclosure. At the end the push gets harder while the module plug slides into the female connector of the back plane.
 The module's front panel should butt against the enclosure.
- Fix the module with the screws at top and bottom of the front panel.
- If there are, close the gaps between plugged-in modules using the cover plates. There are plates with 2, 3 or 5 HP width.

4.1 ADwin-Pro II

The standard enclosure for the *ADwin-Pro II* systems. The backplane of the enclosure connects the processor module with the other modules.

The system fuse is located in a slot in the power supply unit above the female connector for the power supply cable (rear of the enclosure).

16 slots

Number of Slots	16
Main dimensions (l x w x h) Slot area (w x h)	336mm x 447.5mm x 146mm (incl. feet) 84 HP x 3 U
Power supply unit	100V...240V AC at 50/60Hz switching-mode power supply upper power limit >70W
Fuse	5A, slow-blow fuse

Fig. 1 – Enclosure *ADwin-Pro II*: Specification

At the rear of the enclosure, above the power supply connector you will find a label with the revision number:

Revision	Release	Previous versions
E1	Jun. 2006	<i>ADwin-Pro II</i> : New enclosure design and new back plane with Pro I and Pro II bus.

The Pro II enclosure is designed for both Pro I and Pro II modules: The back plane comprises the Pro I bus as well as the Pro II bus. The processor module runs both buses in parallel.

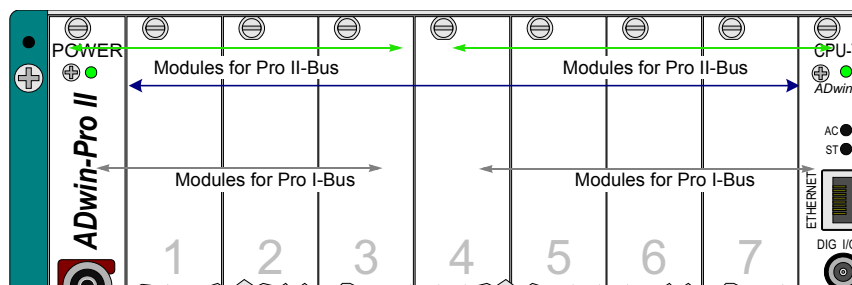


Fig. 2 – Enclosure *ADwin-Pro II*

Please note that modules for Pro I bus (grey in [fig. 2](#)) and for Pro II bus have different plug-in positions. You recognize the right position easily by the color of the bearings:

- White bearings: Modules for Pro II bus.
- Black bearings: Modules for Pro I bus.

Output modules Pro-AOut-x with Rev. A may not be used for technical reasons.

The processor module must be plugged-in at the middle position (white bearings).

There is a gap of half a slot between processor module and Pro I modules (cover plates accompanied), while Pro II modules fit directly besides the processor module.

Pro II-DC
with 16 slots



4.2 ADwin-Pro II-DC

The *ADwin-Pro II-DC* enclosure is similar to the standard enclosure *ADwin-Pro II*, but is equipped with a DC power supply.

If a current-limited power supply unit is used, it should be able to supply a multiple of the idle current during power-up to maintain proper performance of the system.

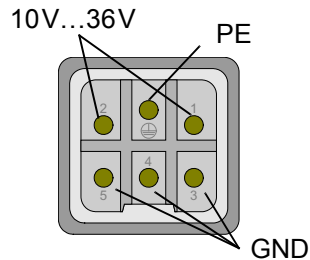


Fig. 3 – Enclosure *ADwin-Pro II-DC*:
Detailed view of the pin assignment

Number of Slots	16
Main dimensions (l x w x h)	336mm × 447.5mm × 146mm (incl. feet)
Slot area (w x h)	84 HP × 3 U
Power supply unit	DC-DC converter 10V...35V upper power limit >80W

Fig. 4 – Enclosure *ADwin-Pro II-DC*: Specification

At the rear of the enclosure, above the power supply connector you will find a label with the revision number:

Revision	Release	Änderung zur Vorgänger-Version
E1	Jun. 2005	<i>ADwin-Pro II</i> : New enclosure design and new back plane with Pro I and Pro II bus. New power supply female connector.

4.3 ADwin-Pro II-BM

In the version "backmounted" of the standard enclosure, the modules are plugged-in at the rear of the enclosure.

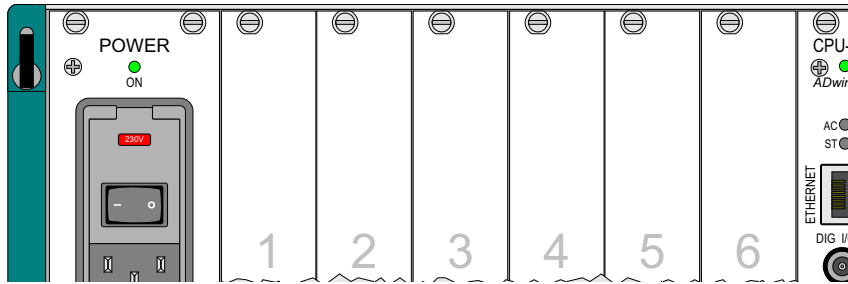


Fig. 5 – Enclosure of ADwin-Pro II-BM (rear panel)

The system fuse is located in a slot in the power supply unit above the female connector for the power supply cable (rear of the enclosure).

Number of Slots	16
Main dimensions (l x w x h)	336mm x 447.5mm x 146mm (incl. feet)
Slot area (w x h)	84 HP x 3 U
Power supply unit	100V...240V AC at 50/60Hz switching-mode power supply upper power limit >70W
Fuse	5A, slow-blow fuse

Fig. 6 – Enclosure of the ADwin-Pro II-BM: Specification

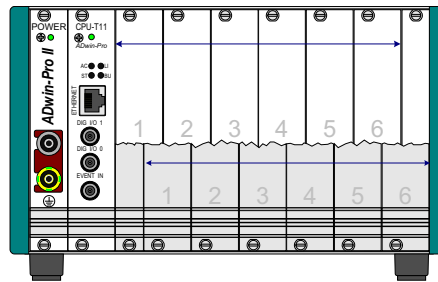
At the rear of the enclosure, above the power supply connector you will find a label with the revision number:

Revision	Release	Previous versions
E1	Jun. 2006	ADwin-Pro II: New enclosure design and new back plane with Pro I and Pro II bus.

Pro II "backmounted"
with 15 slots

Pro II-light with 7 slots

4.4 ADwin-Pro II-light



Enclosure ADwin-Pro II-light

The backplane of the enclosure connects the processor module with the other modules.

Number of Slots	7
Main dimensions (l x w x h) Slot area (w x h)	336mm x 234mm x 146mm (incl. feet) 42 HP x 3 U
Power supply unit	100...240V AC at 50/60Hz switching-mode power supply upper power limit >40W
Fuse	2A, slow-blow fuse

Fig. 7 – Enclosure ADwin-Pro II-light: Specification

At the rear of the enclosure, above the power supply connector you will find a label with the revision number:

Revision	Release	Previous versions
E1	Jun. 2006	ADwin-Pro II: New enclosure design and new back plane with Pro I and Pro II bus.

4.5 ADwin-Pro II-light-DC

The enclosure *ADwin-Pro II-light-DC* is similar to *ADwin-Pro II-light*, but is equipped with a DC power supply.

If a current-limited power supply unit is used, it should be able to supply a multiple of the idle current during power-up to maintain proper performance of the system.

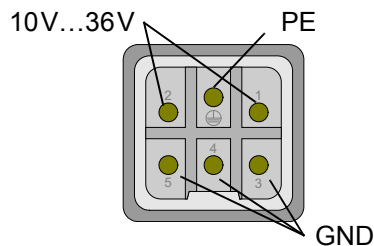


Fig. 8 – Enclosure *ADwin-Pro II-light-DC*: Connector for power supply

Number of Slots	7
Main dimensions (l x w x h)	336mm x 234mm x 146mm (incl. feet)
Slot area (w x h)	42 HP x 3 U
Power supply unit	DC-DC converter 10V...35V upper power limit >80W
Fuse	2A, slow-blow fuse

Fig. 9 – Enclosure *ADwin-Pro II-light-DC*: Specification

At the rear of the enclosure, above the power supply connector you will find a label with the revision number:

Revision	Release	Previous versions
E1	Oct. 2006	First version, <i>ADwin-Pro II</i> only.

The enclosure is delivered with a power supply female connector. The plug may be ordered as follows:

Manufacturer: Regional-Electronic-Distribution Handelsgesellschaft mbH
Postfach 1250, 63084 Rodgau

Connectors: Plug enclosure Harting, series HA.3.XX.X,
Order no. HA.3.STO.1.11
female connector inset Harting, 5+PE, 400V, 16A,
Order no. HE.Q.5.BU.C
4 pcs. female connector contact pin 2.5mm², gold-plated,
Order no. HE-HA.C.BU.2,5.AU

Pro II-light-DC
mit 7 Steckplätzen



Pro II-mini with 5 slots

4.6 ADwin-Pro II-mini

The smallest ADwin-Pro II-mini enclosure has 5 slots and requires an external power supply unit. The power supply connector is located at the rear of the enclosure.

Number of Slots	5
Main dimensions (l x w x h) Slot area (w x h)	253mm x 147.3mm x 146mm (incl. feet) 20 HP x 3 U
External power supply unit	external power supply unit required DC-DC converter 10V...36V DC upper power limit >50W

Fig. 10 – Enclosure ADwin-Pro-mini Specification

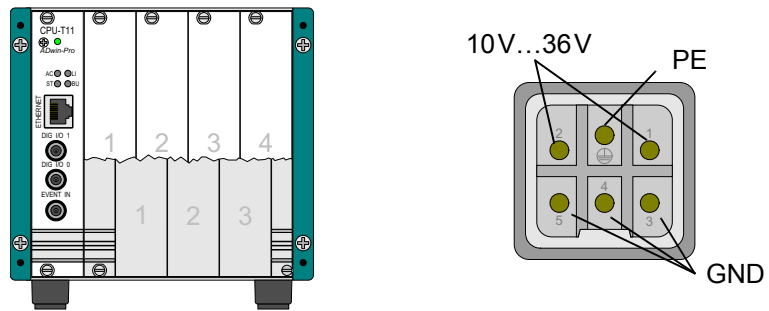


Fig. 11 – Enclosure ADwin-Pro II-mini and power supply connector

At the rear of the enclosure you will find a label with the revision number:

Revision	Release	Previous versions
E1	Dec. 2006	ADwin-Pro II: New enclosure design and new back plane with Pro I and Pro II bus. Variable input voltage.

5 ADwin-Pro Modules

An *ADwin-Pro* module needs one slot (5 HP) in an *ADwin-Pro* system, some modules need 2 slots.

All technical data of the module refer to a device, which is powered-up.

For pluggin-in a module into the enclosure please note the description on [page 4](#), especially with *ADwin-Pro II* enclosures.



5.1 Unsupported Pro I modules

Normally all Pro I modules are usable in an *ADwin-Pro II* enclosure if the processor module [Pro-CPU-T11-ENET](#) is used.

Nevertheless, the following modules are not supported in *ADwin-Pro II* systems anyway and can only be used in an *ADwin-Pro I* system:

- Pro-AOut-4/16 Rev. A
- Pro-AOut-8/16 Rev. A
- Pro-AO-16/8-12 Rev. A

5.2 Setting the module's addresses (Pro I modules, too)

Any *ADwin-Pro* module (except CPU modules) is addressed in an *ADbasic* program via its module address. The module address is free selectable.

Selecting a module's address

Note the following rules for selecting a module's address:

- A module address must be unique inside its module group.
 - Each module is member of a module group:
 - Pro I modules, functional group CPU: processor modules.
 - Pro I modules, functional group ADC: analog input modules.
 - Pro I modules, functional group DAC: analog output modules.
 - Pro I modules, functional group DIO: digital input/output modules, relays and counter modules.
 - Pro I modules, functional group EXT: special modules of all kind.
 - All Pro II modules.
- A module address must be within the following limits:
 - Pro I modules: 1 ... 255.
 - Pro II modules: 1 ... 15.

There are special limits for RSxxx- and fieldbus modules (see below)

It is true that you can select the same module address for modules of different groups. Nevertheless we recommend using unique addresses in order to prevent a mix-up.



Setting the module's address: Pro II modules

With Pro II modules, you set the module address with the program *ADpro*. Settin the address also resets the Pro modules to initial state.

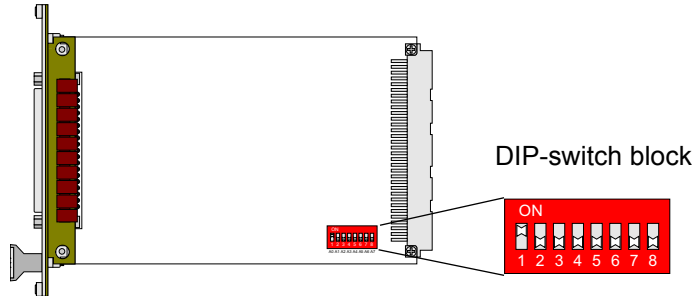
Switch off the power supply of the *ADwin* system and insert the Pro II modules into the casing; please note the description on [page 4](#). Power up the *ADwin* system again.

Afterwards you start the program *ADpro* from the Windows Start menu under Programs ▶ *ADwin*. Select the module address with the menu entry Edit ▶ Set module addresses.

Setting the module's address: Pro I modules

With Pro I modules, you set the module address manually via DIP-switches. The on-board block of DIP-switches is located right to the bottom.

With 8 DIP switches the address is selectable between 1 and 255 (see fig. 12). Each module of the same group needs to have a different address.



Module no.	Settings of DIP switches							
	1	2	3	4	5	6	7	8
1	1	0	0	0	0	0	0	0
2	0	1	0	0	0	0	0	0
3	1	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0
5	1	0	1	0	0	0	0	0
6	0	1	1	0	0	0	0	0
7	1	1	1	0	0	0	0	0
8	0	0	0	1	0	0	0	0
...				...				
254	0	1	1	1	1	1	1	1
255	1	1	1	1	1	1	1	1

Fig. 12 – Address settings of the ADwin-Pro modules with DIP switches



Please note:

- A RSxxx module with 4 serial interfaces uses 2 addresses (group EXT): the set address and the following address.
- A fieldbus module uses 32 addresses (group EXT); the address allocation is shown in fig. 13.

Set module address	Addit. allocated addresses	Settings of DIP switches							
		1	2	3	4	5	6	7	8
1	160...191	1	0	0	0	0	0	0	0
2	192...223	0	1	0	0	0	0	0	0
3	224...255	1	1	0	0	0	0	0	0
4	128...159	0	0	1	0	0	0	0	0

Fig. 13 – Address settings of fieldbus modules with DIP switches

5.3 Processor modules

For each *ADwin-Pro* system one processor module is required. This processor module is the center of a Pro system and performs the following tasks:

- Communication with PC or laptop.
The data connection is established via USB or Ethernet; former version used a serial link connection.
- Communication with other *ADwin-Pro* modules via internal Pro bus.
- Communication with possibly existing *TiCo* processors on *Pro II* modules via internal Pro bus.
- Start and run the user defined processes.

The processor module provides the memory for data and programs. At the time, the following processor modules are available for *ADwin-Pro II*:

- [Pro-CPU-T11-ENET](#)
- [Pro-CPU-T12](#)

The processor modules can be ordered with the options [Pro II-Boot with Storage Device](#) (additional external storage medium) and [Pro II-Boot](#) (stand-alone operation of the processor module).



5.3.1 Pro-CPU-T11-ENET

The processor modul can only be run in a Pro II casing and works both with Pro I and Pro II modules.

The output module Pro-AOut-x runs with T11 up from Rev. B.

For this module, there are several ordering options [Pro II-Boot with Storage Device](#), which provide access to a storage device. For example, the device enables data storage in stand-alone operation of the ADwin system for long-term measurement.

The additional properties of the options are described on [page 19](#).

To be used for Pro system	Pro II
Processor	ADSP TS101S
Clock rate	300MHz
Calculation resolution for float values	40 Bit
Data connection	Ethernet Interface ENET-2, up to 100 Mbit/s Interface ENET-3 (since Rev. E10), up to 1000 Mbit/s
Internal memory	PM: 256 KiB, DM: 256 KiB, EM: 256 KiB
External memory	256MiB
TTL-signal inputs	Event In, with 4,7kΩ pull-down resistor Dig I/O 0, with 4,7kΩ pull-up resistor Dig I/O 1, with 4,7kΩ pull-up resistor

Fig. 14 – Pro-CPU-T11-ENET: Spezifikation

The processor module has a fixed position in the casing. Please see the notes on how to [Plug-in a module](#) on [page 4](#).

The internal memory of the processor is divided into program memory (PM), data memory (DM) and free-for-use extra memory (EM). Each memory section has a size of 256 KiB.

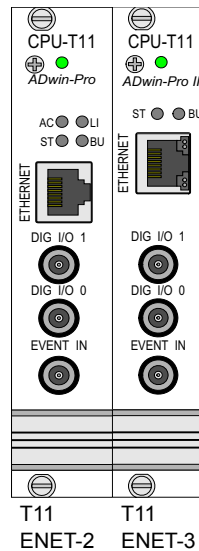


Fig. 15 – Pro-CPU-T11-ENET: Front panel

The processor module show the mode of operation by LEDs at the Ethernet connector. The meaning of the LEDs is described in manual ADwin Installation, chapter 10.5.

The external trigger input (`Event In`) enables the processor module to recognize an external signal as trigger for an event and trigger a process, that is processed immediately and completely (see *ADbasic* manual, chapter: Structure of the *ADbasic* program).

The event signal has to be present for 50ns to be recognized.

Alternatively, the event input of an other module may be used. All event signals arrive at the same signal line of the processor module as the input `Event In` does.

The event input can be configured in *ADbasic* with `CPU_Event_Config`.

The digital channels `DIG I/O 0` and `DIG I/O 1` use TTL signals and can be programmed as input or output.

After start-up, the `DIG I/O` channels are configured as inputs for falling edges.

The inputs and outputs of the CPU module are comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains the following instructions:

Function	Instructions
Configure digital channels	<code>CPU_Dig_IO_Config</code>
Query edge at digital input	<code>CPU_Digin</code>
Set level at digital output	<code>CPU_Digout</code>
Configure event input	<code>CPU_Event_Config</code>

Software changes when switching from T9 / T10

Existing *ADbasic* programs may – if still working with Pro I modules – be further used with few, but inevitable changes:

- The processor T11 needs the include file `<ADwinPro_All.inc>` to be included. In parallel, all other include files for Pro modules should be deleted from the program.
- The time unit of the `Processdelay` (cycle time) is $3.\bar{3}$ ns for both high priority and low priority processes.

All referring values and calculations must be adapted to the new time unit. The greatest possible `Processdelay` corresponds to 7.1 s; a greater cycle time can be achieved using an auxiliary variable.

- The time unit of $3.\bar{3}$ ns is also true for the internal counter, i.e. counter queries with `Read_Timer` must be adapted, too.

Please note: The process timing in connection with I/O instructions has become more complex (see below). Thus, a time difference determined with `Read_Timer` will now refer to a part of the total process timing only.

- The instruction `SLEEP` must be replaced by one of the following new instructions.
 - `CPU_SLEEP` causes the processor to wait. The instruction `SLEEP` had the same function with the processors T9 and T10.
 - `P1_SLEEP` causes the Pro I bus to wait, e.g. to co-ordinate I/O instructions.
 - `P2_SLEEP` causes the Pro II bus to wait, e.g. to co-ordinate I/O instructions.

The new instructions have a time unit of 10ns (`SLEEP`: 100ns).

Which instruction is right? Normally `SLEEP` is used to bridge the waiting time of an I/O instruction, e.g. the settling time of a multiplexer with

Input Event In

Digital Channels DIG I/O

Programming in ADbasic

PROCESSDELAY

READ_TIMER

SLEEP



SET_MUX. In this case, the instruction **P1_SLEEP** fits for previous modules (Pro I bus), and **P2_SLEEP** for Pro II modules.

Please see the notes in the *ADbasic* manual about Using Waiting Times (chapter 5.2.5).

Why are there new instructions? The processor T11 distinguishes processor instructions on the one hand and I/O instructions on the other hand. The processor architecture enables a quasi-parallel processing¹ of both instruction groups and obtains a much faster processing of *ADbasic* processes. This also means that the instruction groups are (mainly) processed independently in respect to timing. Since the process timing shall be controlled by waiting, there needs to be a separate instruction for each group. The separate instruction for each bus is required, because an I/O wait is effected by halting the appropriate bus.

1. The processor architecture differs from T9 and T10 in this point: T9 and T10 processed instructions of both groups sequentially. Thus, halting the processor with a *SLEEP* instruction did make the waiting time for subsequent I/O instructions, too.

5.3.2 Pro-CPU-T12

The processor module CPU-T12 can only be run in a Pro II casing and runs only with Pro II modules.

The module can be ordered with several options [Pro II-Boot with Storage Device](#) (details see [page 19](#)), which allow direct access to an external storage medium. It enables e.g. the data storage with stand-alone-operation of the ADwin system in long-time measurements.

Suitable for casing	Pro II
Processor	XILINX ZYNQ™ with Dual-Core ARM Cortex-A9
Clock rate	1000MHz
Calculation resolution for float values	64 Bit
Data connection	Interface ENET-3, up to 1000 Mbit/s
Internal memory	1000MiB, davon bis zu 6MiB cacheable
TTL-signal inputs	Event In, with 4,7kΩ pull-down resistor Dig I/O 0, with 4,7kΩ pull-up resistor Dig I/O 1, with 4,7kΩ pull-up resistor

Abb. 16 – Pro-CPU-T12: Specification

The processor module has a fixed position in the casing. Please see the notes on how to [Plug-in a module](#) on [page 4](#).

The processor module provides a memory with a size of 1000 MB. Internally the ADwin-CPU uses a cache memory, which allows to process program parts with particular speed.

The processor module shows the mode of operation by LEDs at the Ethernet connector. The meaning of the LEDs is described in manual ADwin Installation, chapter 10.5.

Input Event In

The external trigger input (Event In) enables the processor module to recognize an external signal as trigger for an event and trigger a process, that is processed immediately and completely (see *ADbasic* manual, chapter: Structure of the *ADbasic* program).

The event signal has to be present for 50ns to be recognized.

Alternatively, the event input of an other module may be used. All event signals arrive at the same signal line of the processor module as the input Event In does.

The event input can be configured in *ADbasic* with `CPU_Event_Config`.

Digital Channels DIG I/O

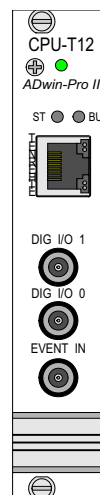
The digital channels DIG I/O 0 and DIG I/O 1 use TTL signals and can be programmed as input or output.

After start-up, the DIG I/O channels are configured as inputs for falling edges.

Programming in ADbasic

The inputs and outputs of the CPU module are comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains the following instructions:



Function	Instructions
Configure digital channels	<code>CPU_Dig_IO_Config</code>
Query edge at digital input	<code>CPU_Digin</code>
Set level at digital output	<code>CPU_Digout</code>
Configure event input	<code>CPU_Event_Config</code>

Software changes when switching from T9 / T10 / T11

Existing *ADbasic* programs may be further used with few changes as long as the programs do only access Pro II modules. The access to Pro I modules is not possible.

Please note:

PROCESSDELAY

- The time unit of the `Processdelay` (cycle time) is 1 ns for both high priority and low priority processes.

All referring values and calculations must be adapted to the new time unit. The greatest possible `Processdelay` corresponds to 2.1 s; a greater cycle time can be achieved using an auxiliary variable.

READ_TIMER

- The time unit of 1 ns is also true for the internal counter, i.e. counter queries with `Read_Timer` must be adapted, too.

Please note: The process timing in connection with I/O instructions has become more complex (find more in the *ADbasic* manual). A time difference determined with `Read_Timer` will now refer to a part of the total process timing only. For hardware access you may want to use the instruction `Read_Timer_Sync` as an alternative.

5.3.3 Pro II-Boot with Storage Device

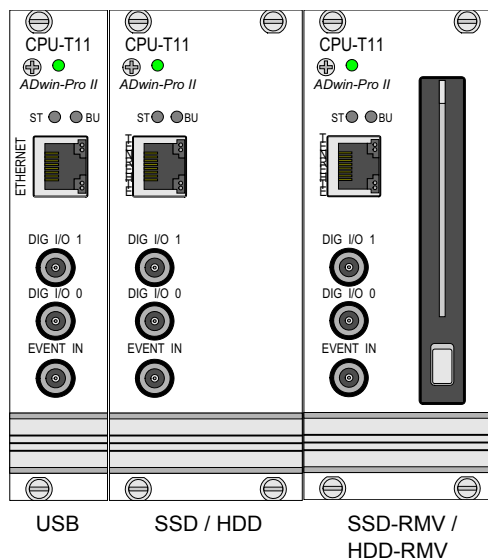
The processor modules [Pro-CPU-T12](#) and [Pro-CPU-T11-ENET](#) can be optionally equipped to access an additional storage device. An upgrade to this option is not possible.

For example, the storage device enables data storage in stand-alone operation of the *ADwin* system for long-term measurement.

The bootloader [Pro II-Boot](#) for stand-alone operation is always included, see description on [page 22](#). The following text describes the use of the additional storage device.

According to the order, the module is delivered with one of the following storage devices:

option	media	min. size	width	
Pro II-Boot-USB	USB flash	16GB	5 HP	
Pro II-Boot-SSD	SSD	240GB	10 HP	built-in
Pro II-Boot-SSD-RMV	SSD	240GB	10 HP	removable
Pro II-Boot-HDD	Hard disc	1TB	10 HP	built-in
Pro II-Boot-HDD-RMV	Hard disc	1TB	10 HP	removable



Inserting a storage device

You can use storage devices with form factor 2,5", max. width is 9.5mm.

This is how to insert a storage device:

- If the storage device is built with an open board, stick the delivered plastic sheet to the open board. Without sheet the board can get into contact with the casing and generate errors.

An instruction how to attach the plastic sheet or the distance piece (see below) is delivered with the storage device.

With a slim device, take the delivered distance piece and attach it to the left side of the storage device.

Some storage devices provide jumpers near to the connector strip. If so, take the delivered protective cap and slide the cap over the jumpers.



The protective cap prevents a contact between the insertion mechanism and the jumpers.

- Open the module flap as you push the „knob“ upward. Pull the flap slightly with the hand until the flap swings upwards (nearly horizontal).

An already inserted storage device will be pushed outward and can be withdrawn.

- Place the storage device with the connector strip facing in direction to the module and to the right.
- Push the storage device completely into the slot.
- Close the module flap with caution.

Attention: If you cannot close the flap or only with effort, the storage device is not inserted correctly. Remove the storage device again and check the correct position of the connector strip (see above). Repeat the inserting procedure.

File systems to be used

The module can format, read, and write storage devices with EXT3 or FAT32 file system. The EXT3 file system is case sensitive, FAT32 is not.

Please note, that *ADbasic* can read only files with a size of up 2 GiB. Using the EXT3 file system, you can create files of more than 2GiB by appending data (readable on a PC, but not in *ADbasic*). In the FAT32 file system, appending data to a file will result in an error if the maximum file size is exceeded.

A removable storage device can be read or written on other PCs with appropriate SATA interface.

Internal clock

Processor modules with option Pro II-Boot-Storage contain an internal clock, which is buffered by an exchangeable battery. The module uses the clock to add a time stamp to file operations. You can query the clock time with an *ADbasic* instruction. The clock is not suitable as time stamp for measurement results.

The clock is set ex works, afterwards the module automatically synchronizes with a time server in the internet as soon as internet access is available. At the moment, the clock cannot be set manually.

Programming in ADbasic

The access to the storage device is comfortably programmed with *ADbasic* instructions. The instructions are described in the online help of *ADbasic* and in the *ADwin-Bootloader* manual.

The include file `ADwinPro_All.inc` contains the following instructions:

Function	Instruction
Set up the storage device	<code>Storage_Create_Partitions_xxx</code> <code>Storage_Format_xxx</code>
Mount the file system	<code>Storage_Mount_xxx</code> <code>Storage_Unmount_xxx</code>
File operations	<code>Storage_File_Create_xxx</code> <code>Storage_File_Delete_xxx</code> <code>Storage_File_Exists_xxx</code> <code>Storage_File_Status_xxx</code> <code>Storage_File_Move_xxx</code>
Read or write data	<code>Storage_File_Write_xxx</code> <code>Storage_File_Read_xxx</code> <code>Storage_File_Append_xxx</code>
Directory operations	<code>Storage_Dir_Create_xxx</code> <code>Storage_Dir_Delete_xxx</code> <code>Storage_Dir_Exists_xxx</code> <code>Storage_Dir_Status_xxx</code>

Software



5.3.4 Pro II-Boot

With Pro II-Boot, you have a boot loader expansion, which can

- boot an *ADwin-Pro II* system.
- load up to 10 processes.
- start process 10 automatically (if present).
- save data.

Pro II-Boot is an ordering option for processor modules with Ethernet interface. An upgrade is not possible.

By installation of the *ADbasic* and the *ADwin* drivers from the CDROM, all files / programs necessary for the boot loader option have already been copied to the hard disk.

If you use the boot loader, an application, which you have written with a program for visualization of measurement data, must not reboot the *ADwin* system.

5.3.5 Module monitoring with Watchdog

You can monitor your processor module with a watchdog. The watchdog generates a reset, when a signal, generated by a program code, does unexpectedly not arrive (see also "*ADwin-Pro* System Specifications - Programming in *ADbasic*").

The watchdog reset signal sets the digital and analog outputs at Pro modules to those values, which correspond to the configuration after power-up, normally digital 0 or 0 Volt. Please note that Pro II modules will only react to the watchdog signal if the T11 processor has revision E06 at least.

Programmig in ADbasic

The watchdog is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains the watchdog instructions:

Function	Instructions
Start and stop the watchdog	<code>StartWatchdog</code> , <code>StopWatchdog</code>
Reset the watchdog counter	<code>ResetWatchdogTimer</code>



Notes in relation to Pro II-Flash-Boot:

- Please pay attention to the fact that the watchdog has to be reset every 1.6s, since a longer time interval between two impulses will be interpreted as an error.
- The watchdog can also be used with the boot loader Pro-Flash-Boot, but does not automatically load and start the software.
- Test your programs always with the watchdog switched off. Activate it only when your programs work properly!



5.4 Pro II: Multi-IO Modules

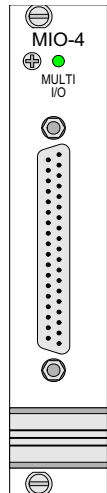
Module	Rev.	Type	Channels	Specific Data	Page
MIO-4	E	TTL input / output	8	U_{In} 5V TTL	24
		Analog input	16 s.e. 8 diff.	1 ADC, 18 bit, $\pm 10V$ Max. conversion time $2\mu s$ max. sample 500 ksample/s	
		Analog output	4	4 DAC, 16 Bit, $\pm 10V$ Settling time $9\mu s$	
		TiCo1 Processor	–	128 KiByte internal memory 4MiByte external SRAM	
MIO-4-ET1	E	TTL input / output	8	U_{In} 5V TTL	31
		Analog input	16 s.e. 8 diff.	1 ADC, 18 bit, $\pm 10V$ Max. conversion time $2\mu s$ max. sample 500 ksample/s	
		Analog output	4	4 DAC, 16 Bit, $\pm 10V$ Settling time $9\mu s$	
		Transistor output	4	$5\dots 30 V_{DC}$ (external), 200mA	
		Optocouple input	4 s.e.	selectable 5V, 12V, 24V	
		Counter block	1	Universal, 32 Bit, 5V diff.	
		SSI decoder	1	max. 12.5MHz	
		EtherCAT interface	1	Slave interface	
MIO-D12	E	Transistor output	12	$5\dots 30 V_{DC}$ (external), 200mA	44
		Optocouple input	12 s.e.	selectable 5V, 12V, 24V	
		Counter block	2	Universal, 32 Bit, 5V diff.	
		SSI decoder	1	max. 12.5MHz	
		TiCo1 Processor	–	56 KiByte internal memory	

5.4.1 Pro II-MIO-4 Rev. E

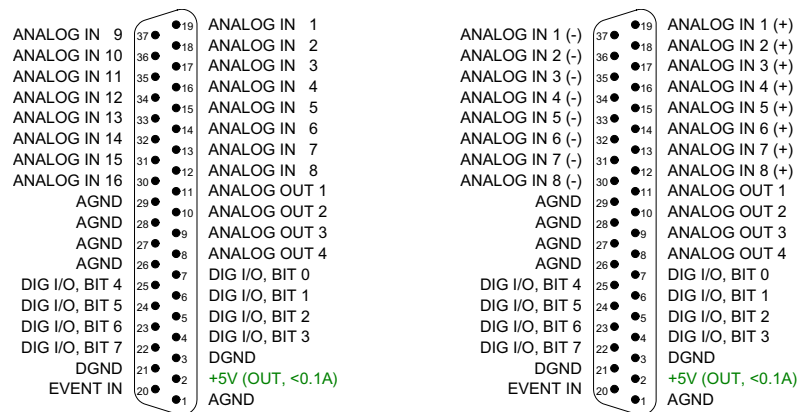
The module [Pro II-MIO-4 Rev. E](#) is equipped with the following hardware:

- 16 analog inputs (16 single-ended or 8 differential), via multiplexer with 18 bit ADC, sample rate up to 500kHz
- 4 analog outputs with 16 Bit DAC.
- 8 digital input / output channels with TTL levels
- 1 Event input
- *TiCo* processor with 128 KiB internal memory and 4 MiB external SRAM memory

If the *TiCo* bootloader is programmed, the module can work on its own and independently from the CPU module of the *ADwin-Pro II* system.



The extended module version [Pro II-MIO-4-ET1 Rev. E](#) is described starting on [page 31](#).



Analog inputs single-ended

Analog outputs differential

Fig. 17 – [Pro II-MIO-4 Rev. E](#): Pin assignment

The module functions are described in the following sections:

- [Analog Inputs](#)
- [Analog Outputs](#)
- [Digital Inputs / Outputs](#)
- [TiCo processor](#)
- [Technical Specification](#)
- [Programming](#)

Analog Inputs

The module [Pro II-MIO-4 Rev. E](#) has 16 single ended inputs or 8 differential inputs (selectable by software). After power-up, the module is set to 8 differential inputs.

The inputs are equipped with a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

The inputs are connected via a multiplexer to the ADC. The ADC has a resolution of 18 bit and can run with a sampling rate of up to 500kSamples/s.

The module has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can monitor each input channel, if an upper or a lower limit—you can set the limits for each input channel separately—has been exceeded.

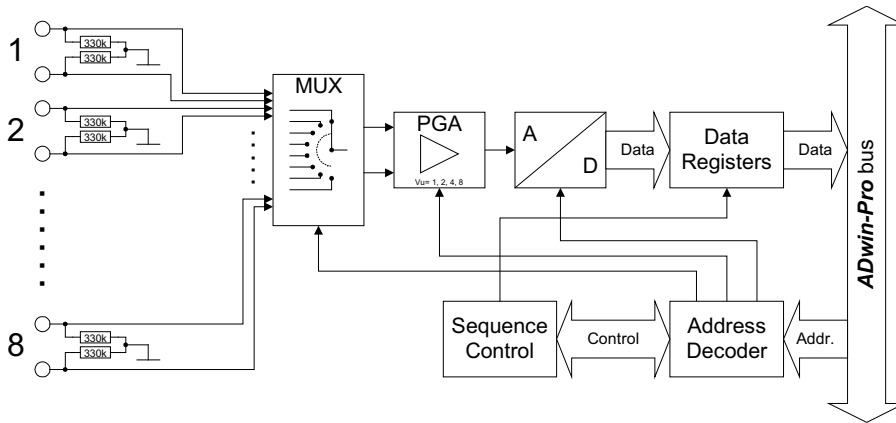


Fig. 18 – Pro II-MIO-4 Rev. E: Block diagram of analog inputs

Analog Outputs

The module Pro II-MIO-4 Rev. E has 4 analog outputs each with a 16 bit DAC. Each output has a fixed 1st order low-pass filters ($f_c = 10MHz$) to suppress noise.

The output voltage range of the DACs is set to $\pm 10V$ bipolar and can't be changed. Offset and gain are adjusted by software (see [chapter 6 "Calibration"](#)).

The outputs are available on a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

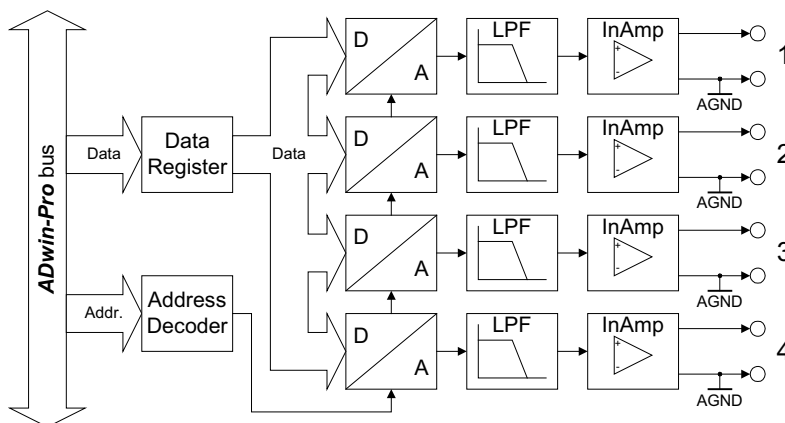


Fig. 19 – Pro II-MIO-4 Rev. E: Block diagram of analog outputs

Digital Inputs / Outputs

The digital input/output module Pro II-MIO-4 Rev. E provides 8 programmable digital input and output channels with TTL levels. The channels can be configured as blocks of 4 as inputs or outputs by *ADbasic* instructions. The channels are configured as inputs after power up.

Via the trigger input EVENT a signal can trigger a process, which will be processed at once and completely (see *ADbasic* manual).

The digital channels are available on a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

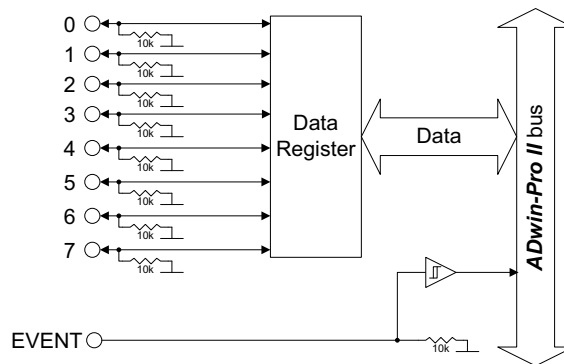


Fig. 20 – Pro II-MIO-4 Rev. E: Block diagram of digital channels

TiCo processor

The module provides the freely programmable *TiCo* processor with 64 kiB program memory, 64 kiB data memory, and 4 MiB external SRAM memory. The internal memory serves as data and program memory. You program the *TiCo* processor with *TiCoBasic*.

The *TiCo* processor has access to all analog and digital input and output channels, as does the *ADwin* CPU. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Technical Specification

Analog Inputs	
Input channels	16 single ended / 8 differential. via multiplexer
Resolution	18 Bit
Conversion time	max. 2 μ s
Sampling rate	max. 500ksps
Multiplexer settling time	5 μ s
Measurement range	\pm 10V
Gain	1, 2, 4, 8 selectable by software
Accuracy INL	typical \pm 4 LSB
Accuracy DNL	max. \pm 1 LSB
Input resistance	330k Ω . \pm 2%
Input over-voltage	\pm 20V
Offset error	adjustable
Offset drift	\pm 30ppm/ $^{\circ}$ C
Analog Outputs	
Output channels	4
Resolution	16 Bit
Settling time	9 μ s (to 0.01% FSR)
Output voltage	\pm 10V
Output current max.	\pm 5mA per channel for optimal function
Accuracy INL	\pm 2 LSB typical
Accuracy DNL	\pm 1 LSB typical
Offset error	adjustable
Gain error	adjustable
Digital Inputs / Outputs	
Digital inputs / outputs	8 channel with TTL logic, configurable in groups of 4 channels
Pull down resistor	10k Ω
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1 μ A
I _{IL}	max. 0.01 mA
Voltage range	-0.5V ... +5.5V
Output current	max. \pm 24mA per channel, max. \pm 50mA per block (4 channels) via V _{CC} or GND
Event input	TTL Logic
Power up status	All channels as inputs
General	
TiCo processor	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal, 4MiByte SRAM external
Connector	37-pin D-Sub female connector

Fig. 21 – Pro II-MIO-4 Rev. E: Specification

Programming in ADbasic

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Range	Instructions
Analog Inputs	
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_SEQ_Read_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Analog Outputs	
Output values	<code>P2_DAC, P2_DAC4, P2_DAC4_Packed</code>
Output values step by step	<code>P2_Write_DAC, P2_Write_DAC4</code> <code>P2_Write_DAC4_Packed</code> <code>P2_Write_DAC32</code> <code>P2_Start_DAC</code>
Digital Inputs / Outputs	
Configure input/outputs	<code>P2_MIO_DigProg</code>
Query input signals	<code>P2_MIO_Digin_Long</code>
Use latch register	<code>P2_MIO_Dig_Latch</code> <code>P2_MIO_Dig_Read_Latch</code> <code>P2_MIO_Dig_Write_Latch</code>
Set and read back output signals	<code>P2_MIO_Digout</code> <code>P2_MIO_Digout_Long</code> <code>P2_MIO_Get_Digout_Long</code>
General	
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `MIO_TiCo.inc` contains instructions for the functions:

Range	Instructions
Analog Inputs	
Set inputs to single-ended or differential	<code>SE_Diff</code>
Do a single conversion – complete or step by step	<code>ADC, ADC24</code> <code>Set_Mux, Start_Conv, Wait_EOC</code> <code>Read_ADC, Read_ADC24</code>
Read value and start new conversion	<code>Read_ADC_SConv</code> <code>Read_ADC_SConv24</code>
Use sequence control	<code>Seq_Init, Seq_Start</code> <code>Seq_Read, Seq_Read24</code> <code>Seq_Wait</code>
Control input limits	<code>ADC_Read_Limit</code> <code>ADC_Set_Limit</code>
Analog Outputs	
Output values	<code>DAC</code>
Output values step by step	<code>Write_DAC, Write_DAC32</code> <code>Start_DAC</code>
Digital Inputs / Outputs	
Configure input/outputs	<code>MIO_DigProg</code>
Query input signals	<code>MIO_Digin_Long</code>
Use latch register	<code>MIO_Dig_Latch</code> <code>MIO_Dig_Read_Latch</code> <code>MIO_Dig_Write_Latch</code>
Set and read back output signals	<code>MIO_Digout</code> <code>MIO_Digout_Long</code> <code>MIO_Get_Digout_Long</code>
General	
Use LED	<code>Check_LED, Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable, Event_Eead</code> <code>Event_Config, Trigger_Event</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>

Programming in TiCoBasic

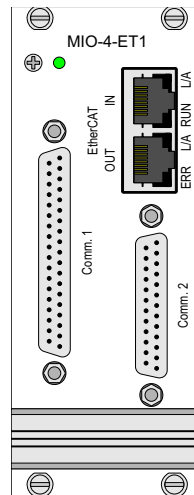
Programming TiCo access

Function	Instructions
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

5.4.2 Pro II-MIO-4-ET1 Rev. E

The module [Pro II-MIO-4-ET1 Rev. E](#) is equipped with the following hardware:

- 16 analog inputs (16 single-ended or 8 differential), via multiplexer with 18 bit ADC, sample rate up to 500kHz
- 4 analog outputs with 16 Bit DAC.
- 8 digital input / output channels with TTL levels
- 1 Event input
- 4 transistor outputs (TRA)
- 4 optically isolated inputs (OPT)
- 1 counter block with two 32-bit counters:
 - one up/down counter with clock/direction or four edge evaluation for connection of encoders.
 - one PWM counter to evaluate high and low times, duty cycle, or frequency.
- 1 SSI decoder to connect an incremental encoder
- 1 Ethercat interface (slave)
- *TiCo* processor with 128 KiByte internal memory and 4MiB external SRAM memory



The *TiCo* processor has access to all inputs and outputs of the module. For more information about usage and programming of the *TiCo* processor please see the manual *TiCoBasic*.

If the *TiCo* bootloader is programmed, the module can work on its own and independently from the CPU module of the *ADwin-Pro II* system.

The module variant [Pro II-MIO-4 Rev. E](#) is described starting on [page 24](#).

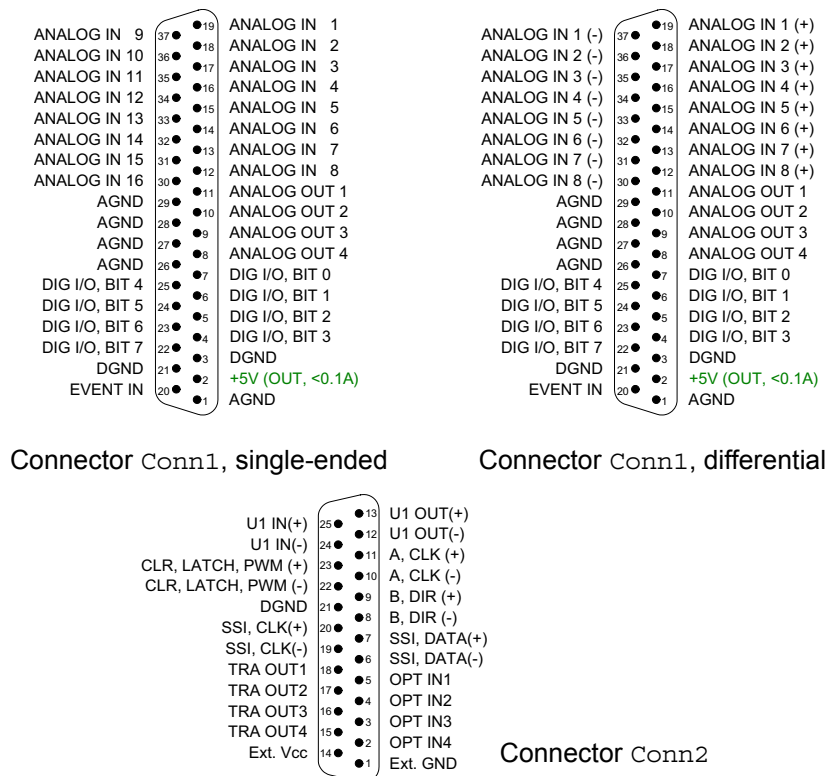


Fig. 22 – Pro II-MIO-4-ET1 Rev. E: Pin assignment

The module functions are described in the following sections:

- Analog Inputs
- Analog Outputs
- Digital Inputs / Outputs
- Transistor Outputs
- Optocouple Inputs
- SSI Decoder
- Counter block
- EtherCAT Interface
- TiCo processor
- Technical Specification
- Programming

Analog Inputs

The module [Pro II-MIO-4-ET1 Rev. E](#) has 16 single ended inputs or 8 differential inputs (selectable by software). After power-up, the module is set to 8 differential inputs.

The inputs are equipped with a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

The inputs are connected via a multiplexer to the ADC. The ADC has a resolution of 18 bit and can run with a sampling rate of up to 500kSamples/s.

The module has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can monitor each input channel, if an upper or a lower limit—you can set the limits for each input channel separately—has been exceeded.

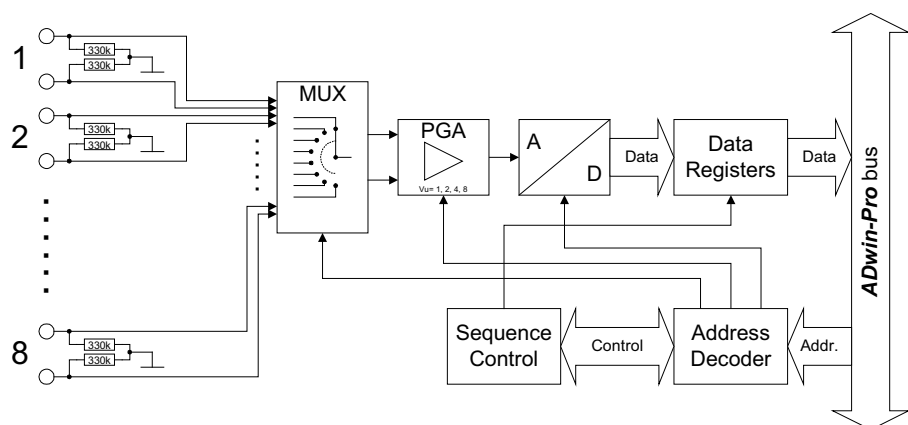


Fig. 23 – [Pro II-MIO-4-ET1 Rev. E](#): Block diagram of analog inputs

Analog Outputs

The module [Pro II-MIO-4-ET1 Rev. E](#) has 4 analog outputs each with a 16 bit DAC. Each output has a fixed 1st order low-pass filters ($f_c = 10MHz$) to suppress noise.

The output voltage range of the DACs is set to $\pm 10V$ bipolar and can't be changed. Offset and gain are adjusted by software (see [chapter 6 "Calibration"](#)).

The outputs are available on a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

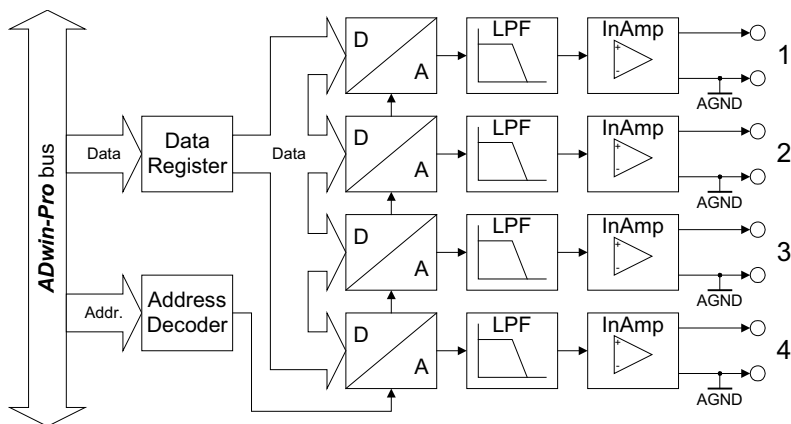


Fig. 24 – Pro II-MIO-4-ET1 Rev. E: Block diagram of analog outputs

Digital Inputs / Outputs

The digital input/output module [Pro II-MIO-4-ET1 Rev. E](#) provides 8 programmable digital input and output channels with TTL levels. The channels can be configured as blocks of 4 as inputs or outputs by *ADbasic* instructions. The channels are configured as inputs after power up.

Via the trigger input EVENT a signal can trigger a process, which will be processed at once and completely (see *ADbasic* manual).

The digital channels are available on a 37-pin D-Sub female connector; for pin assignment see [fig. 17](#).

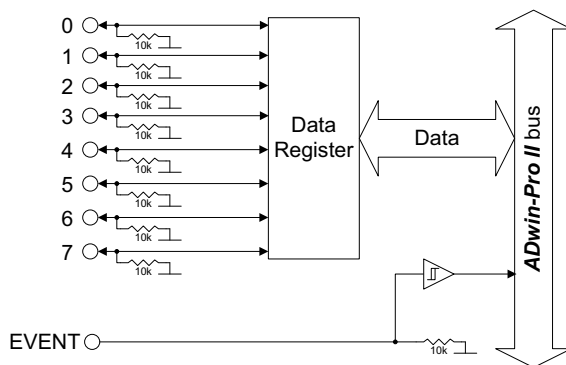


Fig. 25 – Pro II-MIO-4-ET1 Rev. E: Block diagram of digital channels

Transistor Outputs

The module [Pro II-MIO-4-ET1 Rev. E](#) provides 4 channels of galvanically isolated transistor outputs. The outputs switch to V_{CC} .

The common switching voltage V_{CC} has to be provided by an external power supply at the pins *EXT VCC* and *EXT GND*. Please note that *EXT GND* is also used for optocouple inputs.

The channels as well as the event-input are optically isolated from system circuitry.

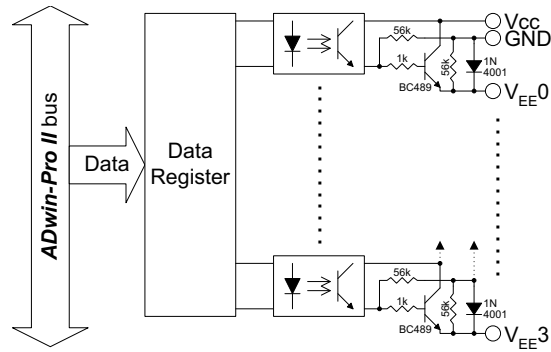


Fig. 26 – Pro II-MIO-4-ET1 Rev. E: Block diagram transistor outputs

The outputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see fig. 22.

Optocouple Inputs

The module Pro II-MIO-4-ET1 Rev. E provides 4 channels of optically isolated digital inputs. The input voltage range can be set by jumpers (5V, 12V, 24V). The default setting of the input voltage range is 24V. The switching time of only 100ns allows the sampling of high-speed digital inputs.

Each channel is optically isolated from the system circuitry and from the other inputs. The event-input is optically isolated from the system as well.

Please not that EXT GND is also used for transistor outputs.

The inputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see fig. 22.

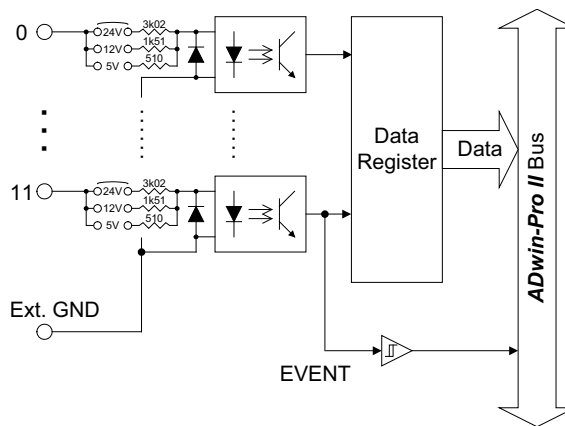


Fig. 27 – Pro II-MIO-4-ET1 Rev. E: Block diagram optocouple inputs

SSI Decoder

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

You can set the clock rate (6kHz ... 12.5MHz) as well as the data resolution (up to 32 bit) of the decoder via software.

A decoder either reads out an individual value (on request) or continuously provides the current value.

Via the two pins U1 IN you can supply a voltage, which is then available at the pins U1 OUT e.g. for external incremental encoders. The maximum current at each of the output pins is 100mA; there is an appropriate fuse between each input and output pin.

The decoder inputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see [fig. 22](#).

Counter block

The module Pro II-MIO-4-ET1 Rev. E provides a configurable multi-purpose counter block. The counter block contains two 32-bit counters: First an up/down counter or four edge evaluation for connection of encoders. Second, a PWM counter to evaluate high and low times, duty cycle, or frequency. Both counters of a block can be operated in parallel.

The counter inputs run differential.

The up/down counter of a block can be operated in 2 modes:

- Clock / direction (CLK and DIR signals)

A negative edge at the CLK input is the counting impulse for the 32-bit counter. The DIR signal sets the counting direction, TTL high means a count-up, TTL low means a count-down.

You can latch the counter values program-controlled or you can influence the counter by an external CLR/LATCH signal.

Depending on the programming the CLR/LATCH signal has either the effect that the counter values are cleared (CLR) or that the counter values are latched (LATCH). This function will only be effective when it is released by the instructions `P2_CNT_CLEAR_ENABLE` or `P2_CNT_LATCH_ENABLE`.

The counter is cleared or latched with a rising edge at input CLR/LATCH. During the latch process the frequency of the measurement can be determined by getting the difference of two read latch values, because this difference defines the number of pulses between the two reading processes.

- Four edge evaluation (A and B signals)

The four edge evaluation changes the signals (which should be 90° phase-shifted) of a connected incremental encoder at the inputs A and B to CLK and DIR signals. For this you have to program the inputs correspondingly (see "ADwin-Pro System Description, Programming in ADbasic").

Since every edge of the A and B signals generates a count impulse, the resolution is increased by factor 4. If the encoder has a reference signal, it can be used to clear or latch the counter (after release of the CLR or LATCH input). The counter is cleared when the signals A, B and CLR are on logic "1" (software-selectable: clear, when only the CLR signal is on logic "1").

The PWM counter of the counter block analyzes the signals at the PWM inputs. Via software instructions the following data can be read directly:

- frequency and duty cycle (with `P2_Cnt_Get_PW`)
- high and low time (with `P2_Cnt_Get_PW_HL`)

The counter inputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see [fig. 22](#).

If PWM counters are evaluated with standard instructions, no further knowledge is required.

If you require other evaluation methods, you can use several registers assigned to each PWM counter to create a solution. You find a description of PWM registers on [page 140](#) (module Pro II-CNT-x Rev. E). Use the evaluation with PWM registers for special solutions only.

Up/down counter

PWM Counter

EtherCAT Interface

The module provides a fieldbus node with the functionality of an EtherCAT slave. All settings are done via software.

After power-on you must initialize the fieldbus node in *TiCoBasic*. The initialization determines the size of the input and output areas.

There is a range each for data input and data output; each range has a size of 16 Longs or 64 bytes. Please note, that the terms "input" and "output" are used as the fieldbus controller sees them.

The interface has a plug connector of type RJ45 for both data input (IN) and data output (OUT). Each connector has a top LED "Link / Activity" (L/A), which displays the operating status of the node in the EtherCAT bus. The two other LEDs display the status of the EtherCAT state machine (RUN) and the occurrence of communication errors (ERR).

LED	Status	Meaning
Link / Acti- vity	off	Offline (or no power).
	green	Fieldbus node online, no data exchange.
	green, flickering	Fieldbus node online, with data exchange.
RUN	off	Status INIT: interface being initialized (or no power).
	blinks green	Status PRE-OP: Interface has contact to bus master.
	flashes green once	Status SAFE-OP: Interface can read data from the bus, but not send.
	green	Status OP: Interface is completely ready, inputs and outputs are active.
	red	Status EXCEPTION.
ERR	off	No error (or no power).
	blinks red	Invalid configuration.
	flashes red once	Local error in the interface; EtherCAT status has been changed.
	flashes red twice	Application watchdog timeout.
	red	Critical communication error.

Fig. 28 – Pro II-MIO-4-ET1 Rev. E: Meaning of EtherCAT LEDs

If both LEDs RUN and ERR turn red, a serious error has occurred in the interface. Please inform the support of Jäger Messtechnik; you find the address on the inner side of the cover page of the manual.

The EtherCAT slave can only be accessed from *TiCoBasic*. You initialize slave using the instruction **MIO_ECAT_Init**. Afterwards, you project the EtherCAT bus with a configuration tool suitable for the bus master, e.g. the program "TwinCAT System Manager" of the Beckhoff company. To do so, you have to copy the description file `ADwin-EtherCAT.xml` of the EtherCAT interface from the directory `C:\ADwin\Fieldbus\EtherCAT\` to the directory of the configuration tool.

TiCo processor

The module provides the freely programmable *TiCo* processor with 64 kiB program memory, 64 kiB data memory, and 4 MiB external SRAM memory. The

internal memory serves as data and program memory. You program the *TiCo* processor with *TiCoBasic*.

The *TiCo* processor has access to all analog and digital input and output channels, as does the *ADwin* CPU. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Technical Specification

Analog Inputs	
Input channels	16 single ended / 8 differential. via multiplexer
Resolution	18 Bit
Conversion time	max. 2 μ s
Sampling rate	max. 500ksps
Multiplexer settling time	5 μ s
Measurement range	± 10 V
Gain	1, 2, 4, 8 selectable by software
Accuracy INL	typical ± 4 LSB
Accuracy DNL	max. ± 1 LSB
Input resistance	330k Ω . $\pm 2\%$
Input over-voltage	± 20 V
Offset error	adjustable
Offset drift	± 30 ppm/ $^{\circ}$ C
Analog Outputs	
Output channels	4
Resolution	16 Bit
Settling time	9 μ s (to 0.01% FSR)
Output voltage	± 10 V
Output current max.	± 5 mA per channel for optimal function
Accuracy INL	± 2 LSB typical
Accuracy DNL	± 1 LSB typical
Offset error	adjustable
Gain error	adjustable
Digital Inputs / Outputs	
Digital inputs / outputs	8 channel with TTL logic, configurable in groups of 4 channels
Pull down resistor	10k Ω
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1 μ A
I _{IL}	max. 0.01 mA
Voltage range	-0.5V ... +5.5V

Fig. 29 – Pro II-MIO-4-ET1 Rev. E: Specification

Output current	max. ± 24 mA per channel, max. ± 50 mA per block (4 channels) via V_{CC} or GND
Event input	TTL Logic
Power up status	All channels as inputs
Transistor Outputs	
Output channels	4
Switching voltage V_{CC}	5...30V DC durch externe Spannungsversorgung
Switching current	200 mA max. per channel
Voltage drop	0.5V
Switching time	2.5 μ s
Isolation	42V channel-channel / channel-GND common ground for all OPT and TRA channels
Optocouple Inputs	
Input channels	4
Input current	typ. 3.5 mA / max. 7.5 mA
Input voltage range (selectable via jumpers)	0...5V / 0...12V / 0...24V
Switching threshold for 0-low	0...0.8V / 0...1.6V / 0...3.2V
Switching threshold for 1-high	4.5...5V / 10...12V / 20...24V
Input over-voltage	-5V ... 8V / -5V ... 16V / -5V ... 30V /
Switching time	100 ns
Isolation	42V channel-channel / channel-GND common ground for all OPT and TRA channels
Counter	
Number	1 universal counter block
Counter resolution	32 bit
Reference clock	50 MHz
Clock frequency four edge evaluation	12.5 MHz max. (at 90° phase-shift of the signals)
Clock frequency up/down counter	15 MHz max.
Reference frequency PWM analysis	100 MHz
Input / output level	compatible to RS422/485 (5V differential, 120 Ω bus terminating resistor)
Isolation	none
SSI Decoder	
Number	1
Clock frequency SSI decoder (CLK)	6 kHz ... 12.5 MHz
Ethercat Interface	
Number	1, to be programmed in <i>TiCoBasic</i>
General	

Fig. 29 – Pro II-MIO-4-ET1 Rev. E: Specification

TiCo processor	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal, 4MiByte SRAM external
Connectors	37-pin D-Sub female connector Conn1 37-pin D-Sub female connector Conn2 2 female connectors type RJ45 for EtherCAT data input (IN) and output (OUT)
Module width	10 HP

Fig. 29 – Pro II-MIO-4-ET1 Rev. E: Specification

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Range	Instructions
Analog Inputs	
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_SEQ_Read_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Analog Outputs	
Output values	<code>P2_DAC, P2_DAC4, P2_DAC4_Packed</code>
Output values step by step	<code>P2_Write_DAC, P2_Write_DAC4</code> <code>P2_Write_DAC4_Packed</code> <code>P2_Write_DAC32</code> <code>P2_Start_DAC</code>
Digital Inputs / Outputs	
Configure input/outputs	<code>P2_MIO_DigProg</code>
Query input signals	<code>P2_MIO_Digin_Long</code>
Use latch register	<code>P2_MIO_Dig_Latch</code> <code>P2_MIO_Dig_Read_Latch</code> <code>P2_MIO_Dig_Write_Latch</code>
Set and read back output signals	<code>P2_MIO_Digout</code> <code>P2_MIO_Digout_Long</code> <code>P2_MIO_Get_Digout_Long</code>
Transistor Outputs	

Programming in ADbasic

Programming in
TiCoBasic

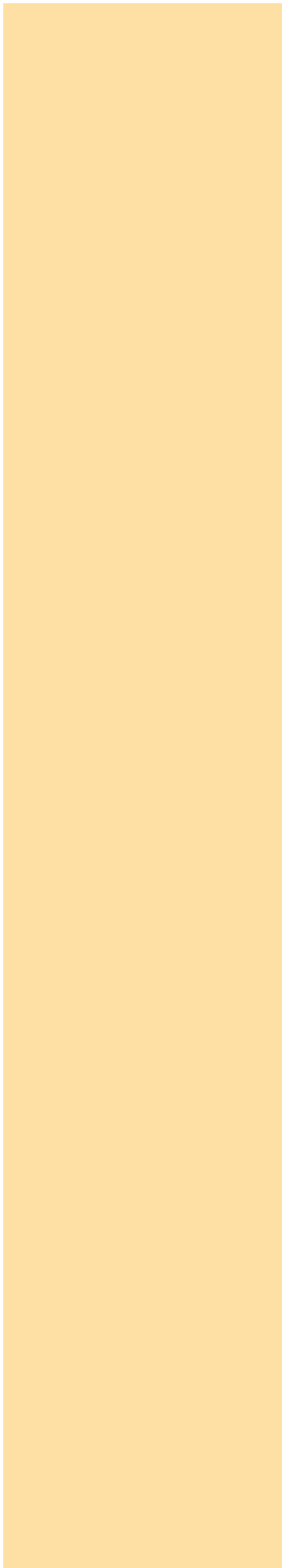
Range	Instructions
Set and reset output signals	<code>P2_Digout</code> , <code>P2_Digout_Long</code> <code>P2_Digout_Bits</code> <code>P2_Get_Digout_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Sync_All</code> <code>P2_Dig_Write_Latch</code>
Optocouple Inputs	
Query input signals	<code>P2_Digin_Edge</code> <code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Sync_All</code> <code>P2_Dig_Read_Latch</code>
Counter Block	
Configure counter	<code>P2_Cnt_Enable</code> , <code>P2_Cnt_Mode</code>
Use counter	<code>P2_Cnt_Clear</code> , <code>P2_Cnt_Get_Status</code> <code>P2_Cnt_Latch</code> , <code>P2_Cnt_Sync_Latch</code> <code>P2_Cnt_Read</code> , <code>P2_Cnt_Read4</code> <code>P2_Cnt_Read_Latch</code> <code>P2_Cnt_Read_Latch4</code>
Use PWM counter	<code>P2_Cnt_PW_Enable</code> , <code>P2_Cnt_PW_Latch</code> <code>P2_Cnt_Get_PW</code> , <code>P2_Cnt_Get_PW_HL</code> , <code>P2_Cnt_Read_Int_Register</code>
SSI Decoder	
Use SSI decoder	<code>P2_SSI_Mode</code> , <code>P2_SSI_Set_Bits</code> <code>P2_SSI_Set_Clock</code> <code>P2_SSI_Set_Delay</code> , <code>P2_SSI_Read</code> <code>P2_SSI_Read2</code> <code>P2_SSI_Start</code> , <code>P2_SSI_Status</code>
EtherCat Interface	
Use interface	only possible in <i>TiCoBasic</i>
General	
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `MIO_TiCo.inc` contains instructions for the functions:

Range	Instructions
Analog Inputs	
Set inputs to single-ended or differential	<code>SE_Diff</code>
Do a single conversion – complete or step by step	<code>ADC</code> , <code>ADC24</code> <code>Set_Mux</code> , <code>Start_Conv</code> , <code>Wait_EOC</code> <code>Read_ADC</code> , <code>Read_ADC24</code>
Read value and start new conversion	<code>Read_ADC_SConv</code> <code>Read_ADC_SConv24</code>

Range	Instructions
Use sequence control	<code>Seq_Init</code> , <code>Seq_Start</code> <code>Seq_Read</code> , <code>Seq_Read24</code> <code>Seq_Wait</code>
Control input limits	<code>ADC_Read_Limit</code> <code>ADC_Set_Limit</code>
Analog Outputs	
Output values	<code>DAC</code>
Output values step by step	<code>Write_DAC</code> , <code>Write_DAC32</code> <code>Start_DAC</code>
Digital Inputs / Outputs	
Configure input/outputs	<code>MIO_DigProg</code>
Query input signals	<code>MIO_Digin_Long</code>
Use latch register	<code>MIO_Dig_Latch</code> <code>MIO_Dig_Read_Latch</code> <code>MIO_Dig_Write_Latch</code>
Set and read back output signals	<code>MIO_Digout</code> <code>MIO_Digout_Long</code> <code>MIO_Get_Digout_Long</code>
Transistor Outputs	
Set and reset output signals	<code>MIO_Digout</code> , <code>MIO_Digout_Long</code> <code>MIO_Get_Digout_Long</code>
Use latch register	<code>MIO_Dig_Latch</code> <code>MIO_Dig_Write_Latch</code>
Optocouple Inputs	
Query input signals	<code>MIO_Digin_Long</code>
Use latch register	<code>MIO_Dig_Latch</code> <code>MIO_Dig_Read_Latch</code>
Counter Block	
Configure counter	<code>Cnt_Enable</code> , <code>Cnt_Mode</code>
Use counter	<code>Cnt_Clear</code> , <code>Cnt_Get_Status</code> <code>Cnt_Latch</code> , <code>Cnt_Read</code> <code>Cnt_Read_Int_Register</code> <code>Cnt_Sync_Latch</code> , <code>Cnt_Read_Latch</code>
Use PWM counter	<code>Cnt_PW_Enable</code> , <code>Cnt_PW_Latch</code> <code>Cnt_Get_PW_HL</code> <code>P2_Cnt_Read_Int_Register</code>
SSI Decoder	
Use SSI decoder	<code>SSI_Mode</code> , <code>SSI_Set_Bits</code> <code>SSI_Set_Clock</code> <code>SSI_Set_Delay</code> , <code>SSI_Read</code> <code>SSI_Start</code> , <code>SSI_Status</code>
EtherCat Interface	
Use interface	<code>MIO_ECAT_Init</code> <code>MIO_ECAT_Get_Driver_Version</code> <code>MIO_ECAT_Check_State_Transition</code> <code>MIO_ECAT_Read_Data_16L</code> <code>MIO_ECAT_Write_Data_16L</code>
General	
Use LED	<code>Check_LED</code> , <code>Set_LED</code>



Range	Instructions
Use interrupt and event inputs	<code>Event_Enable, Event_Read</code> <code>Event_Config, Trigger_Event</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	P2_TDrv_Init P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block P2_Get_TiCo_RingBuffer, P2_Set_TiCo_RingBuffer P2_RingBuffer_Empty P2_RingBuffer_Full
Control <i>TiCo</i> processor	P2_TiCo_Reset, P2_TiCo_Start, P2_TiCo_Stop P2_Get_TiCo_Bootloader_Status P2_Get_TiCo_Status, P2_Workload
Control <i>TiCo</i> processes	P2_Process_Status P2_TiCo_Get_Processdelay P2_TiCo_Set_Processdelay P2_TiCo_Start_Process P2_TiCo_Stop_Process
Transfer <i>TiCo</i> programs	P2_TiCo_Flash, P2_TiCo_Load

Programming *TiCo* access

5.4.3 Pro II-MIO-D12 Rev. E

The module Pro II-MIO-D12 Rev. E is equipped with the following hardware:

- 12 transistor outputs (TRA)
- 12 optically isolated inputs (OPT)
- 1 Event input (see OPT inputs)
- 2 counter blocks with two 32-bit counters:
 - one up/down counter with clock/direction or four edge evaluation for connection of encoders.
 - one PWM counter to evaluate high and low times, duty cycle, or frequency.
- 1 SSI decoder to connect an incremental encoder
- *TiCo* processor with 56 KiByte internal memory

If the *TiCo* bootloader is programmed, the module can work on its own and independently from the CPU module of the *ADwin-Pro II* system.

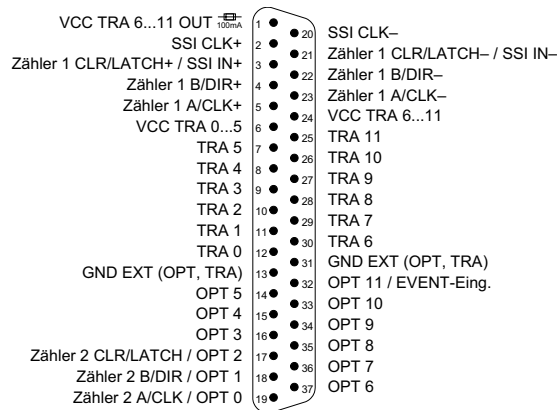


Fig. 30 – Pro II-MIO-D12 Rev. E: Pin assignment

Some pins have double functionality; all connected functions can be used at the same time, even though it may not be useful.

The module functions are described in the following sections:

- [Transistor Outputs \(TRA\)](#)
- [Optocouple Inputs \(OPT\)](#)
- [SSI decoder](#)
- [Counter block](#)
- [TiCo processor](#)
- [Technical Specification](#)
- Programmierung mit *ADbasic* and *TiCoBasic*

Transistor Outputs

The module Pro II-MIO-D12 Rev. E provides 12 channels of galvanically isolated transistor outputs. The outputs switch to V_{CC} . All TRA and OPT channels use a common GND channel.

The switching voltage V_{CC} has to be provided by an external power supply. You can supply 2 voltages one for every 6 switching outputs (TRA 0 ... TRA 5 and TRA 6 ... TRA 11).

The channels as well as the event-input are optically isolated from system circuitry.

The switching voltage of the outputs TRA 6 ... TRA 11 is also present on the pin VCC TRA 6...11 OUT to supply external devices. This output is protected with a 100mA fuse.

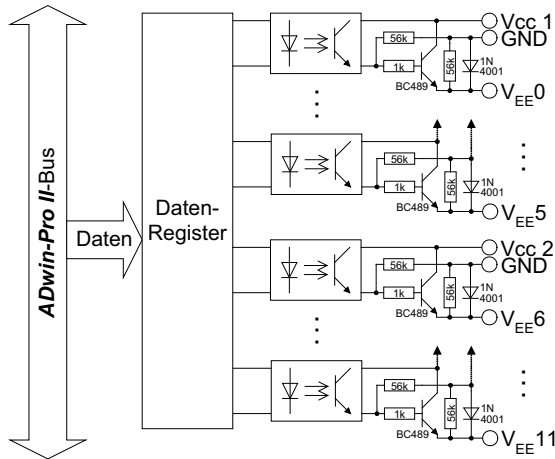


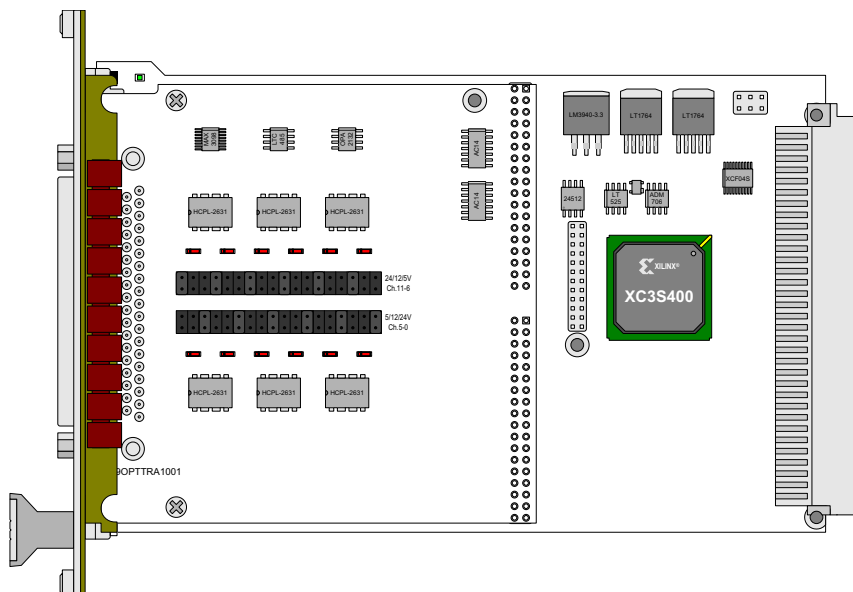
Fig. 31 – Pro II-MIO-D12 Rev. E: Block diagram transistor outputs

The module may output levels at defined points in time to TRA outputs as stand-alone. A FIFO serves as buffer where the user-defined level patterns and points in time are stored, maximum 511 Wertepaare. The output time can be defined with an accuracy of 5ns.

The FIFO can be used either for edge detection on the OPT inputs or for level output on TRA outputs.

Optocouple Inputs

The module Pro II-MIO-D12 Rev. E provides 12 channels of optically isolated digital inputs. The input voltage range can be set by jumpers (5V, 12V, 24V). The switching time of only 100ns allows the sampling of high-speed digital inputs.



The module can automatically monitor the edges of input channels. With every change, the current input levels are saved together with a time stamp in a FIFO. The FIFO data can be read and processed.

The FIFO can be used either for edge detection on the OPT inputs or for level output on TRA outputs.

Each channel is optically isolated from the system circuitry and from the other inputs. All TRA and OPT channels use a common GND channel.

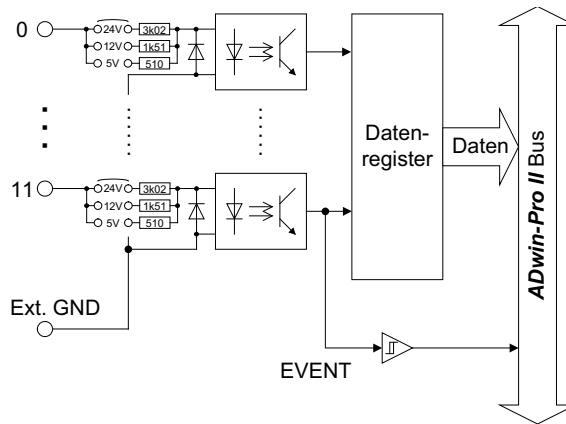


Fig. 32 – Pro II-MIO-D12 Rev. E: Block diagram optocoupler inputs

The inputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see [fig. 30](#).

The optically isolated input OPT 11 can also be used as event input. If the event input is enabled a trigger signal at the event input starts the externally triggered *ADbasic* process.

The inputs OPT 0...OPT 2 can also be used as inputs of counter 2. It is technically permissible, to use both functions (optical input and timer) at the same time.

SSI decoder

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

You can set the clock rate (6kHz ... 12.5MHz) as well as the data resolution (up to 32 bit) of the decoder via software.

Pin assignment of the decoder inputs see [fig. 30](#). The input SSI IN+/- can also be used as CLR/LATCH input of counter 1. It is technically permissible, to use both functions (SSI decoder and timer) at the same time.

A decoder either reads out an individual value (on request) or continuously provides the current value.

The SSI decoder can be set via software:

- clock rate can be set to 6,1kHz...12,5MHz with **SSI_Set_Clock**.
- Resolution can be set up to 32 bit with **SSI_Set_Bits**.

Counter block

The module Pro II-MIO-D12 Rev. E provides 2 configurable multi-purpose counter blocks. Each counter block contains two 32-bit counters: First an up/down counter or four edge evaluation for connection of encoders. Second, a PWM counter to evaluate high and low times, duty cycle, or frequency. Both counters of a block can be operated in parallel.

Counter 1 inputs run differential, counter 2 has optically isolated inputs (single ended).

Please note, that the pins of the counter inputs have double functionality. It is technically permissible, to use both functions (timer and other functions) at the same time, though it may not be useful.

The up/down counter of a block can be operated in 2 modes:

- Clock / direction (CLK and DIR signals)

A negative edge at the CLK input is the counting impulse for the 32-bit counter. The DIR signal sets the counting direction, TTL high means a count-up, TTL low means a count-down.

Via software, you can invert the signals at the inputs CLK and DIR (see **P2_Cnt_Mode**) and thus change the triggering edge as well as the count direction.

You can latch the counter values program-controlled or you can influence the counter by an external CLR/LATCH signal.

Depending on the programming the CLR/LATCH signal has either the effect that the counter values are cleared (CLR) or that the counter values are latched (LATCH). This function will only be effective when it is released by **P2_Cnt_Clear_Enable** or **P2_Cnt_Latch_Enable**.

The counter is cleared or latched with a rising edge at input CLR/LATCH. During the latch process the frequency of the measurement can be determined by getting the difference of two read latch values, because this difference defines the number of pulses between the two reading processes.

- Four edge evaluation (A and B signals)

The four edge evaluation changes the signals (which should be 90° phase-shifted) of a connected incremental encoder at the inputs A and B to CLK and DIR signals. For this you have to program the inputs correspondingly (see "ADwin-Pro System Description, Programming in ADbasic").

Since every edge of the A and B signals generates a count impulse, the resolution is increased by factor 4. If the encoder has a reference signal, it can be used to clear or latch the counter (after release of the CLR or LATCH input). The counter is cleared when the signals A, B and CLR are on logic "1" (software-selectable: clear, when only the CLR signal is on logic "1").

The PWM counter of the counter block analyzes the signals at the PWM inputs. With **P2_Cnt_Mode**, you set the input pin (A/CLK, B/DIR or CLR/LATCH) and the triggering edge.

Via software instructions the following data can be read directly:

- frequency and duty cycle (with **P2_Cnt_Get_PW**)
- high and low time (with **P2_Cnt_Get_PW_HL**)

The counter inputs are provided on the 25-pole D-Sub connector Conn2; pin assignment see [fig. 30](#).

If PWM counters are evaluated with standard instructions, no further knowledge is required.

If you require other evaluation methods, you can use several registers assigned to each PWM counter to create a solution. You find a description of PWM registers on [page 140](#) (module Pro II-CNT-x Rev. E). Use the evaluation with PWM registers for special solutions only.

Up/down counter

PWM Counter

TiCo processor

The module provides the freely programmable *TiCo* processor with 28kiB program memory, and 28KiB data memory. The internal memory serves as data and program memory. You program the *TiCo* processor with *TiCoBasic*.

The *TiCo* processor has access to all analog and digital input and output channels, as does the *ADwin* CPU. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Technical Specification

Transistor Outputs	
Output channels	12
Switching voltage V_{CC}	5...30V DC durch externe Spannungsversorgung
Switching current	200mA max. per channel
Voltage drop	0.5V
Switching time	2.5µs
Isolation	42V channel-channel / channel-GND common ground for all OPT and TRA channels
Optocouple Inputs	
Input channels	12
Input current	typ. 3.5mA / max. 7.5mA
Input voltage range (selectable via jumpers)	0...5V / 0...12V / 0...24V
Switching threshold for 0-low	0...0.8V / 0...1.6V / 0...3.2V
Switching threshold for 1-high	4.5...5V / 10...12V / 20...24V
Input over-voltage	-5V ... 8V / -5V ... 16V / -5V ... 30V /
Switching time	100ns
Isolation	42V channel-channel / channel-GND common ground for all OPT and TRA channels
Counter	
Number	2 universal counter blocks
Counter resolution	32 bit
Reference clock	50MHz
Clock frequency four edge evaluation	12.5MHz max. (at 90° phase-shift of the signals)
Clock frequency up/down counter	15MHz max.
Reference frequency PWM analysis	100MHz
Input / output level	1 counter compatible to RS422/485 (5V differential, 120 Ω bus terminating resistor) 1 counter via optocouples

Fig. 33 – Pro II-MIO-D12 Rev. E: Specification

Isolation	optocouple inputs 42V channel-channel / channel-GND
SSI Decoder	
Number	1
Clock frequency SSI decoder (CLK)	6kHz ... 12.5MHz
General	
TiCo processor	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal
Connectors	37-pin D-Sub female connector

Fig. 33 – Pro II-MIO-D12 Rev. E: Specification

Programmierung

The module is comfortably programmed with *ADbasic*. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Range	Instructions
Transistor Outputs	
Set and reset output signals	<code>P2_MIO_Digout</code> , <code>P2_MIO_Digout_Long</code> <code>P2_MIO_Get_Digout_Long</code>
Use latch register	<code>P2_MIO_Dig_Latch</code> , <code>P2_Sync_All</code> <code>P2_MIO_Dig_Write_Latch</code>
Set output levels automatically	<code>P2_Dig_FIFO_Mode</code> <code>P2_Digout_FIFO_Clear</code> <code>P2_Digout_FIFO_Empty</code> <code>P2_Digout_FIFO_Enable</code> <code>P2_Digout_FIFO_Read_Timer</code> <code>P2_Digout_FIFO_Start</code> <code>P2_Digout_FIFO_Write</code>
Optocouple Inputs	
Query input signals	<code>P2_Digin_Edge</code> <code>P2_MIO_Digin_Long</code>
Use latch register	<code>P2_MIO_Dig_Latch</code> , <code>P2_Sync_All</code> <code>P2_MIO_Dig_Read_Latch</code>
Control edges at input channels	<code>P2_Dig_FIFO_Mode</code> <code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Counter Block	
Configure counter	<code>P2_Cnt_Enable</code> , <code>P2_Cnt_Mode</code>

Programming in ADbasic

Programming in
TiCoBasic

Range	Instructions
Use counter	P2_Cnt_Clear, P2_Cnt_Get_Status P2_Cnt_Latch, P2_Cnt_Sync_Latch P2_Cnt_Read, P2_Cnt_Read4 P2_Cnt_Read_Latch P2_Cnt_Read_Latch4
Use PWM counter	P2_Cnt_PW_Enable, P2_Cnt_PW_Latch P2_Cnt_Get_PW, P2_Cnt_Get_PW_HL, P2_Cnt_Read_Int_Register

SSI Decoder

Use SSI decoder	P2_SSI_Mode, P2_SSI_Set_Bits P2_SSI_Set_Clock P2_SSI_Set_Delay, P2_SSI_Read P2_SSI_Start, P2_SSI_Status
-----------------	--

General

Use LED	P2_Check_LED, P2_Set_LED
Synchronize	P2_Sync_All, P2_Sync_Enable P2_Sync_Stat
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config

The module can be programmed with *TiCoBasic*. The instructions are described in *TiCoBasic* online help.

The include file MIO-D12_TiCo.inc contains instructions for the functions:

Range	Instructions
Transistor Outputs	
Set and reset output signals	MIO_Digout, MIO_Digout_Long MIO_Get_Digout_Long
Use latch register	MIO_Dig_Latch MIO_Dig_Write_Latch
Optocouple Inputs	
Query input signals	MIO_Digin_Long
Use latch register	MIO_Dig_Latch MIO_Dig_Read_Latch
Counter Block	
Configure counter	Cnt_Enable, Cnt_Mode
Use counter	Cnt_Clear, Cnt_Get_Status Cnt_Latch, Cnt_Read Cnt_Read_Int_Register Cnt_Sync_Latch, Cnt_Read_Latch
Use PWM counter	Cnt_PW_Enable, Cnt_PW_Latch Cnt_Get_PW_HL
	Cnt_PW_Enable, Cnt_PW_Latch Cnt_Get_PW_HL P2_Cnt_Read_Int_Register
SSI Decoder	
Use SSI decoder	SSI_Mode, SSI_Set_Bits SSI_Set_Clock SSI_Set_Delay, SSI_Read SSI_Start, SSI_Status

Range	Instructions
General	
Use LED	<code>Check_LED</code> , <code>Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable</code> , <code>Event_Read</code> <code>Event_Config</code> , <code>Trigger_Event</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

Programming *TiCo* access

5.5 Pro II: Analog Input Modules

This section describes analog input modules for *ADwin-Pro II*.

Analog input modules for *ADwin-Pro I* be found in the manual "ADwin-Pro Hardware" on page 18 ff.

Module name	AIn 8/18	AIn 8/18-TiCo	AIn 32/18-D	AIn 32/18-D-TiCo	AIn 16/18-C	AIn 8/18-8B	AIn 16/18-8B
Revision	E	E	E	E	E	E	E
Number ADC	1	1	1	1	1	1	1
Resolution [Bit]	18	18	18	18	18	18	18
Max. conv. time [µs]	2	2	2	2	2	2	2
max. sampl. rate [ksample/s]	500	500	500	500	500	500	500
Channels diff.	8	8	16	16	16	8	–
Channels sng. end.	–	–	32	32	–	–	–
Channels 8B	–	–	–	–	–	8	16
Meas. range	±10V				±20mA	±10V	
Gain	1, 2, 4, 8						
Calibration	by software						
TiCo processor	–	TiCo1	–	TiCo1	–	–	–
page	53	53	57	57	61	64	67

Module name	AIn F-4/14	AIn F-8/14	AIn F-4/16	AIn F-8/16	AIn F-4/18	AIn F-8/18
Revision	E	E	E	E	E	E
Number ADC	4	8	4	8	4	8
Resolution [Bit]	14	14	16	16	18	18
Max. conv. time [µs]	0.02	0.02	0.25	0.25	2	2
max. sampl. rate [ksample/s]	50000	4×50000 8×25000	4000	4000	500	500
Channels diff.	4	8	4	8	4	8
Meas. range	±10V					
Gain	1		1, 2, 4, 8		1	
Calibration	by software					
page	70	74	78	82	86	89

Note for open-ended inputs



Open-ended inputs can cause errors - above all in an environment where interferences may occur. You can avoid open-ended inputs this way:

- Separate unused inputs from open-ended lines.
- Apply a specified level (for instance GND) to unused inputs. Make the connection as close to the connector as possible.

5.5.1 Pro II-AIn-8/18 Rev. E

Analog input module Pro II-AIn-8/18 Rev. E with an 18-bit ADC, 8 differential inputs and a programmable amplifier (PGA). The module can be combined with amplifiers, Pro-TC and Pro-PT modules.

The module can be ordered in the following variants:

	no filter	filter 5kHz	filter 50kHz	filter 10kHz, range $\pm 30V$
LEMO 1-pole	AIn-8/18	AIn-8/18-LP5	–	AIn-8/18-LP-30V
LEMO 1-pole, TiCo processor	–	–	AIn-8/18-LP50-TiCo	AIn-8/18-LP-30V-TiCo
D-Sub	AIn-8/18-D	AIn-8/18-LP5-D	–	AIn-8/18-LP-30V-D
D-Sub, TiCo processor	–	–	AIn-8/18-LP50-D-TiCo	AIn-8/18-LP-30V-D-TiCo

The variants Pro II-AIn-8/18-xxx-TiCo additionally provide a freely programmable *TiCo* processor with 28KiByte data memory and 28KiByte program memory, which has access to all inputs of the module. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The variants Pro II-AIn-8/18-xxx-LP contain an input filter, a 4th grade low pass of Butterworth type. According to the variant the frequency is set to 5kHz, 10kHz, or 50kHz.

The inputs are available with shielded LEMO connectors (1-pole, CAMAC European norm) or a DSub socket 37-pin, see [fig. 36](#).

The module Pro II-AIn-8/18 Rev. E has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can control an upper and lower limit for each channel separately.

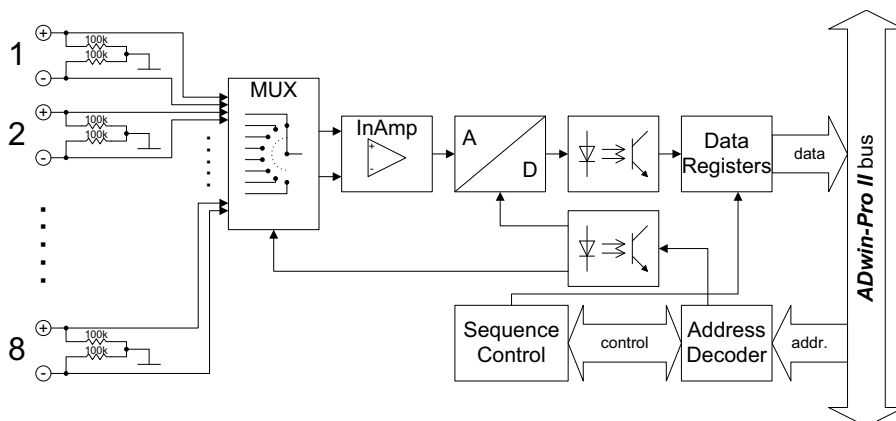


Fig. 34 – Pro II-AIn-8/18 Rev. E: Block diagram

TiCo processor

low pass filter

Input channels	8 differential via multiplexer	
Resolution	18 Bit	
Conversion time	max. 2 μ s	
Sampling rate	max. 500 ksps	
Multiplexer settling time	2.5 μ s	
Measurement range	± 10 V	
Gain	1, 2, 4, 8 software selectable	
Accuracy	INL	± 4 LSB typical
	DNL	max. ± 1 LSB
Input resistance	330k Ω , $\pm 2\%$	
Input over-voltage	± 35 V	
Offset error	adjustable	
Offset drift	± 30 ppm/ $^{\circ}$ C	
TiCo processor, only with Pro II-AIn-8/18-L2-TiCo Rev. E	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal	
Connector	8 LEMO female connectors, 1-pole or 37-polige Sub-D-Buchse	

Fig. 35 – Pro II-AIn-8/18 Rev. E: Specification

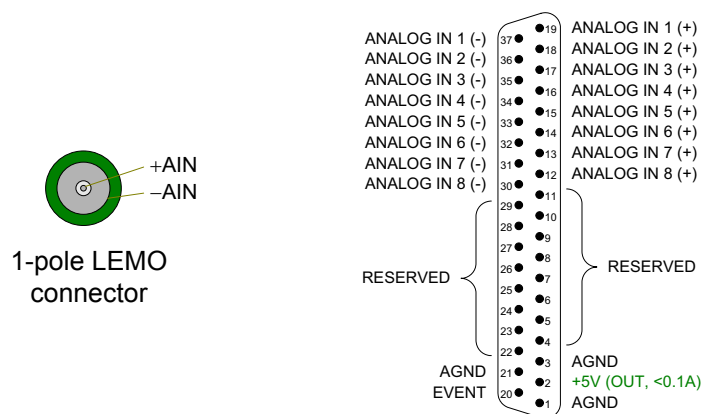


Fig. 36 – Pro II-AIn-8/18 Rev. E: Pin assignment

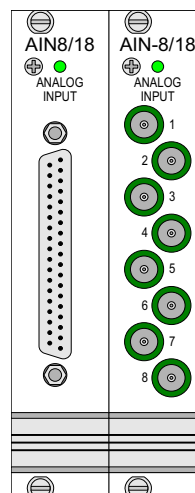


Fig. 37 – Pro II-AIn-8/18 Rev. E: Front panels

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_Seq_Read24_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Synchronize	<code>P2_Sync_All</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config, P2_Event2_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AInTiCo.inc` contains instructions for the following functions:

Function	Instruction
Do a single conversion – complete or step by step	<code>ADC, ADC24</code> <code>Set_Mux, Start_Conv, Wait_EOC</code> <code>Read_ADC, Read_ADC24</code>
Read value and start new conversion	<code>Read_ADC_SConv</code> <code>Read_ADC_SConv24</code>
Use LED	<code>Check_LED, Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable, Event_Read</code> <code>Event_Config, Trigger_Event</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>

Programming in ADbasic

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

5.5.2 Pro II-AIn-32/18-D Rev. E

Analog input module Pro II-AIn-32/18-D Rev. E with an 18-bit ADC, 32 differential inputs and a programmable amplifier (PGA). The module can be combined with amplifiers, Pro-TC and Pro-PT modules.

The variant Pro II-AIn-32/18-D-TiCo Rev. E additionally provides a freely programmable *TiCo* processor, which has access to all inputs of the module. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The module has 32 single-ended or 16 differential inputs (software selectable). The inputs are equipped with a 37-pin D-Sub female connector; for pin assignment see [fig. 40](#) and [41](#).

After power-up the module is set to 16 differential inputs.

The module Pro II-AIn-32/18-D Rev. E has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Ex works the inputs are connected to the ground of the Pro device. Alternatively a GND level signal—common for all inputs—can be connected to one of the AGND pins. The ground connection to the Pro device should be split up, by switching the DIL switch (see [fig. 42](#)) to position GND LIFT.

Do not run the module without ground connection.

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can control an upper and lower limit for each channel separately.

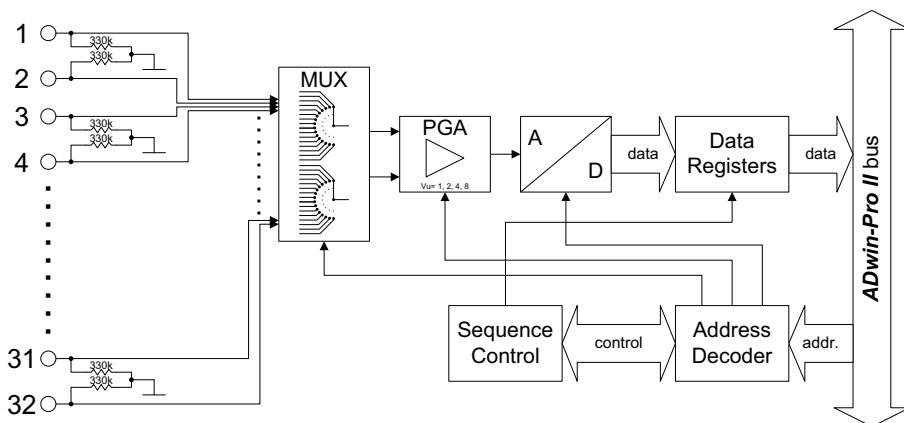


Fig. 38 – Pro II-AIn-32/18-D Rev. E: Block diagram

Input channels:	32 single ended or 16 differential; via multiplexer
Resolution:	18 bit
Conversion time:	max. 2 μ s
Sampling rate:	max. 500 ksp/s
Multiplexer settling time:	2.5 μ s
Measurement range:	$\pm 10V$

Fig. 39 – Pro II-AIn-32/18-D Rev. E: Specification



Gain:		1, 2, 4, 8 software selectable
Accuracy	INL	±4 LSB typical
	DNL	max. ±1 LSB
Input resistance:		330kΩ, ±2%
Input over-voltage:		±35 V
Offset error:		adjustable
Offset drift:		±30ppm/°C
Connector:		37-pin D-Sub female connector
TiCo processor, only with Pro II-AIn-8/18-L2-TiCo Rev. E		Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal

Fig. 39 – Pro II-AIn-32/18-D Rev. E: Specification

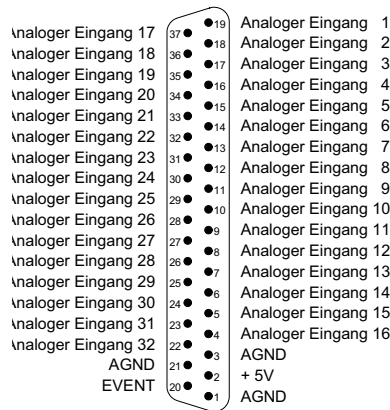


Fig. 40 – Pro II-AIn-32/18-D Rev. E: Pin assignment single ended

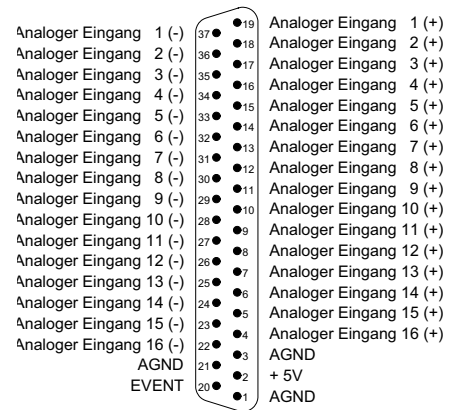


Fig. 41 – Pro II-AIn-32/18-D Rev. E: Pin assignment differential

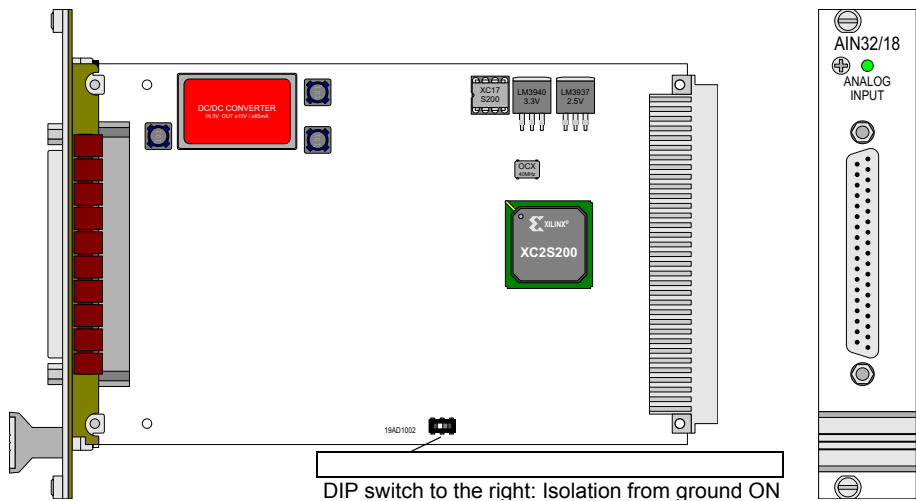


Fig. 42 – Pro II-AIn-32/18-D Rev. E: Board and front panel

Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_Seq_Read24_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Synchronize	<code>P2_Sync_All</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config, P2_Event2_Config</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start,</code> <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AInTiCo.inc` contains instructions for the following functions:

Function	Instruction
Set inputs to s.e. or differential	<code>SE_Diff</code>

Programming TiCo access

Programming in TiCoBasic

Function	Instruction
Do a single conversion – complete or step by step	<code>ADC, ADC24</code> <code>Set_Mux, Start_Conv, Wait_EOC</code> <code>Read_ADC, Read_ADC24</code>
Read value and start new conversion	<code>Read_ADC_SConv</code> <code>Read_ADC_SConv24</code>
Use LED	<code>Check_LED, Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable, Event_Read</code> <code>Event_Config, Trigger_Event</code>

5.5.3 Pro II-AIn-16/18-C Rev. E

Analog input module Pro II-AIn-16/18-C Rev. E with an 18-bit ADC, 16 differential inputs and a programmable amplifier (PGA). The module can be combined with amplifiers, Pro-TC and Pro-PT modules.

The module has 16 differential current inputs. The inputs are equipped with a 37-pin D-Sub female connector; for pin assignment see [fig. 45](#).

The module variant with voltage inputs Pro II-AIn-32/18-D Rev. E is described on [page 57](#).

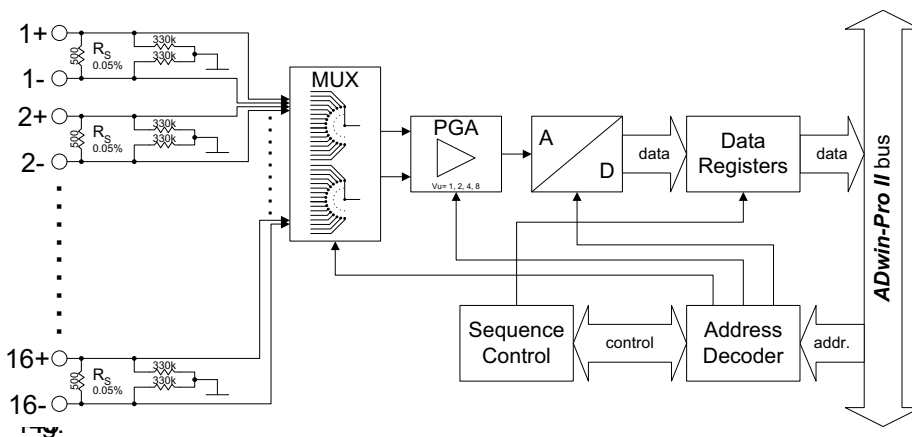
The module Pro II-AIn-16/18-C Rev. E has an input current range of $\pm 20\text{mA}$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Ex works the inputs are connected to the ground of the Pro device. Alternatively a GND level signal—common for all inputs—can be connected to one of the AGND pins. The ground connection to the Pro device should be split up, by switching the DIL switch (see [fig. 42](#)) to position GND LIFT.

Do not run the module without ground connection.

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can control an upper and lower limit for each channel separately.



Pro II-AIn-16/18-C Rev. E: Block diagram

Input channels:	16 differential via multiplexer	
Resolution:	18 bit	
Conversion time:	max. 2 μs	
Sampling rate:	max. 500 ksps	
Multiplexer settling time:	2.5 μs	
Measurement range:	$\pm 20\text{mA}$	
Gain:	1, 2, 4, 8 software selectable	
Accuracy	INL	± 4 LSB typical $\pm 0.05\%$ of measured voltage by input resistance
	DNL	max. ± 1 LSB $\pm 0.05\%$ of measured voltage by input resistance
Input resistance:	500 k Ω , $\pm 0.05\%$	
Input over-voltage:	$\pm 15\text{V}$	

Fig. 44 – Pro II-AIn-16/18-C Rev. E: Specification

Offset error:	adjustable
Offset drift:	±30ppm/°C
Connector:	37-pin D-Sub female connector

Fig. 44 – Pro II-AIn-16/18-C Rev. E: Specification

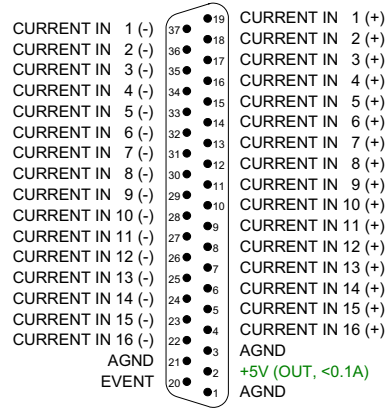


Fig. 45 – Pro II-AIn-16/18-C Rev. E: Pin assignment

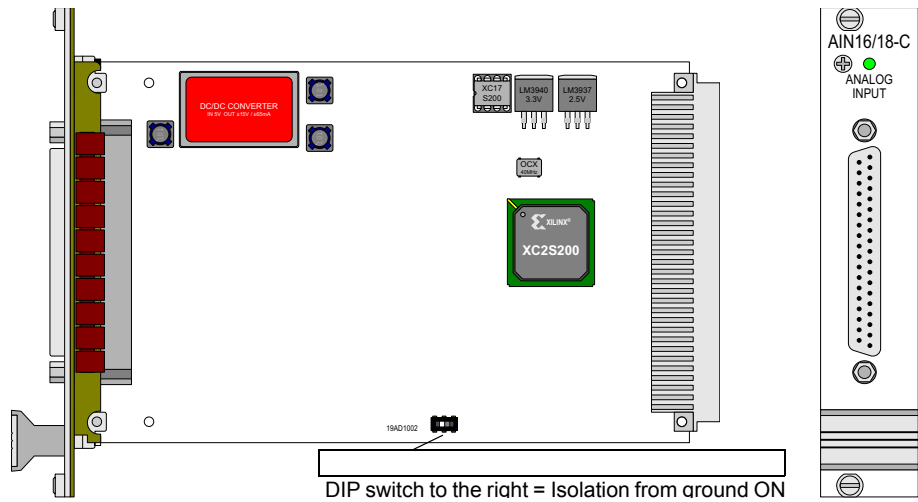


Fig. 46 – Pro II-AIn-16/18-C Rev. E: Board and front panel

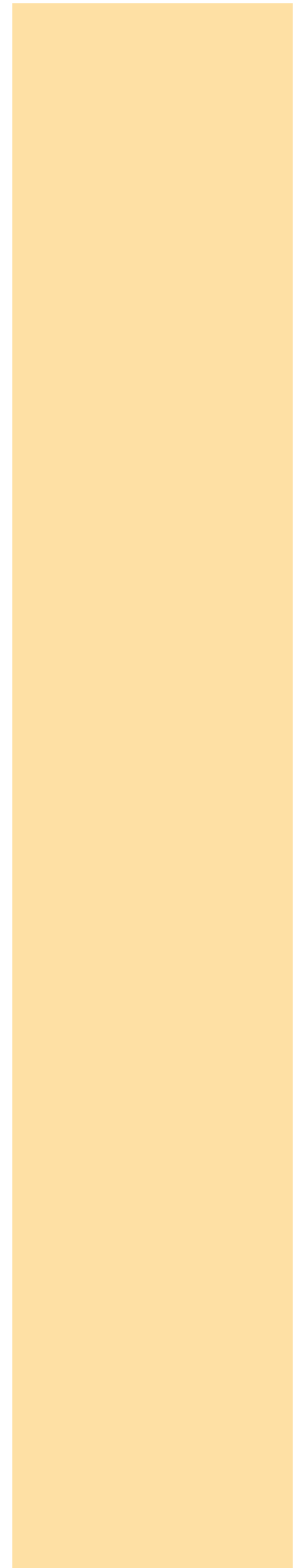
Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions, described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains the following instructions:

Function	Instructions
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_Seq_Read24_Packed</code> <code>P2_Seq_Wait</code>

Function	Instructions
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Synchronize	<code>P2_Sync_All</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code> , <code>P2_Event2_Config</code>



5.5.4 Pro II-AIn-8/18-8B Rev. E

Analog input module Pro II-AIn-8/18-8B Rev. E with an 18-bit ADC, 16 analog inputs and a programmable amplifier (PGA). The module is based upon [Pro II-AIn-32/18-D Rev. E](#) with an additional board holding 8 plug-in slots for 8B modules.

The 16 inputs are divided to 8 inputs for 8B modules and 8 differential inputs. The inputs are provided on two 37-pin D-Sub female connectors; for pin assignments see [fig. 48](#).

The input voltage range after 8B modules is $\pm 10V$. Gain is software selectable to 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Ex works the inputs are connected to the ground of the Pro device. Alternatively a GND level signal—common for all inputs—can be connected to one of the AGND pins. The ground connection to the Pro device should be split up, by switching the DIL switch (see [fig. 42](#)) to position GND LIFT.

Do not run the module without ground connection.

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can control an upper and lower limit for each channel separately.

Input channels	16 via multiplexer: 8 for 8B modules, 8 differential	
Resolution	18 bit	
Conversion time	max. 2 μs	
Sampling rate	max. 500 ksps	
Multiplexer settling time	2.5 μs	
Measurement range	$\pm 10V$	
Gain	1, 2, 4, 8 software selectable	
Accuracy	INL	± 4 LSB typical
	DNL	max. ± 1 LSB
Input resistance	330k Ω , $\pm 2\%$	
Input over-voltage	$\pm 35V$	
Offset error	adjustable	
Offset drift	± 30 ppm/ $^{\circ}C$	
Connector	37-pin D-Sub female connector	
Module width	15 HP	

Fig. 47 – Pro II-AIn-8/18-8B Rev. E: Specification

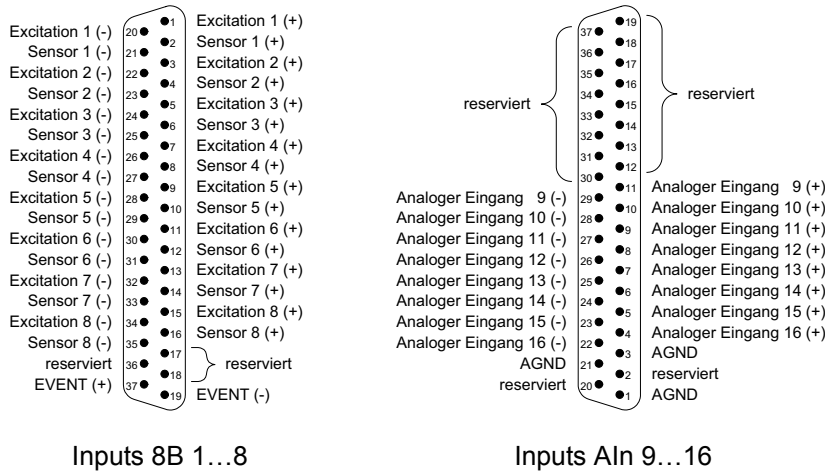


Fig. 48 – Pro II-AIn-8/18-8B Rev. E: Pin assignments

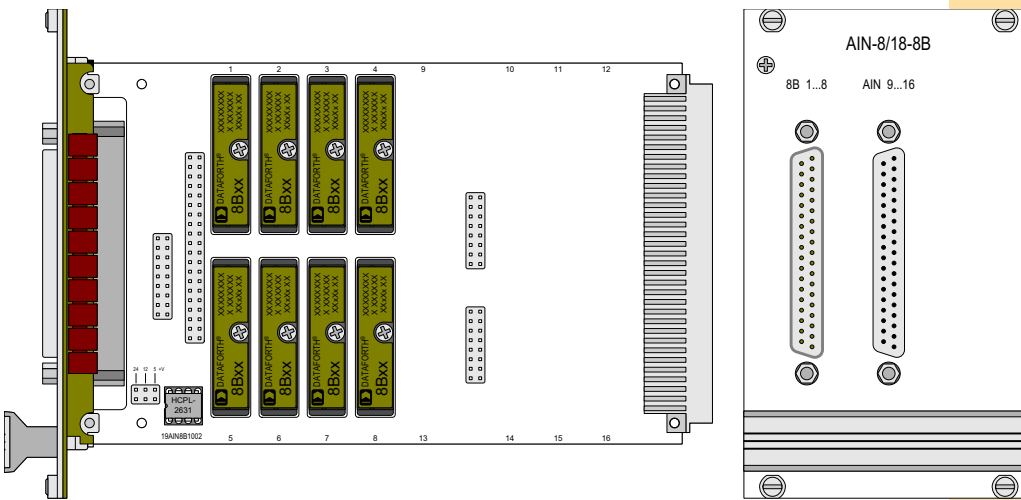


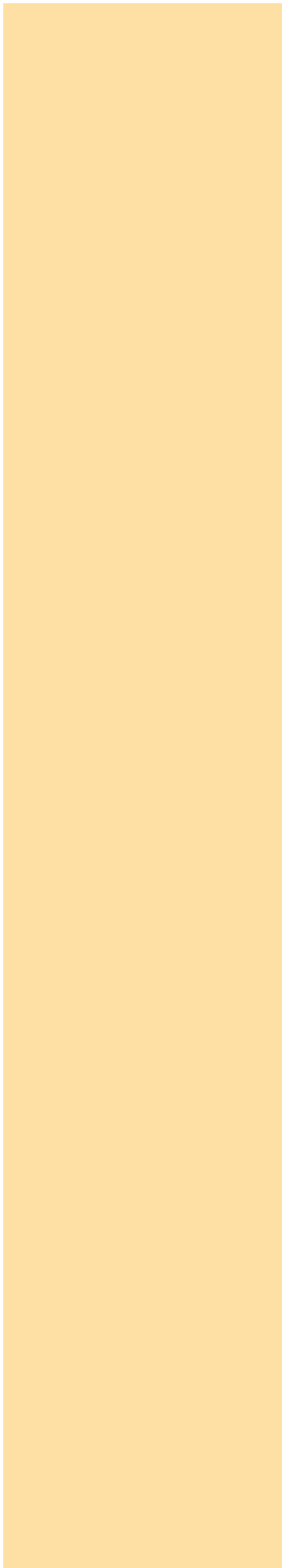
Fig. 49 – Pro II-AIn-8/18-8B Rev. E: Board and front panel

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Do a single conversion – complete or step by step	<code>P2_ADC, P2_ADC24</code> <code>P2_Set_Mux, P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC, P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init, P2_Seq_Start</code> <code>P2_Seq_Read, P2_Seq_Read24</code> <code>P2_Seq_Read24_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>
Synchronize	<code>P2_Sync_All</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>

Programming



Function	Instructions
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code> , <code>P2_Event2_Config</code>

5.5.5 Pro II-AIn-16/18-8B Rev. E

Analog input module Pro II-AIn-16/18-8B Rev. E with an 18-bit ADC, 16 analog inputs and a programmable amplifier (PGA). The module is based upon [Pro II-AIn-32/18-D Rev. E](#) with an additional board holding 16 plug-in slots for 8B modules.

The module has 16 inputs (via 8B modules) provided on two 37-pin D-Sub female connectors; for pin assignments see [fig. 51](#).

The input voltage range after 8B modules is $\pm 10V$. Gain is software selectable to 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Ex works the inputs are connected to the ground of the Pro device. Alternatively a GND level signal—common for all inputs—can be connected to one of the AGND pins. The ground connection to the Pro device should be split up, by switching the DIL switch (see [fig. 42](#)) to position GND LIFT.

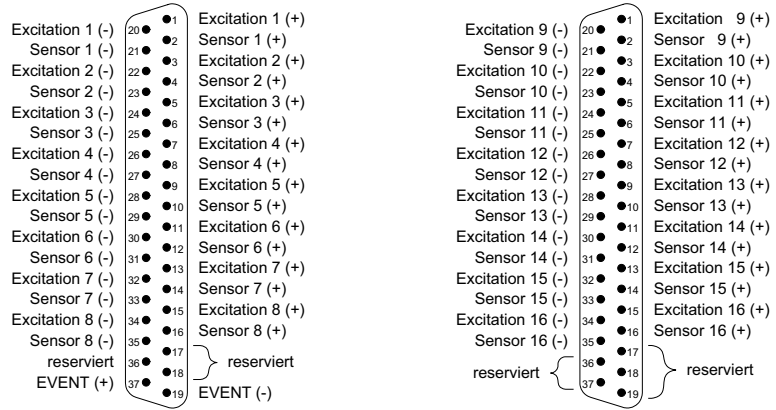
Do not run the module without ground connection.

The module includes a sequential control, which can read measurement values from several or all input channels sequentially.

The module can control an upper and lower limit for each channel separately.

Input channels	16 via multiplexer for 8B modules.	
Resolution	18 bit	
Conversion time	max. 2 μ s	
Sampling rate	max. 500ksps	
Multiplexer settling time	2.5 μ s	
Measurement range	$\pm 10V$	
Gain	1, 2, 4, 8 software selectable	
Accuracy	INL	± 4 LSB typical
	DNL	max. ± 1 LSB
Input resistance	330k Ω , $\pm 2\%$	
Input over-voltage	$\pm 35V$	
Offset error	adjustable	
Offset drift	± 30 ppm/ $^{\circ}C$	
Connector	37-pin D-Sub female connector	
Module width	15 HP	

Fig. 50 – Pro II-AIn-16/18-8B Rev. E: Specification



Eingänge 8B 1...8

Eingänge 8B 9...16

Fig. 51 – Pro II-AIn-16/18-8B Rev. E: Pin assignments

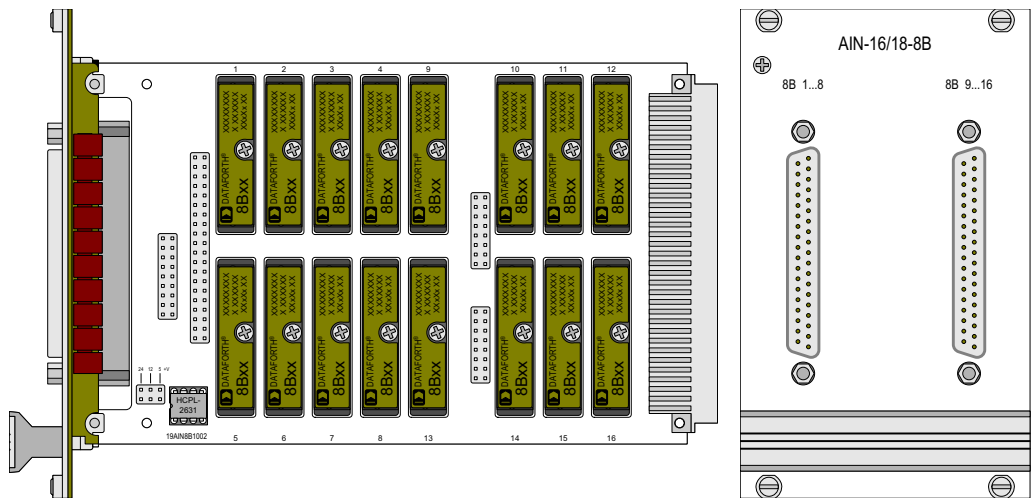


Fig. 52 – Pro II-AIn-16/18-8B Rev. E: Board and front panel

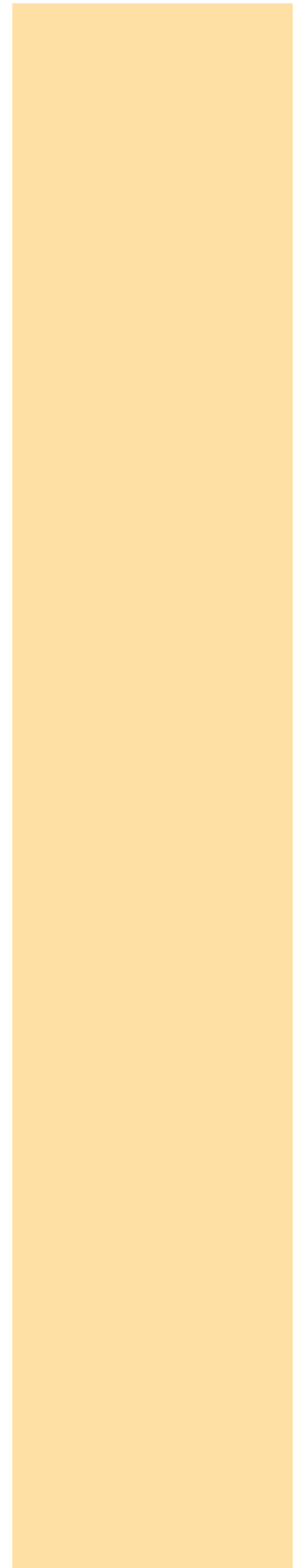
Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Set inputs to single-ended or differential	<code>P2_SE_Diff</code>
Do a single conversion – complete or step by step	<code>P2_ADC</code> , <code>P2_ADC24</code> <code>P2_Set_Mux</code> , <code>P2_Start_Conv</code> <code>P2_Wait_EOC</code> <code>P2_Read_ADC</code> , <code>P2_Read_ADC24</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code>
Use sequence control	<code>P2_Seq_Init</code> , <code>P2_Seq_Start</code> <code>P2_Seq_Read</code> , <code>P2_Seq_Read24</code> <code>P2_Seq_Read24_Packed</code> <code>P2_Seq_Wait</code>
Control input limits	<code>P2_ADC_Read_Limit</code> <code>P2_ADC_Set_Limit</code>

Function	Instructions
Synchronize	<code>P2_Sync_All</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config, P2_Event2_Config</code>



5.5.6 Pro II-AIn-F-4/14 Rev. E

The analog input module Pro II-AIn-F-4/14 Rev. E has 4 Fast-ADC of 14 Bit and 4 differential inputs.

The inputs are available with the following connectors:

- Pro II-AIn-F-4/14: shielded LEMO connectors, 1-pole, CAMAC European norm.
- Pro II-AIn-F-4/14-L2: shielded LEMO connectors, 2-pole, CAMAC European norm.
- Pro II-AIn-F-4/14-D: D-Sub female connector 37-pin.
- Pro II-AIn-F-4/14-B: BNC female connectors.

The module's converters run stand-alone with a fixed clock rate of 50MHz; the program fetches only the recently converted measurement values. The module memory enables to buffer the great data volume of burst sequences.

The module Pro II-AIn-F-4/14 Rev. E has an input voltage range of $\pm 10V$. The adjustment of gain and offset is made by software. (see [chapter 6 "Calibration"](#)).

The module provides several operation modes for converting analog signals:

- Single measurement: The module converts stand-alone (with fixed clock rate). The *ADbasic* program queries the current measurement value and processes them.
- Single burst sequence: The *ADbasic* program starts a complete burst measurement sequence, that is a defined number of single measurements.
- Continuous burst sequence: The *ADbasic* program starts a burst measurement sequence, that continuously does single measurements until the sequence is stopped. The data is stored in a fifo-like memory.

As an option, the module can return the moving average of 2...32 measurement values instead of simple measurement values. For each average value, the module separately samples the number of set measurement values.

The module can control an upper and lower limit for each channel separately.

The module processes burst sequences independent from the processor module of the *ADwin* system. The measurement values—number and measuring rate to be defined ahead in the program—are stored in the burst-memory of the module. The processor module then reads the stored values (even during a burst sequence) and processes them.

With a continuous burst-sequence, the measuring rate must be synchronized to the reading rate. Please note:

- Measurement values are always read in blocks, the block size is selectable. The greater the data blocks are, the faster the average reading is done.

Look out: While a block is read, other processes even with higher priority may be delayed. The probability of a delay rises with the block size.

- The time offset between continuous conversion and blockwise reading demands a data buffer. Therefore, the initialization of the burst sequence must allocate sufficient memory range (`P2_Burst_Init`, parameter `samples`).

With module version Pro II-AIn-F-4/14-D Rev. E (D-Sub female connector), a burst sequence can be controlled by external event signals, i.e. each (resulting) event signal has a measurement value stored.

Burst Sequence

Event Inputs

As an option one channel of the burst sequence may be used as time channel, which holds the counter value of the internal module timer for each event signal.

The module is equipped with 3 differential event inputs: EVENT/A, B, ENABLE, whose signals are processed to the resulting event signal. This pre-processing of signals be configured with **P2_Event2_Config**.

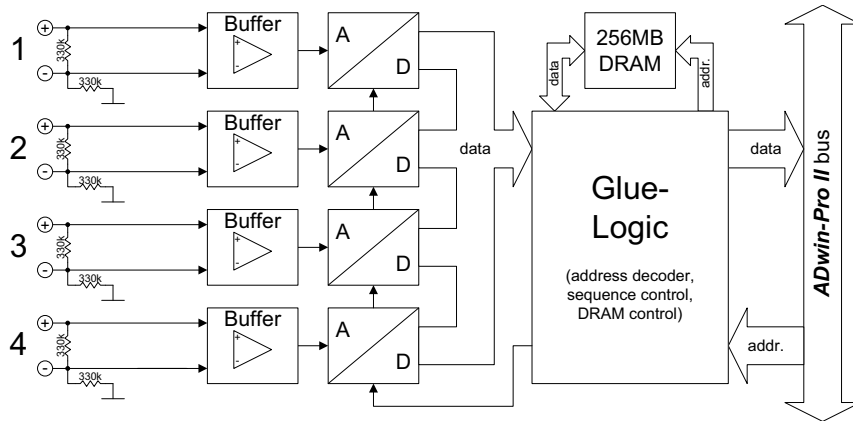


Fig. 53 – Pro II-Aln-F-4/14 Rev. E: Block diagram

Input channels	4 differential
Resolution	14 Bit
Conversion time	0.02 μ s (per ADC)
Sampling rate	50000ksps (per ADC)
Input band width	0 ... 4MHz
Memory size	256MiB or $2^{27} = 134217728$ values total
Measurement range	± 10 V with max. offset 3.5 V
Accuracy	INL typical ± 1.2 LSB, max. ± 5 LSB
	DNL typical ± 0.5 LSB, max. ± 1 LSB
Input resistance	330k Ω , $\pm 2\%$
Input over-voltage	± 35 V
Offset error	adjustable
Offset drift	± 30 ppm/ $^{\circ}$ C of full scale range
Event input (D-Sub only)	3 differential; RS422/485 compatible (5V differential, 120 Ω bus terminating resistor) max. signal frequency 16MHz
Connector	4 LEMO female connectors. 1-pole, or 4 LEMO female connectors. 2-pole, or 37-pin D-Sub female connector, or 4 BNC female connectors.

Fig. 54 – Pro II-Aln-F-4/14 Rev. E: Specification

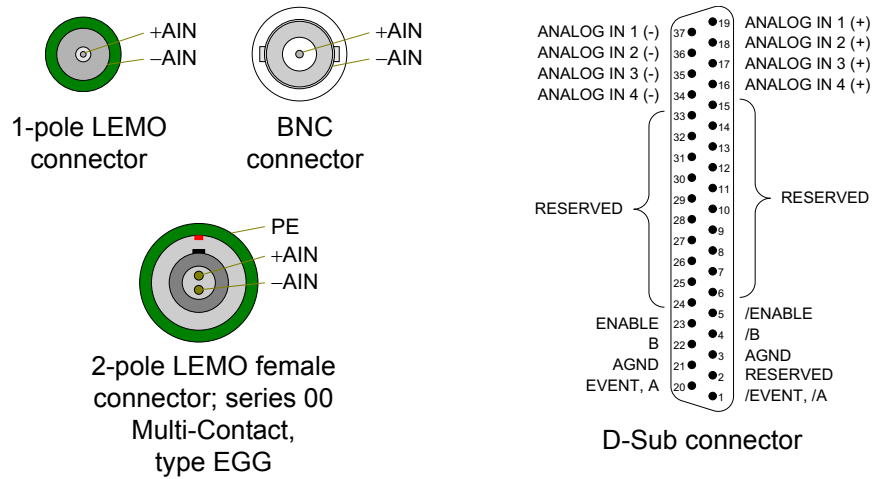


Fig. 55 – Pro-AIn-F-4/14-D Rev. E: Pin assignment differential

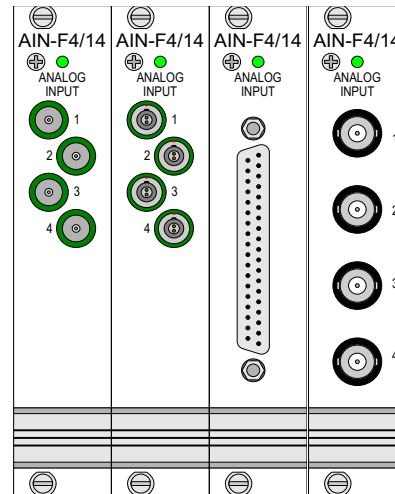


Fig. 56 – Pro II-AIn-F-4/14 Rev. E: Front panels

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode</code>
Do a single conversion – complete or step by step	<code>P2_ADCF</code> <code>P2_Start_Conv, P2_Wait_EOCF</code> <code>P2_Read_ADCF</code>
Read several values	<code>P2_Read_ADCF32, P2_Read_ADCF4</code> <code>P2_Read_ADCF4_Packed</code>

Function	Instructions
Use burst conversion	P2_Burst_Init P2_Burst_Start, P2_Burst_Stop P2_Burst_Status, P2_Burst_Reset P2_Burst_Read P2_Burst_Read_Index, P2_Burst_Read_Index P2_Burst_Read_Unpacked1/2/4 P2_Burst_CRead_Unpacked1/2/4 P2_Burst_CRead_Pos_Unpacked1/2/4
Synchronize	P2_Sync_All, P2_Sync_Enable P2_Sync_Stat
Control input limits	P2_ADCF_Read_Limit P2_ADCF_Set_Limit
Use LED	P2_Check_LED, P2_Set_LED
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config, P2_Event2_Config



5.5.7 Pro II-AIn-F-8/14 Rev. E

The analog input module Pro II-AIn-F-8/14 Rev. E has 8 Fast-ADC of 14 Bit and 8 differential inputs.

The inputs are available with the following connectors:

- Pro II-AIn-F-8/14-L2: shielded LEMO female connectors, 1-pole, CAMAC European norm.
- Pro II-AIn-F-8/14-L2: shielded LEMO female connectors, 2-pole, CAMAC European norm.
- Pro II-AIn-F-8/14-D: D-Sub female connector 37-pin.
- Pro II-AIn-F-8/14-B: BNC female connectors.

The module's converters run stand-alone with a fixed clock rate of 50MHz. If converting measurement values from all 8 channels, the sampling rate is limited to 25MHz due to the maximum memory access rate.

The module Pro II-AIn-F-8/14 Rev. E has an input voltage range of $\pm 10V$. The adjustment of gain and offset is made by software. (see [chapter 6 "Calibration"](#)).

The great data volume of burst sequences is buffered in the module memory.

The module provides several operation modes for converting analog signals:

- Single measurement: The module converts stand-alone (with fixed clock rate). The *ADbasic* program queries the current measurement value and processes them.
- Single burst sequence: The *ADbasic* program starts a complete burst measurement sequence, that is a defined number of single measurements.
- Continuous burst sequence: The *ADbasic* program starts a burst measurement sequence, that continuously does single measurements until the sequence is stopped. The data is stored in a fifo-like memory.

As an option, the module can return the moving average of 2...32 measurement values instead of simple measurement values. For each average value, the module separately samples the number of set measurement values.

The module can control an upper and lower limit for each channel separately.

The module processes burst sequences independent from the processor module of the *ADwin* system. The measurement values—number and measuring rate to be defined ahead in the program—are stored in the burst-memory of the module. The processor module then reads the stored values (even during a burst sequence) and processes them.

With a continuous burst-sequence, the measuring rate must be synchronized to the reading rate. Please note:

- Measurement values are always read in blocks, the block size is selectable. The greater the data blocks are, the faster the average reading is done.

Look out: While a block is read, other processes even with higher priority may be delayed. The probability of a delay rises with the block size.

- The time offset between continuous conversion and blockwise reading demands a data buffer. Therefore, the initialization of the burst sequence must allocate sufficient memory range (`P2_Burst_Init`, parameter `samples`).

Burst Sequence

With module version Pro II-AIn-F-8/14-D Rev. E (D-Sub female connector), a burst sequence can be controlled by external event signals, i.e. each (resulting) event signal has a measurement value stored.

As an option one channel of a burst sequence may be used as time channel, holding the counter value of the internal module timer for each event signal.

The module is equipped with 3 differential event inputs: EVENT/A, B, ENABLE, whose signals are processed to the resulting event signal. This pre-processing of signals be configured with **P2_Event2_Config**.

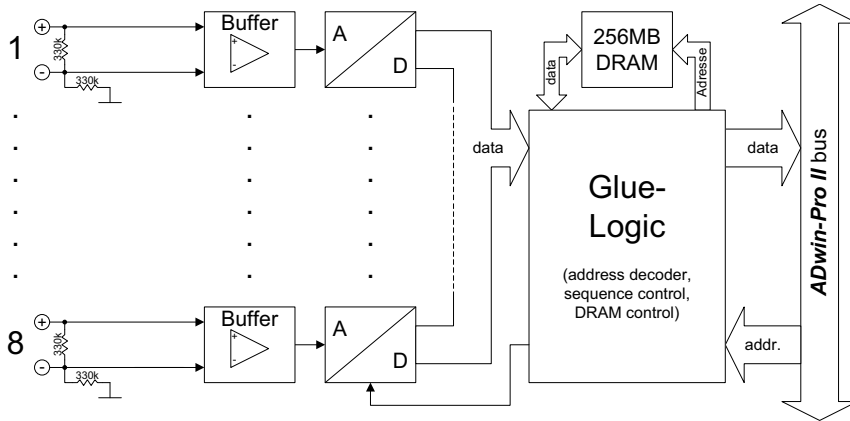


Fig. 57 – Pro II-AIn-F-8/14 Rev. E: Block diagram

Input channels	8 differential
Resolution	14 Bit
Conversion time	0.02 μ s (per ADC)
Sampling rate	4 \times 50000ksp/s (per ADC) or 8 \times 25000ksp/s (per ADC)
Input band width	0 ... 4MHz
Memory size	256 MiB or $2^{27} = 134217728$ values total
Measurement range	± 10 V with max. offset 3.5 V
Accuracy	INL typical ± 1.2 LSB, max. ± 5 LSB DNL typical ± 0.5 LSB, max. ± 1 LSB
Input resistance	330k Ω , $\pm 2\%$
Input over-voltage	± 35 V
Offset error	adjustable
Offset drift	± 30 ppm/ $^{\circ}$ C of full scale range
Event input (D-Sub only)	3 differential; RS422/485 compatible (5V differential, 120 Ω bus terminating resistor) max. signal frequency 16MHz
Connector	8 LEMO female connectors. 1-pole, or 8 LEMO female connectors. 2-pole, or 37-pin D-Sub female connector, or 8 BNC female connectors.
Module width	10 HP

Fig. 58 – Pro II-AIn-F-8/14 Rev. E: Specification

Event Inputs

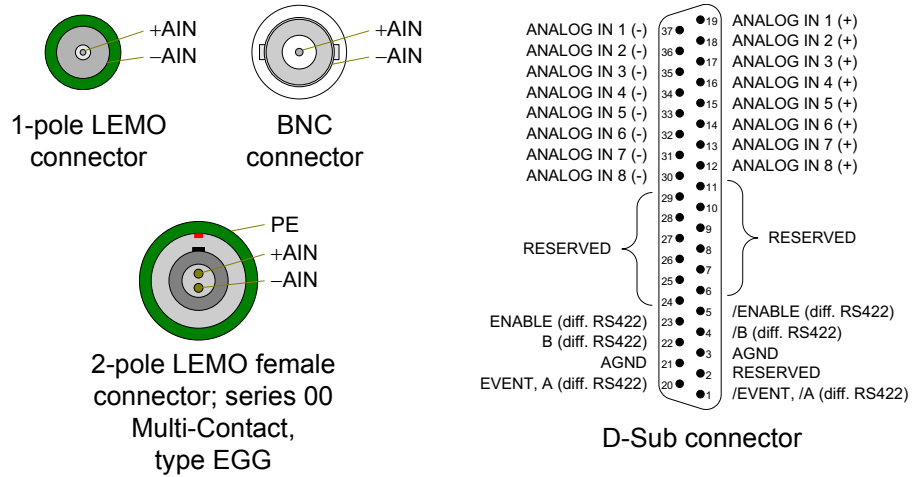


Fig. 59 – Pro-AIn-F-8/14-D Rev. E: Pin assignment differential

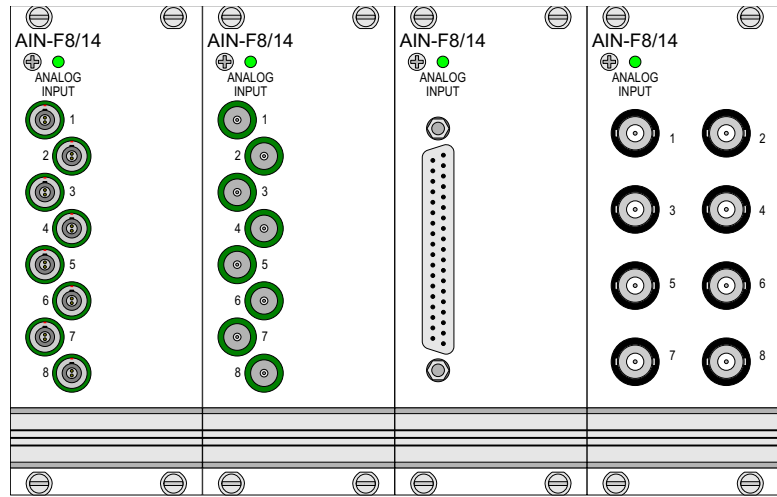


Fig. 60 – Pro II-AIn-F-8/14 Rev. E: Front panels

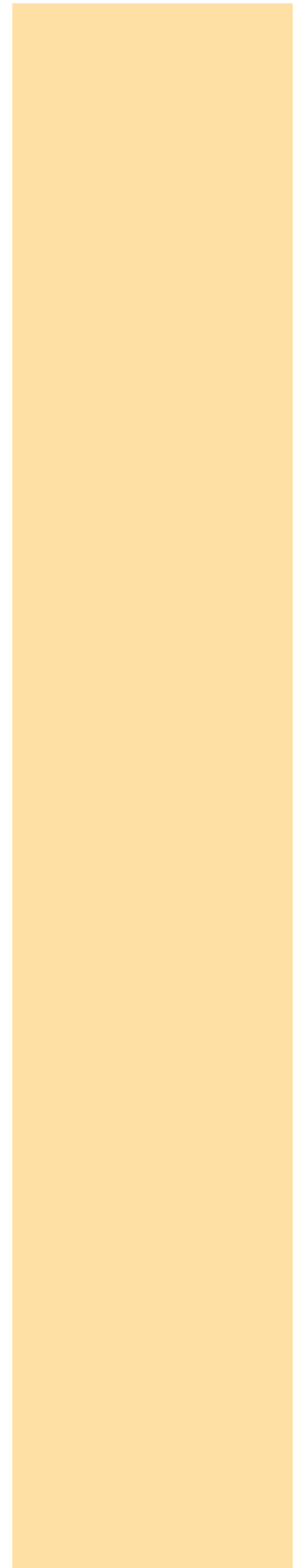
Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode</code>
Do a single conversion – complete or step by step	<code>P2_ADCF</code> <code>P2_Start_Conv, P2_Wait_EOCF</code> <code>P2_Read_ADCF</code>
Read several values	<code>P2_Read_ADCF32</code> <code>P2_Read_ADCF4, P2_Read_ADCF8</code> <code>P2_Read_ADCF4_Packed</code> <code>P2_Read_ADCF8_Packed</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv32</code>

Function	Instructions
Use burst conversion	P2_Burst_Init P2_Burst_Start, P2_Burst_Stop P2_Burst_Status, P2_Burst_Reset P2_Burst_Read P2_Burst_Read_Index, P2_Burst_Read_Index P2_Burst_Read_Unpacked1/2/4/8 P2_Burst_CRead_Unpacked1/2/4/8 P2_Burst_CRead_Pos_Unpacked1/2/4/8
Synchronize	P2_Sync_All, P2_Sync_Enable P2_Sync_Mode, P2_Sync_Stat
Control input limits	P2_ADCF_Read_Limit P2_ADCF_Set_Limit P2_ADCF_Reset_Min_Max, P2_ADCF_Read_Min_Max4
Use LED	P2_Check_LED, P2_Set_LED
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config, P2_Event2_Config



5.5.8 Pro II-AIn-F-4/16 Rev. E

The analog input module Pro II-AIn-F-4/16 Rev. E has 4 Fast-ADC of 16 Bit and 4 differential inputs.

The inputs are available with the following connectors:

- Pro II-AIn-F-4/16: shielded LEMO connectors, 1-pole, CAMAC European norm.
- Pro II-AIn-F-4/16-L2: shielded LEMO connectors, 2-pole, CAMAC European norm.
- Pro II-AIn-F-4/16-D: D-Sub female connector 37-pin.
- Pro II-AIn-F-4/16-B: BNC female connectors.

The module's converters run with a clock rate of up to 4MHz. Since revision E04, module memory enables to buffer data—especially with burst sequences.

The module Pro II-AIn-F-4/16 Rev. E has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Since revision E04, the module provides several operation modes for converting analog signals:

- Single measurement: You start each conversion manually, query the measurement value and processes it in the *ADbasic* program.
- Single burst sequence: The *ADbasic* program starts a complete burst measurement sequence, that is a defined number of single measurements.
- Continuous burst sequence: The *ADbasic* program starts a burst measurement sequence, that continuously does single measurements until the sequence is stopped. The data is stored in a fifo-like memory.

As an alternative to the operation modes above, the module can perform a simultaneous conversion on all channels using a sequence control. This disburdens—compared to single measurements with the instruction `P2_ADCF`—the processor module, which only reads and processes the converted measurement values. The conversion can be either triggered periodically with regular time delays or by external event signals.

As an option, the module can return the moving average of 2...32 measurement values instead of simple measurement values. For each average value, the module separately samples the number of set measurement values.

The module can control an upper and lower limit for each channel separately.

The module collects the maximum and minimum value of each channel and provides them via software instruction.

Burst Sequence

The module processes burst sequences independent from the processor module of the *ADwin* system. The measurement values—number and measuring rate to be defined ahead in the program—are stored in the burst-memory of the module. The processor module then reads the stored values (even during a burst sequence) and processes them.

Burst sequences are available since rev. E04.

With a continuous burst-sequence, the measuring rate must be syntonized to the reading rate. Please note:

- Measurement values are always read in blocks, the block size is selectable. The greater the data blocks are, the faster the average reading is done.

Look out: While a block is read, other processes even with higher priority may be delayed. The probability of a delay rises with the block size.

- The time offset between continuous conversion and blockwise reading demands a data buffer. Therefore, the initialization of the burst sequence must allocate sufficient memory range (**P2_Burst_Init**, parameter *samples*).

With module version Pro II-AIn-F-4/16-D Rev. E (D-Sub female connector), a burst sequence can be controlled by external event signals, i.e. each (resulting) event signal has a measurement value stored.

As an option one channel of the burst sequence may be used as time channel, which holds the counter value of the internal module timer for each event signal.

The module is equipped with 3 differential event inputs: *EVENT/A, B, ENABLE*, whose signals are processed to the resulting event signal. This pre-processing of signals be configured with **P2_Event2_Config**.

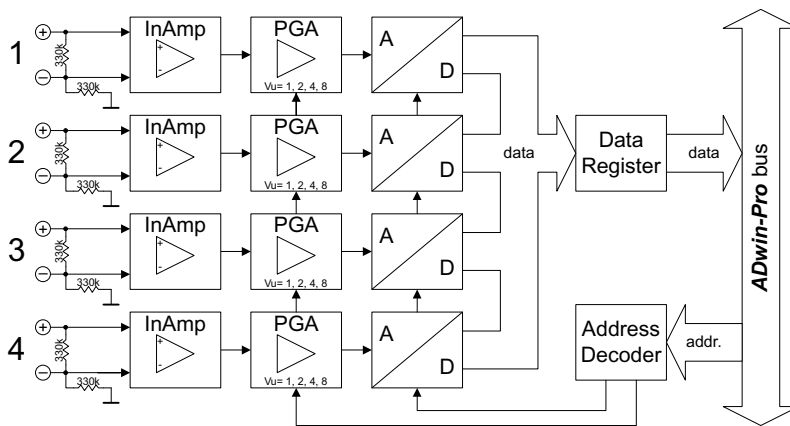


Fig. 61 – Pro II-AIn-F-4/16 Rev. E: Block diagram

Input channels	4 differential
Resolution	16 Bit
Conversion time	0.25µs (per ADC)
Input band width	0 ... 600kHz
Memory size (since rev. E04)	256MiB or $2^{27} = 134217728$ values total
Measurement range	±10V with max. offset 3.5 V
Gain	1, 2, 4, 8 software selectable
Accuracy	INL typical ±1.2 LSB, max. ±5 LSB
	DNL typical ±0.5 LSB, max. ±1 LSB
Input resistance	330kΩ, ±2%
Input over-voltage	±20V
Offset error	adjustable
Offset drift	±30ppm/°C of full scale range
Event input (D-Sub only)	3 differential; RS422/485 compatible (5V differential, 120 Ω bus terminating resistor)

Fig. 62 – Pro II-AIn-F-4/16 Rev. E: Specification

Event Inputs

Connector	4 LEMO female connectors. 1-pole, or 4 LEMO female connectors. 2-pole, or 37-pin D-Sub female connector, or 4 BNC female connectors
-----------	---

Fig. 62 – Pro II-AIn-F-4/16 Rev. E: Specification

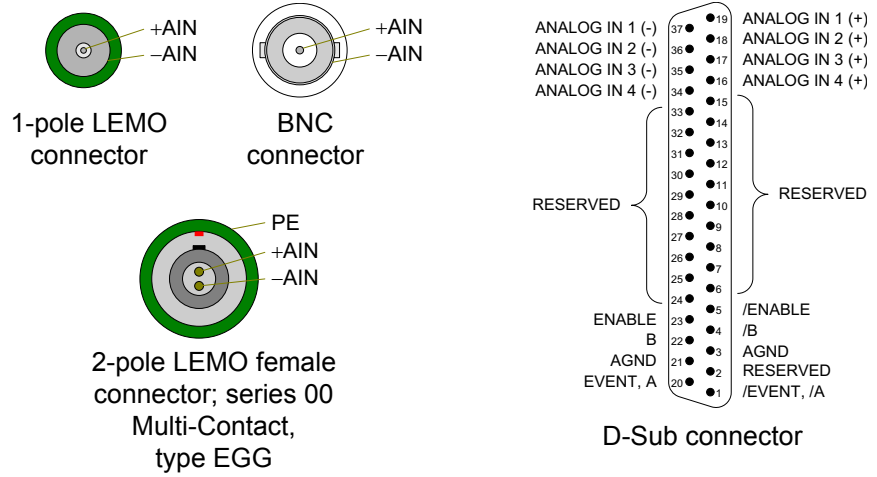


Fig. 63 – Pro-AIn-F-4/16-D Rev. E: Pin assignment differential

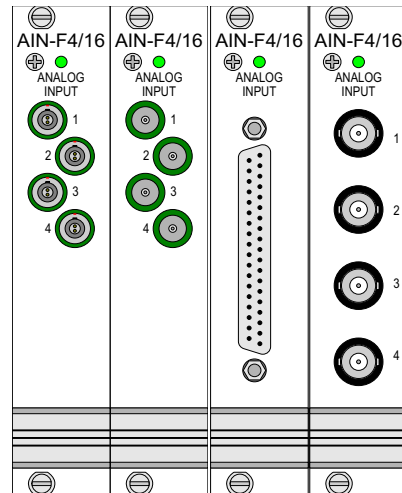


Fig. 64 – Pro II-AIn-F-4/16 Rev. E: Front panels

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode,</code> <code>P2_Set_Average_Filter,</code> <code>P2_Set_Gain</code>
Do a single conversion – complete or step by step	<code>P2_ADCF</code> <code>P2_Start_ConvF, P2_Wait_EOCF</code> <code>P2_Read_ADCF</code>

Function	Instructions
Read several values	P2_Read_ADCF32 P2_Read_ADCF4 P2_Read_ADCF4_Packed
Read value and start new conversion	P2_Read_ADC_SConv P2_Read_ADC_SConv32
Use burst conversion	P2_Burst_Init P2_Burst_Start, P2_Burst_Stop P2_Burst_Status, P2_Burst_Reset P2_Burst_Read P2_Burst_Read_Index, P2_Burst_Read_Index P2_Burst_Read_Unpacked1/2/4 P2_Burst_CRead_Unpacked1/2/4 P2_Burst_CRead_Pos_Unpacked1/2/4
Synchronize	P2_Sync_All, P2_Sync_Enable P2_Sync_Mode, P2_Sync_Stat
Control input limits	P2_ADCF_Read_Limit P2_ADCF_Set_Limit P2_ADCF_Reset_Min_Max, P2_ADCF_Read_Min_Max4, P2_ADCF_Read_Min_Max8
Use LED	P2_Check_LED, P2_Set_LED
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config, P2_Event2_Config

5.5.9 Pro II-AIn-F-8/16 Rev. E

The analog input module Pro II-AIn-F-8/16 Rev. E has 8 Fast-ADC of 16 Bit and 8 differential inputs.

The module can be ordered in the following variants:

	no filter	filter 50kHz	filter 10kHz, range $\pm 30V$
LEMO 1-pole	AIn-F-8/16	AIn-F-8/16-LP50	AIn-F-8/16-LP-30V
LEMO 2-pole	AIn-F-8/16-L2	AIn-F-8/16-LP50-L2	AIn-F-8/16-LP-30V-L2
D-Sub	AIn-F-8/16-D	AIn-F-8/16-LP50-D	AIn-F-8/16-LP-30V-D
BNC	AIn-F-8/16-B	AIn-F-8/16-LP50-B	AIn-F-8/16-LP-30V-B

The variants Pro II-AIn-8/18-xxx-LP contain an input filter, a 4th grade low pass of type Typ Butterworth. According to the variant the frequency is set to 10kHz or 50kHz.

The inputs are provided on shielded LEMO sockets (1-pole or 2-pole, CAMAC European norm, see [fig. 67](#)), BNC sockets, or 37-pin D-Sub connector.

The module's converters run with a clock rate of up to 4MHz. Since revision E04, module memory enables to buffer data—especially with burst sequences.

The module Pro II-AIn-F-8/16 Rev. E has an input voltage range of $\pm 10V$ and a software selectable gain of 1, 2, 4 or 8. The adjustment of gain and offset is done by software (see [chapter 6 "Calibration"](#)).

Since revision E04, the module provides several operation modes for converting analog signals:

- Single measurement: You start each conversion manually, query the measurement value and processes it in the *ADbasic* program.
- Single burst sequence: The *ADbasic* program starts a complete burst measurement sequence, that is a defined number of single measurements.
- Continuous burst sequence: The *ADbasic* program starts a burst measurement sequence, that continuously does single measurements until the sequence is stopped. The data is stored in a fifo-like memory.

As an alternative to the operation modes above, the module can perform a simultaneous conversion on all channels using a sequence control. This disburdens—compared to single measurements with the instruction `P2_ADCF`—the processor module, which only reads and processes the converted measurement values. The conversion can be either triggered periodically with regular time delays or by external event signals.

As an option, the module can return the average of 2...32 measurement values instead of simple measurement values. In this case, the selected number of measurement values is converted before the average can be calculated.

The module can control a lower and an upper limit for each channel separately.

The module gathers the maximum and the minimum value for each channel and provides the values per software instructions.

The module processes burst sequences independent from the processor module of the *ADwin* system. The measurement values—number and measuring rate to be defined ahead in the program—are stored in the burst-memory of the module. The processor module then reads the stored values (even during a

Burst Sequence

burst sequence) and processes them.
Burst sequences are available since rev. E04.

With a continuous burst-sequence, the measuring rate must be synchronized to the reading rate. Please note:

- Measurement values are always read in blocks, the block size is selectable. The greater the data blocks are, the faster the average reading is done.

Look out: While a block is read, other processes even with higher priority may be delayed. The probability of a delay rises with the block size.

- The time offset between continuous conversion and blockwise reading demands a data buffer. Therefore, the initialization of the burst sequence must allocate sufficient memory range (**P2_Burst_Init**, parameter **samples**).

With module version Pro II-AIn-F-8/16-D Rev. E (D-Sub female connector), a burst sequence can be controlled by external event signals, i.e. each (resulting) event signal has a measurement value stored.

As an option one channel of a burst sequence may be used as time channel, holding the counter value of the internal module timer for each event signal.

The module is equipped with 3 differential event inputs: **EVENT/A, B, ENABLE**, whose signals are processed to the resulting event signal. This pre-processing of signals be configured with **P2_Event2_Config**.

Event Inputs

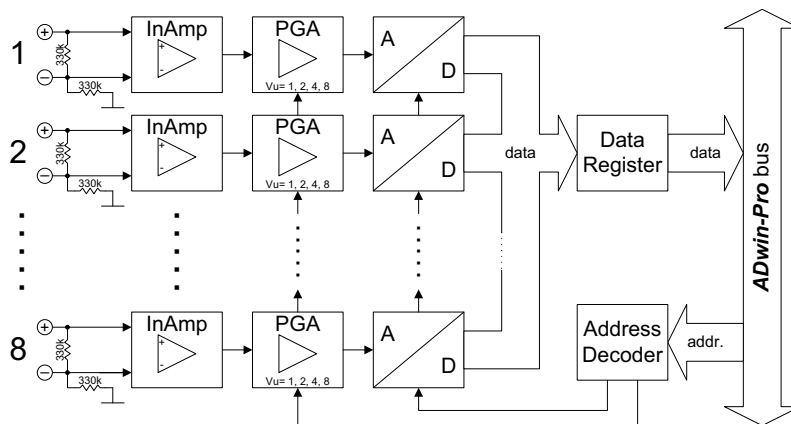


Fig. 65 – Pro II-AIn-F-8/16 Rev. E: Block diagram

Input channels	8 differential
Resolution	16 Bit
Conversion time	0.25µs (per ADC)
Input band width	0 ... 600kHz
Memory size (since rev. E04)	256MiB or $2^{27} = 134217728$ values total
Measurement range	±10V with max. offset 3.5 V
Gain	1, 2, 4, 8 software selectable
Accuracy	INL typical ±1.2 LSB, max. ±5 LSB
	DNL typical ±0.5 LSB, max. ±1 LSB
Input resistance	330kΩ, ±2%
Input over-voltage	±20V
Offset error	adjustable

Fig. 66 – Pro II-AIn-F-8/16 Rev. E: Specification

Offset drift	±30ppm/°C of full scale range
Event input (D-Sub only)	3 differential; RS422/485 compatible (5V differential, 120 Ω bus terminating resistor)
Connector	8 LEMO female connectors. 1-pole, or 8 LEMO female connectors. 2-pole, or 37-pin D-Sub female connector, or 8 BNC female connectors.
Module width	5 HP; with BNC connectors: 10 HP

Fig. 66 – Pro II-AIn-F-8/16 Rev. E: Specification

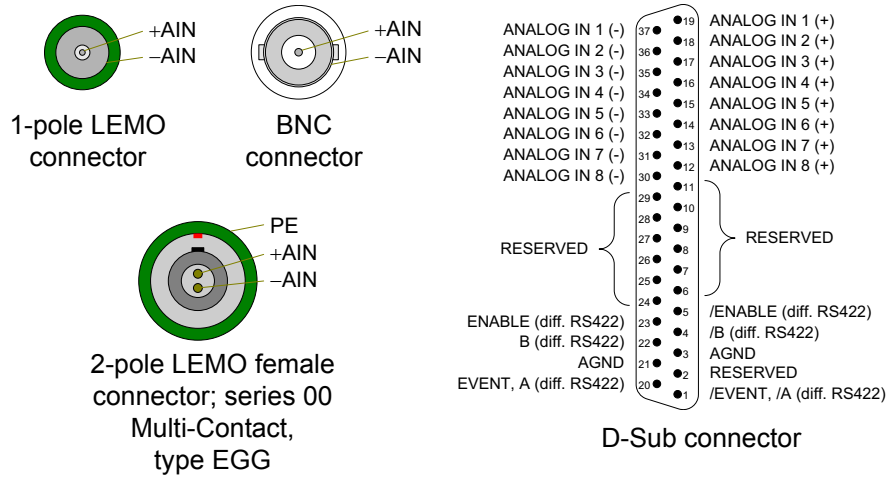


Fig. 67 – Pro II-AIn-F-8/16 Rev. E: Pin assignment differential

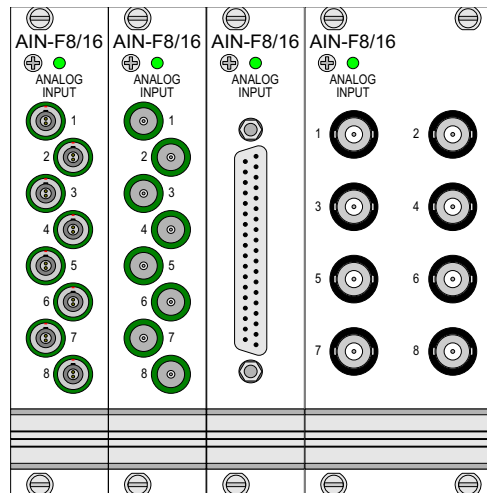


Fig. 68 – Pro II-AIn-F-8/16 Rev. E: Front panels

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode,</code> <code>P2_Set_Average_Filter,</code> <code>P2_Set_Gain</code>

Function	Instructions
Do a single conversion – complete or step by step	P2_ADCF P2_Start_ConvF, P2_Wait_EOCF P2_Read_ADCF
Read several values	P2_Read_ADCF32 P2_Read_ADCF4, P2_Read_ADCF8 P2_Read_ADCF4_Packed P2_Read_ADCF8_Packed
Read value and start new conversion	P2_Read_ADC_SConv P2_Read_ADC_SConv32
Use burst conversion	P2_Burst_Init P2_Burst_Start, P2_Burst_Stop P2_Burst_Status, P2_Burst_Reset P2_Burst_Read P2_Burst_Read_Index, P2_Burst_Read_Index P2_Burst_Read_Unpacked1/2/4/8 P2_Burst_CRead_Unpacked1/2/4/8 P2_Burst_CRead_Pos_Unpacked1/2/4/8
Synchronize	P2_Sync_All, P2_Sync_Enable P2_Sync_Stat
Control input limits	P2_ADCF_Read_Limit P2_ADCF_Set_Limit
Provide maximum and minimum values	P2_ADCF_Reset_Min_Max P2_ADCF_Read_Min_Max4 P2_ADCF_Read_Min_Max8
Use LED	P2_Check_LED, P2_Set_LED
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config, P2_Event2_Config

5.5.10 Pro II-AIn-F-4/18 Rev. E

The analog input module Pro II-AIn-F-4/18 Rev. E has 4 Fast ADC of 18 Bit and 4 differential inputs. The inputs are galvanically isolated from each other as well as from other modules.

The inputs are available with the following connectors:

- Pro II-AIn-F-4/18: shielded LEMO female connectors, CAMAC European norm.
- Pro II-AIn-F-4/18-L2: shielded LEMO female connectors, 2-pole, CAMAC European norm.
- Pro II-AIn-F-4/18-D: D-Sub female connector 37-pin.
- Pro II-AIn-F-4/18-B: BNC connectors.

The module Pro II-AIn-F-4/18 Rev. E has input voltage range of $\pm 10V$. The adjustment of gain and offset is made by software. (see [chapter 6 "Calibration"](#)).

The module can perform a simultaneous conversion on all channels using a sequence control. This disburdens—compared to single measurements with the instruction **P2_ADCF**—the processor module, which only reads and processes the converted measurement values. The conversion can be either triggered periodically with regular time delays or by external event signals.

The module can control an upper and lower limit for each channel separately.

In order to use a channel as single-ended input, connect the input signal to the plus pin and ground to the minus pin.

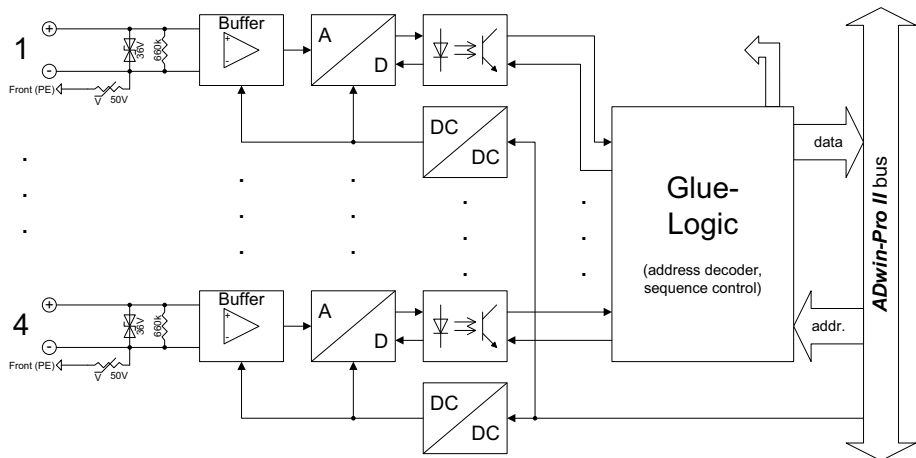


Fig. 69 – Pro II-AIn-F-4/18 Rev. E: Block diagram

As ground connection the enclosure’s ground (PE) is available. All analog inputs are galvanically isolated from enclosure’s ground.

With module versions -L2 (2-pole Lemo) and -D (D-Sub connector), the enclosure’s ground is also available at the shield of the connectors.

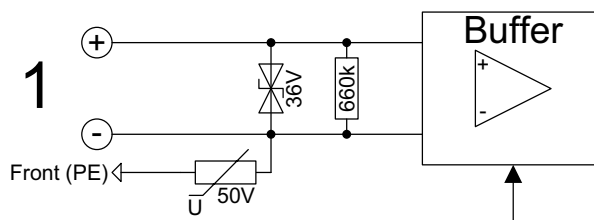


Fig. 70 – Pro II-AIn-F-4/18 Rev. E: Input circuitry

Input channels	4 differential, galvanically isolated
Resolution	18 Bit
Conversion time	max. 2µs (per ADC)
Sampling rate	max. 500ksps (per ADC)
Input band width	0 ... 0.6MHz
Measurement range	±10V
Accuracy	INL max. ±4 LSB
	DNL max. ±3 LSB
Input resistance	660kΩ, ±2%
Input over-voltage:	±35V
Offset error	adjustable
Offset drift	±30ppm/°C of full scale range
Connector	4 LEMO female connectors, 1-pole or 4 LEMO female connectors, 2-pole or 37-pin D-Sub female connector or 4 BNC connectors

Fig. 71 – Pro II-AIn-F-4/18 Rev. E: Specification

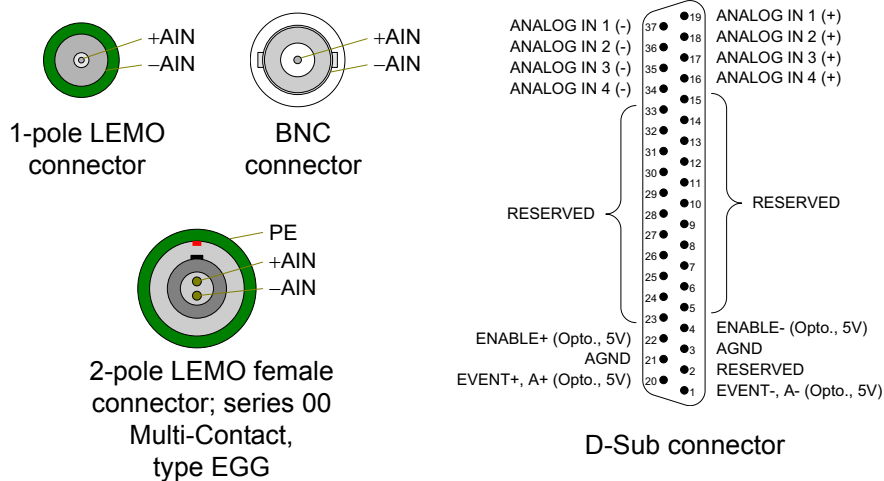


Fig. 72 – Pro II-AIn-F-4/18 Rev. E: Pin assignment differential

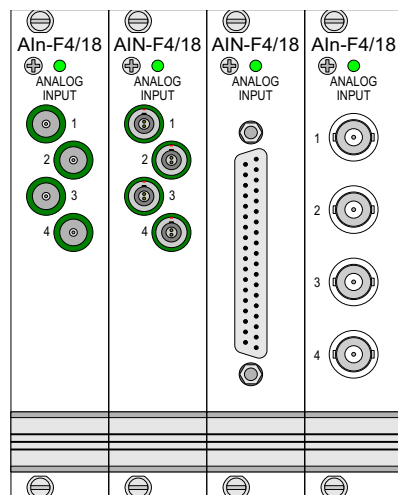


Fig. 73 – Pro II-AIn-F-4/18 Rev. E: Front panels

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode</code>
Do a single conversion – complete or step by step	<code>P2_ADCF, P2_ADCF24</code> <code>P2_Start_ConvF, P2_Wait_EOCF</code> <code>P2_Read_ADCF, P2_Read_ADCF24</code>
Read several values	<code>P2_Read_ADCF32</code> <code>P2_Read_ADCF4, P2_Read_ADCF4_24B</code> <code>P2_Read_ADCF4_Packed</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code> <code>P2_Read_ADC_SConv32</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Control input limits	<code>P2_ADCF_Read_Limit</code> <code>P2_ADCF_Set_Limit</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config, P2_Event2_Config</code>

5.5.11 Pro II-AIn-F-8/18 Rev. E

The analog input module Pro II-AIn-F-8/18 Rev. E has 8 Fast-ADC of 18 Bit and 8 differential inputs. The inputs are galvanically isolated from each other as well as from other modules.

The inputs are available with the following connectors:

- Pro II-AIn-F-8/18: shielded LEMO female connectors, CAMAC European norm.
- Pro II-AIn-F-8/18-L2: shielded LEMO female connectors, 2-pole, CAMAC European norm.
- Pro II-AIn-F-8/18-D: D-Sub female connector 37-pin.
- Pro II-AIn-F-8/18-B: BNC connectors.

The module Pro II-AIn-F-8/18 Rev. E has input voltage range of $\pm 10V$. The adjustment of gain and offset is made by software. (see [chapter 6 "Calibration"](#)).

The module can perform a simultaneous conversion on all channels using a sequence control. This disburdens—compared to single measurements with the instruction **P2_ADCF**—the processor module, which only reads and processes the converted measurement values. The conversion can be either triggered periodically with regular time delays or by external event signals.

The module can control an upper and lower limit for each channel separately.

In order to use a channel as single-ended input, connect the input signal to the plus pin and ground to the minus pin.

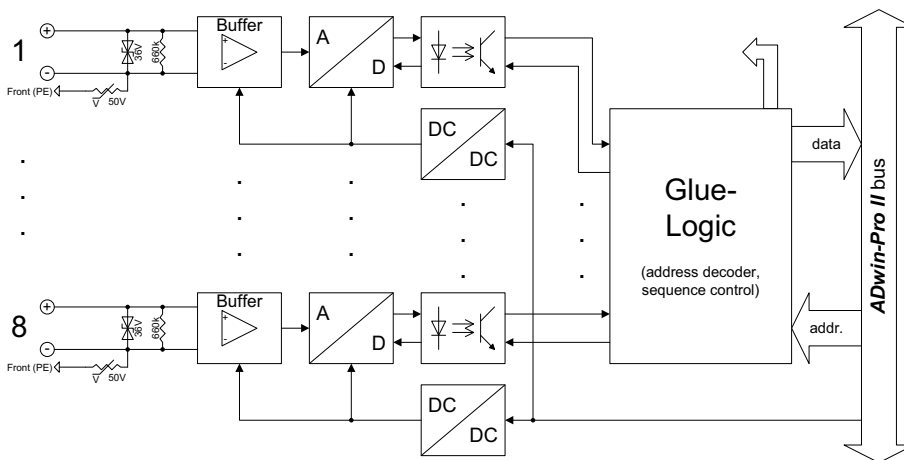


Fig. 74 – Pro II-AIn-F-8/18 Rev. E: Block diagram

As ground connection the enclosure's ground (PE) is available. All analog inputs are galvanically isolated from enclosure's ground.

With module versions -L2 (2-pole Lemo) and -D (D-Sub connector), the enclosure's ground is also available at the shield of the connectors.

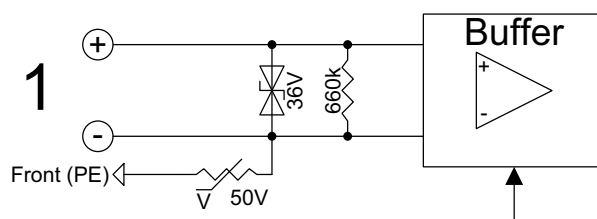


Fig. 75 – Pro II-AIn-F-8/18 Rev. E: Input circuitry

Input channels	8 differential, galvanically isolated	
Resolution	18 Bit	
Conversion time	max. 2µs (per ADC)	
Sampling rate	max. 500 ksps (per ADC)	
Input band width	0 ... 0.6MHz	
Measurement range	±10V	
Accuracy	INL	max. ±4 LSB
	DNL	max. ±3 LSB
Input resistance	660kΩ, ±2%	
Input over-voltage	±35V	
Offset error	adjustable	
Offset drift	±30ppm/°C of full scale range	
Connector	8 LEMO female connectors, 1-pole or 8 LEMO female connectors, 2-pole or 37-pin D-Sub female connector or 8 BNC connectors	
Module width	5 HP; with BNC connectors: 10 HP	

Fig. 76 – Pro II-AIn-F-8/18 Rev. E: Specification

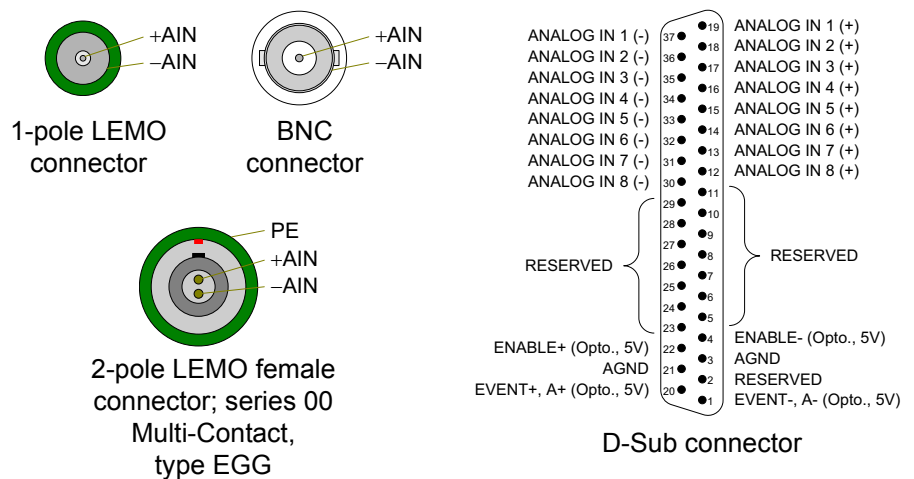


Fig. 77 – Pro II-AIn-F-8/18 Rev. E: Pin assignment differential

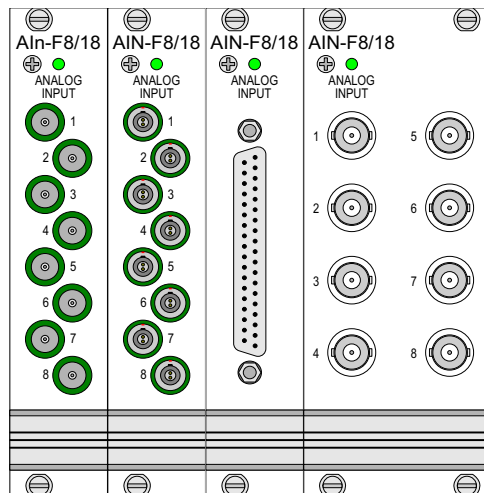


Fig. 78 – Pro II-AIn-F-8/18 Rev. E: Front panels

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure sequence control	<code>P2_ADCF_Mode</code>
Do a single conversion – complete or step by step	<code>P2_ADCF, P2_ADCF24</code> <code>P2_Start_ConvF, P2_Wait_EOCF</code> <code>P2_Read_ADCF, P2_Read_ADCF24</code>
Read several values	<code>P2_Read_ADCF32</code> <code>P2_Read_ADCF4, P2_Read_ADCF4_24B</code> <code>P2_Read_ADCF4_Packed</code> <code>P2_Read_ADCF8, P2_Read_ADCF8_24B</code> <code>P2_Read_ADCF8_Packed</code>
Read value and start new conversion	<code>P2_Read_ADC_SConv</code> <code>P2_Read_ADC_SConv24</code> <code>P2_Read_ADC_SConv32</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Control input limits	<code>P2_ADCF_Read_Limit</code> <code>P2_ADCF_Set_Limit</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config, P2_Event2_Config</code>

Programming in ADbasic

5.6 Pro II: Analog Output Modules

This section describes analog output modules for *ADwin-Pro II*. Analog output modules for *ADwin-Pro I* be found in the manual "ADwin-Pro Hardware" from page 65.

Module name	AOut 4/16	AOut 4/16-TiCo	AOut 8/16	AOut 8/16-TiCo	AOut 1/16
Revision	E	E	E	E	E
Number DAC	4	4	8	8	1
Resolution [bit]	16	16	16	16	16
max. settling time [μs]	< 3	< 3	< 3	< 3	0.015
Channels sng. end.	4	4	8	8	1
Output voltage	±10V	±10V	±10V	±10V	±2V
Calibration by software	yes	yes	yes	yes	yes
TiCo processor	–	TiCo1	–	TiCo1	–
page	93	93	96	96	99

5.6.1 Pro II-AOut-4/16 Rev. E

The analog output module Pro II-AOut-4/16 Rev. E has 4 DAC (16 bit) with fixed 1st order low-pass filters ($f_c = 10\text{MHz}$).

The module can be ordered in following variants:

	without TiCo processor	with TiCo processor
shielded LEMO female connectors, 1-pole, CAMAC European norm	Pro II-AOut-4/16	Pro II-AOut-4/16-TiCo
shielded LEMO female connectors, 2-pole, CAMAC European norm	Pro II-AOut-4/16-L2	Pro II-AOut-4/16-L2-TiCo
D-Sub female connector 37-pin	Pro II-AOut-4/16-D	Pro II-AOut-4/16-D-TiCo
BNC sockets	Pro II-AOut-4/16-B	Pro II-AOut-4/16-B-TiCo

The output voltage range of the DACs is set to $\pm 10\text{V}$ bipolar and can't be changed. Offset and gain are adjusted by software (see [chapter 6 "Calibration"](#)).

Modules with D-Sub female connector have an event input; an event given may be forwarded as trigger signal to the processor module.

The variants Pro II-Aln-8/18-xxx-TiCo additionally provide a freely programmable *TiCo* processor with 28KiByte data memory and 28KiByte program memory, which has access to all inputs of the module. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

TiCo processor

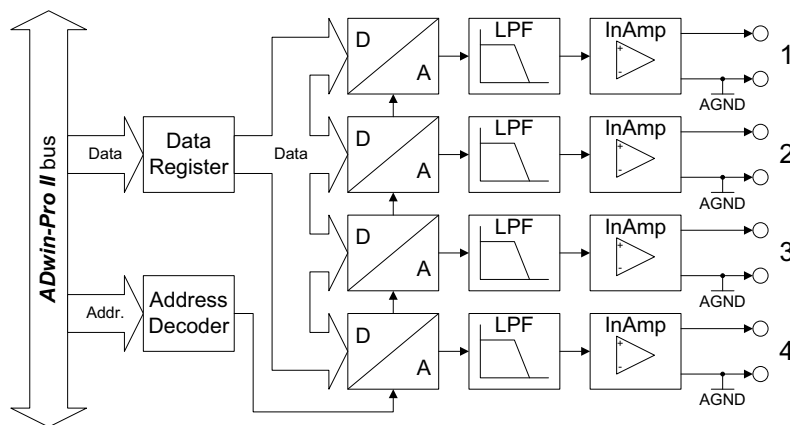


Fig. 79 – Pro II-AOut-4/16 Rev. E: Block diagram

Output channels	4 single-ended
Resolution	16 bit
Settling time to 0.01% FSR	$< 3\mu\text{s}$
Output voltage	$\pm 10\text{V}$

Fig. 80 – Pro II-AOut-4/16 Rev. E: Specification

Output current max.	±5mA per channel for optimal function ±35mA technically possible short-circuit-proof	
Accuracy	INL	±2 LSB typical
	DNL	±1 LSB typical
Offset error	adjustable	
Gain error	adjustable	
Offset drift	±10 µV/°C	
TiCo processor, only with Pro II-AIn-8/18-L2-TiCo Rev. E	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal	
Connector	4 LEMO female connectors, 1-pole or 4 LEMO female connectors, 2-pole or 37-pin D-Sub female connector or 4 BNC connectors	

Fig. 80 – Pro II-AOut-4/16 Rev. E: Specification

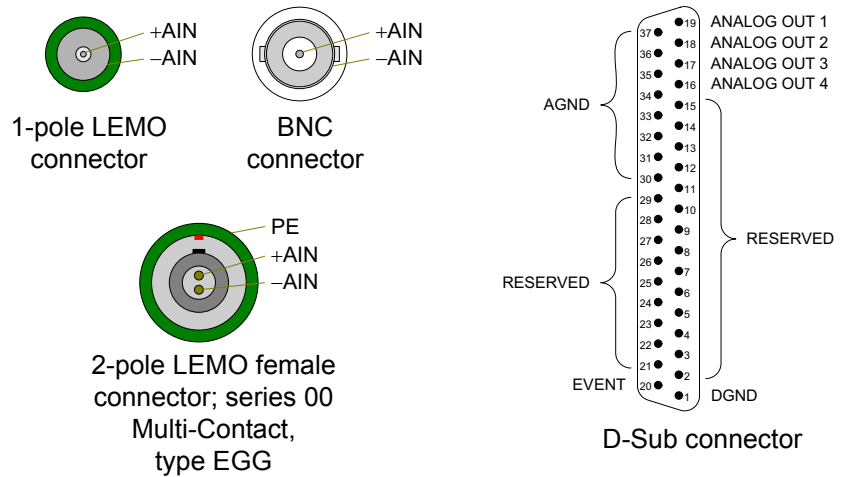


Fig. 81 – Pro II-AOut-4/16 Rev. E: Pin assignment

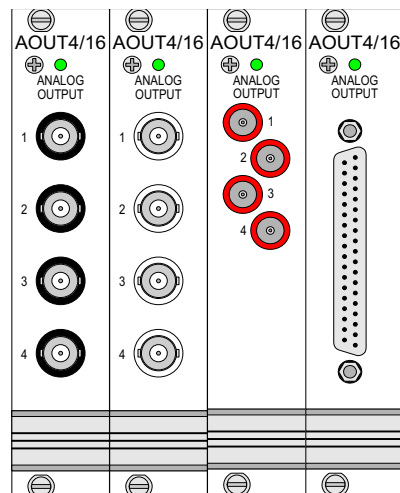


Fig. 82 – Pro II-AOut-4/16 Rev. E: Front covers

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Output values	<code>P2_DAC, P2_DAC4, P2_DAC4_Packed</code>
Output values step by step	<code>P2_Write_DAC, P2_Write_DAC4 P2_Write_DAC4_Packed P2_Write_DAC32 P2_Start_DAC</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AOut_TiCo.inc` contains instructions for the following functions:

Function	Instruction
Output values	<code>DAC</code>
Output values step by step	<code>Write_DAC, Write_DAC32, Start_DAC</code>
Use LED	<code>Check_LED, Set_LED</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block P2_Get_TiCo_RingBuffer, P2_Set_TiCo_RingBuffer P2_RingBuffer_Empty P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start, P2_TiCo_Stop P2_Get_TiCo_Bootloader_ Status P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status P2_TiCo_Get_Processdelay P2_TiCo_Set_Processdelay P2_TiCo_Start_Process P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

Programming

Programming in *TiCoBasic*

Programming *TiCo* access

TiCo processor

5.6.2 Pro II-AOut-8/16 Rev. E

The analog output module Pro II-AOut-8/16 Rev. E has 8 DAC (16 bit) with fixed 1st order low-pass filters ($f_c = 10\text{MHz}$).

The module can be ordered in following variants:

	without TiCo processor	with TiCo processor
shielded LEMO female connectors, 1-pole, CAMAC European norm	Pro II-AOut-8/16	Pro II-AOut-8/16-TiCo
shielded LEMO female connectors, 2-pole, CAMAC European norm	Pro II-AOut-8/16-L2	Pro II-AOut-8/16-L2-TiCo
D-Sub female connector 37-pin	Pro II-AOut-8/16-D	Pro II-AOut-8/16-D-TiCo
BNC sockets	Pro II-AOut-8/16-B	Pro II-AOut-8/16-B-TiCo

The output voltage range of the DACs is set to $\pm 10\text{V}$ bipolar and can't be changed. Offset and gain are adjusted by software (see chapter 6 "Calibration").

Modules with D-Sub female connector have an event input; an event given may be forwarded as trigger signal to the processor module.

The variants Pro II-AIn-8/18-xxx-TiCo additionally provide a freely programmable *TiCo* processor with 28KiByte data memory and 28KiByte program memory, which has access to all inputs of the module. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

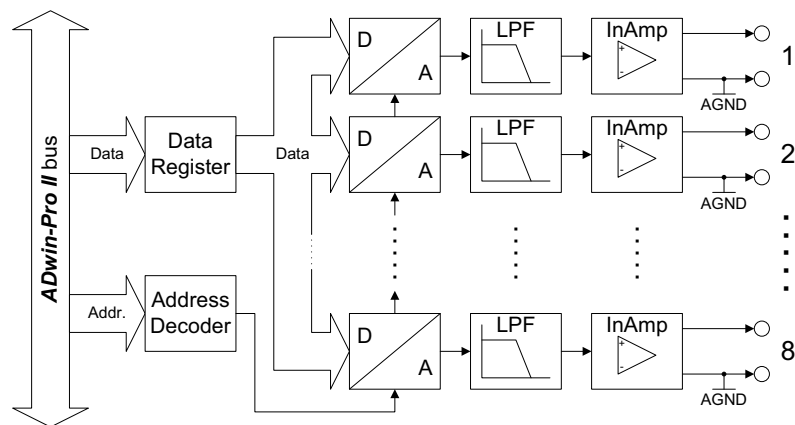


Fig. 83 – Pro II-AOut-8/16 Rev. E: Block diagram

Output channels	8 single-ended
Resolution	16 bit
Settling time to 0.01% FSR	$< 3\ \mu\text{s}$
Output voltage	$\pm 10\text{V}$

Fig. 84 – Pro II-AOut-8/16 Rev. E: Specification

Output current max.	±5mA per channel for optimal function ±35mA technically possible, short-circuit-proof	
Accuracy	INL	±2 LSB typical
	DNL	±1 LSB typical
Offset error	adjustable	
Gain error	adjustable	
Offset drift	±10 µV/°C	
TiCo processor, only with Pro II-AIn-8/18-L2-TiCo Rev. E	Processor type: TiCo1 Clock frequency: 50MHz Memory size: 28KiB PM internal, 28KiB DM internal	
Connector	8 LEMO female connectors, 1-pole or 8 LEMO female connectors, 2-pole or 37-pin D-Sub female connector or 8 BNC connectors	
Module width	5 HP; with BNC or LEMO 2-pole: 10 HP	

Fig. 84 – Pro II-AOut-8/16 Rev. E: Specification

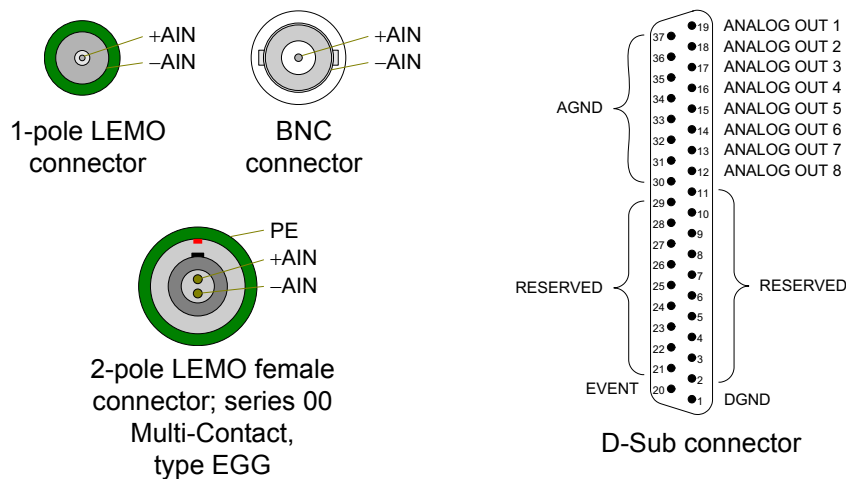


Fig. 85 – Pro II-AOut-8/16 Rev. E: Pin assignment

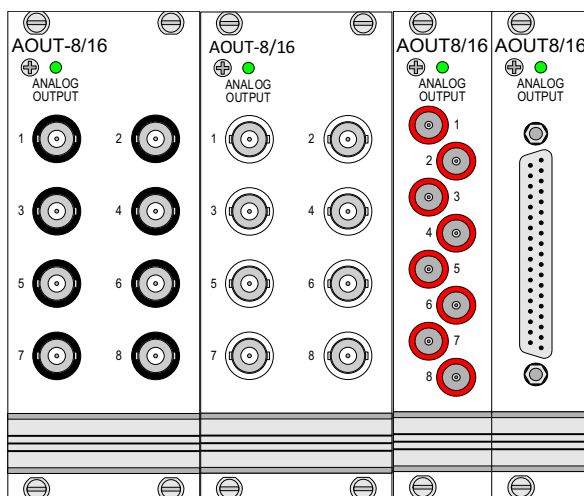


Fig. 86 – Pro II-AOut-8/16 Rev. E: Front covers

Programming

The module is comfortably programmed with *ADbasic* instructions. They are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for:

Function	Instructions
Output values	<code>P2_DAC, P2_DAC4, P2_DAC4_Packed</code> <code>P2_DAC8, P2_DAC8_Packed</code>
Output values step by step	<code>P2_Write_DAC, P2_Write_DAC4</code> <code>P2_Write_DAC4_Packed</code> <code>P2_Write_DAC8</code> <code>P2_Write_DAC8_Packed</code> <code>P2_Write_DAC32</code> <code>P2_Start_DAC</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config</code>

Programming in TiCoBasic

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AOut_TiCo.inc` contains instructions for the following functions:

Function	Instruction
Output values	<code>DAC</code>
Output values step by step	<code>Write_DAC, Write_DAC32, Start_DAC</code>
Use LED	<code>Check_LED, Set_LED</code>

Programming TiCo access

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start,</code> <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

5.6.3 Pro II-AOut-1/16 Rev. E

The module Pro II-AOut-1/16 Rev. E is equipped with the following hardware:

- 1 analog output with 16 Bit DAC, output rate 50MHz, galvanically isolated.
- Each 16 digital inputs and outputs with TTL levels
- 1 Event input
- *TiCo* processor with 56 KiB internal memory and 256MiB external DRAM

The *TiCo* processor has access to all inputs and outputs of the module, as does the *ADwin* CPU. You find more information about usage and programming of the *TiCo* processor in the *TiCoBasic* manual.

Analog output

The DAC runs with 50MHz speed. The output voltage range of the DAC is set to $\pm 2V$ bipolar and cannot be changed. The calibration of gain and offset is done via software (see [chapter 6 "Calibration"](#)).

The DAC output is located in a separate SMB connector (female) and is galvanically isolated from enclosing's ground and from other modules.

Please note: The DAC has an output impedance of 50Ω and must be operated with a 50Ω system. As soon as the output was calibrated, any change to the impedance value—for example by exchanging a cable or a plug—may lead to a strong aberration in the output voltage value.

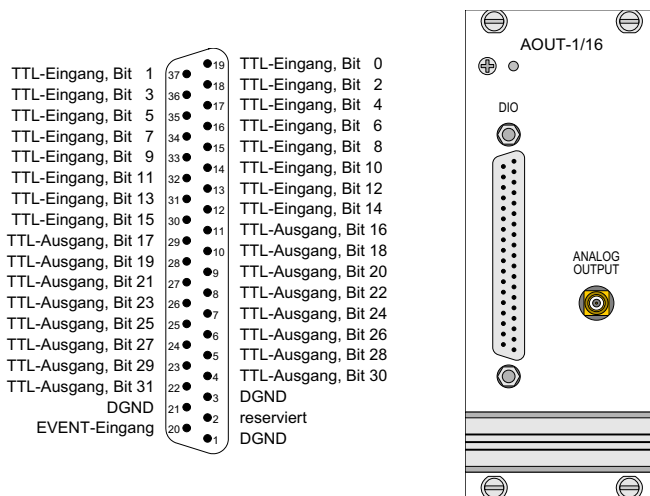


Fig. 87 – Pro II-AOut-1/16 Rev. E: Pin assignment and front cover

To output DAC values there are 3 variants:

1. Single value output
2. Continuous output via an output Fifo. The output values can be either refilled regularly or will be repeated in an endless loop.

With each output voltage value, 14 digital outputs can be set either.

3. Output of voltage ramps being defined by DAC start value, DAC end value, and span of time. The combination of consecutive ramps enables the output of curves.

In addition to a voltage ramp, at the start and/or at the end of the ramp TTL signals can be set at the 16 digital outputs.

The 3 output variants can only be used one after the other but not in parallel. If you change to a different output variant there will in any case be a time delay until the next voltage value is output.

Digital Inputs/Outputs

16 input and 16 outputs with TTL levels are available on a 37-pin D-Sub female connector, pin assignment see above. The channel direction cannot be changed.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

The FIFO can be used either for edge control of inputs or for output of level patterns at the outputs.

Via the EVENT input a trigger signal can start an externally triggered *ADbasic* process, which is processed immediately and completely (see *ADbasic* manual).

TiCo processor

The module provides the freely programmable *TiCo* processor with 28kiB program memory, 28KiB data memory, and 256MiB external DRAM memory. The internal memory serves as data and program memory. You program the *TiCo* processor with *TiCoBasic*.

The *TiCo* processor has access to all analog and digital input and output channels, as does the *ADwin* CPU. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Technical Specification

Analog Output		
Output channels	1 single ended	
Resolution	16 Bit	
Settling time to 0.01% FSR	15ns	
Output voltage range	±2V	
Maximum output current	±40mA at 50Ω load short-term short-circuit proof	
Accuracy	INL	±8 LSB max.
	DNL	±1 LSB typically
Offset error	adjustable	
Gain error	adjustable	
Offset drift	±10 μV/°K	
Digital Inputs/Outputs		
Digital inputs/outputs	16 inputs and 16 outputs with TTL logic	

Fig. 88 – Pro II-AOut-1/16 Rev. E: Specification

Pull down resistor	10kΩ
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 500μA
I _{IL}	max. 10μA
Voltage range	-0,5V ... +5,5V
Output current	max. ±24mA per channel via V _{CC} or GND, max. ±70mA per block of 8 channels via V _{CC} or GND
Event input	TTL logic
General	
Input / output FIFO	Size: 511 value pairs Frequency: 100MHz
TiCo	Prozessor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal, 256MiByte DRAM external
Connector	37-pin D-Sub female connector, SMB con- nector, female
Module width	10 HP

Fig. 88 – Pro II-AOut-1/16 Rev. E: Specification

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Analog Output	
Output single value	<code>P2_DAC, P2_DAC1_DIO</code>
Output value step by step	<code>P2_Write_DAC, P2_Start_DAC</code>
Output DAC value and digital pattern via output Fifo	<code>P2_Digout_Fifo_Clear</code> <code>P2_Digout_Fifo_Read_Timer</code> <code>P2_Digout_Fifo_Empty</code> <code>P2_Dig_Fifo_Mode</code> <code>P2_Digout_Fifo_Start</code> <code>P2_Digout_Fifo_Enable</code> <code>P2_Digout_Fifo_Write</code>
Ramp mode	<code>P2_DAC_Ramp_Write</code> <code>P2_DAC_Ramp_Status</code> <code>P2_DAC_Ramp_Buffer_Free</code> <code>P2_DAC_Ramp_Stop</code>
Digital I/Os	
Query input signalst	<code>P2_Digin_Long</code>
Monitor edges of input channels	<code>P2_Digin_Fifo_Clear</code> <code>P2_Digin_Fifo_Read</code> <code>P2_Digin_Fifo_Read_Fast</code> <code>P2_Digin_Fifo_Read_Timer</code> <code>P2_Digin_Fifo_Full</code> <code>P2_Digin_Fifo_Enable</code>

Programming in ADbasic

Programming in
TiCoBasic

Function	Instructions
Set and read back output signals	P2_Digout, P2_Digout_Long P2_Get_Digout_Long P2_Digout_Bits P2_Digout_Set, P2_Digout_Reset
Use latch register	P2_Dig_Latch P2_Dig_Read_Latch P2_Dig_Write_Latch
General	
Start actions synchronously	P2_Sync_All, P2_Sync_Enable P2_Sync_Stat
Use LED	P2_Check_LED, P2_Set_LED
Use interrupt and event inputs	P2_Event_Enable, P2_Event_Read P2_Event_Config

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AOUT1_TiCo.inc` contains instructions for the following functions:

Function	Instructions
Analog Output	
Output single value	DAC, DAC1_DIO
Output value step by step	Write_DAC, Start_DAC
Output DAC value and digital pattern via output Fifo	Digout_Fifo_Clear Digout_Fifo_Read_Timer Digout_Fifo_Empty Dig_Fifo_Mode Digout_Fifo_Start Digout_Fifo_Enable Digout_Fifo_Write
Ramp mode	DAC_Ramp_Write DAC_Ramp_Status DAC_Ramp_Buffer_Free DAC_Ramp_Stop
Digital I/Os	
Query input signalst	Digin_Long
Monitor edges of input channels	Digin_Fifo_Clear Digin_Fifo_Read_Timer Digin_Fifo_Full Digin_Fifo_Enable
Set and read back output signals	Digout, Digout_Long Get_Digout_Long Digout_Bits Digout_Set, Digout_Reset
Use latch register	Dig_Latch Dig_Read_Latch Dig_Write_Latch
General	
Use LED	Check_LED, Set_LED
Use interrupt and event inputs	Event_Enable, Event_Read Event_Config

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

Programming *TiCo* access

5.7 Pro II: Digital-I/O Modules

Digital I/O Modules

Module	Rev.	Type	Channels	Input Voltage U_{In}		High Level [mA]	Isolation [V]	Page	
				[V]					
DIO-32	E	TTL input / output	32	5	TTL	-	-	105	
DIO-32-TiCo	E	TTL input / output, with TiCo1 processor	32	5	TTL	-	-	107	
DIO-32/1-TiCo	E	TTL input / output single selection, with TiCo1 processor	32	5	TTL	-	-	111	
DIO-32-TiCo2	E	TTL input / output, with TiCo2 processor	32	5	TTL	-	-	115	
DIO-8-D12	E	8 TTL channels, 12 diff. channels with TiCo2 processor	8 + 12	5	TTL	-	-	119	
OPT-16	E	Optocouple input	16	5, 12, 24	DC	-	42	123	
OPT-32-24V	E	Optocouple input	32	24	DC	-	42	126	
REL-16	E	Relay output	16	max. 30	AC / DC	500	42	128	
TRA-16	E	Transistor output	16	5...30	DC	200	42	130	
PWM-16	E	PWM output signal	16	5	TTL	-	-	132	
PWM-16-I	E	PWM output signal	16	5...30	DC	100	42	132	
COMP-16	E	Digital inputs with comparators	16	-2...32	DC	-	-	135	
MIO-D12	E	Transistor output	12	5...30	DC	200		44	
		Optocouple input	12 s.e.	5, 12, 24	DC	-			
		Counter block	2	Universal, 32 Bit, 5V diff.					
		SSI decoder	1	max. 12.5MHz					
		TiCo1 Processor	-	56 KiByte internal memory					

Counter Modules

Module	Rev.	Channels	Counters			Input voltage. U_{In}		Isolation [V]	Page
			No.	Type ^a	Resol. [Bit]	[V]	Type		
CNT-T	E	4, with TiCo1 processor	1	U	32	5	TTL	-	139
CNT-I	E	4, with TiCo1 processor	1	U	32	5, 12, 24	DC	42	139
CNT-D	E	4 + 2 SSI, with TiCo1 processor	1	U	32	5 diff.	RS422/RS485	-	139

a. U: Universal counter = UD + I + PWM

5.7.1 Pro II-DIO-32 Rev. E

The digital input/output module Pro II-DIO-32 Rev. E provides 32 programmable digital input and output channels at TTL levels. The channels can be configured as blocks of 8 bits as inputs or outputs by *ADbasic* instructions. The channels are configured as inputs after power up.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

The variant Pro II-DIO-32-TiCo Rev. E provides in addition the freely programmable *TiCo* processor and may output levels at defined points in time to digital outputs as stand-alone. Find more on [page 107](#).

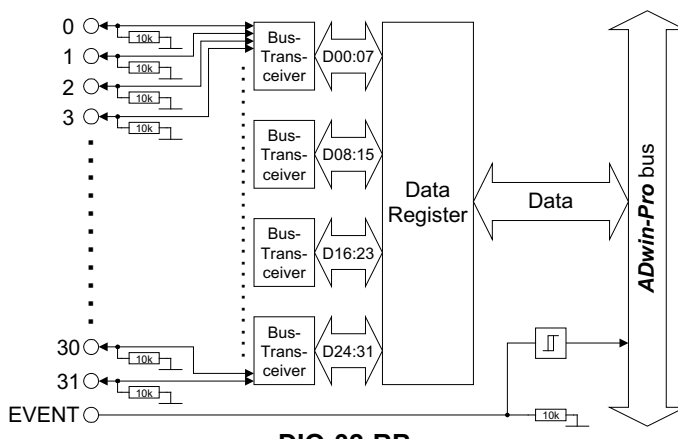


Fig. 89 – Pro II-DIO-32 Rev. E: Block diagram

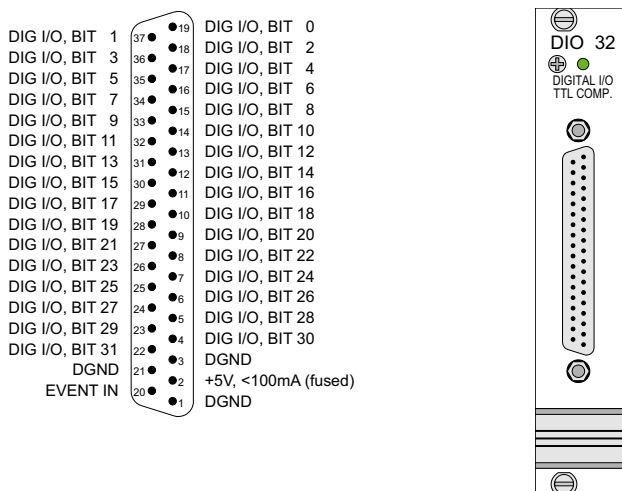


Fig. 90 – Pro II-DIO-32 Rev. E: Front panel and Pin assignment

Digital inputs	TTL logic
Input/output channels	32; programmable via software as inputs/outputs in blocks of 8
Pull down resistor	10kΩ

Fig. 91 – Pro II-DIO-32 Rev. E: Specification

V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1 μA
I _{IL}	max. 0.01 mA
Voltage range	-0.5V ... +5.5V
Output current	max. ±35mA per channel, max. ±70mA per block (8 channels) via V _{CC} or GND
Event input	TTL logic
Power up status	All channels as inputs
Connector	37-pin D-Sub female connector

Fig. 91 – Pro II-DIO-32 Rev. E: Specification

Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>P2_DigProg</code>
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Query edges of input channels	<code>P2_Digin_Edge</code>
Set and read back output signals	<code>P2_Digout</code> , <code>P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Get_Digout_Long</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

5.7.2 Pro II-DIO-32-TiCo Rev. E

The digital input/output module Pro II-DIO-32-TiCo Rev. E provides 32 programmable digital input and output channels at TTL levels. The channels can be configured as blocks of 8 bits as inputs or outputs by *ADbasic* instructions. The channels are configured as inputs after power up.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

At the input channels spikes can be suppressed using an adjustable filter. Each channel has its own filter, but filter settings are the same for all channels. After power-up, all filters are disabled.

Different from variant Pro II-DIO-32 Rev. E the module provides in addition the freely programmable *TiCo* processor with 28KiB program memory, 28KiB data memory, and 256MiB external memory. The *TiCo* processor has access to all digital input and output channels. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

From revision E 03 the module may output levels at defined points in time to digital outputs as stand-alone. A FIFO serves as buffer where the user-defined level patterns and points in time are stored.

The FIFO can either be used for edge control of the inputs or for level output at the outputs.

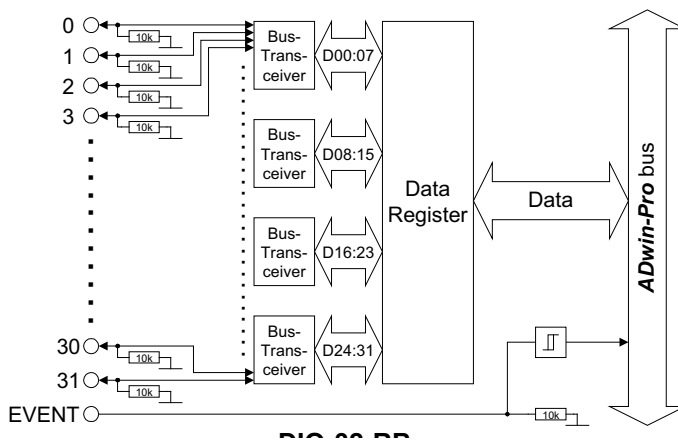


Fig. 92 – Pro II-DIO-32-TiCo Rev. E: Block diagram

Edge monitor

Spike filter

TiCo processor

Timed level output

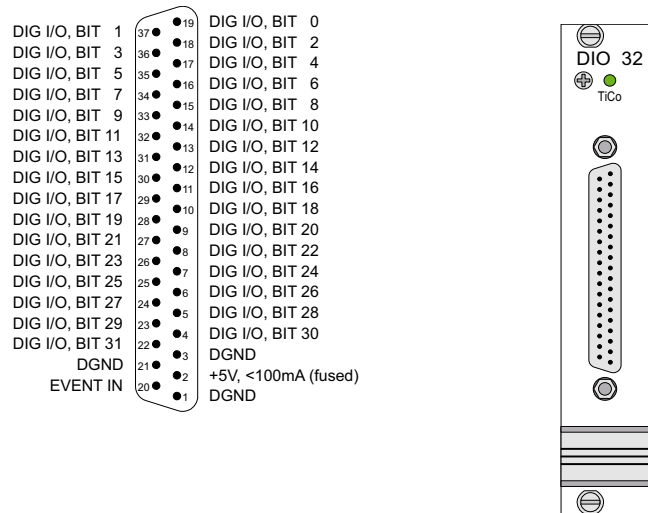


Fig. 93 – Pro II-DIO-32-TiCo Rev. E: Front panel and Pin assignment

Digital inputs	TTL logic
Input/output channels	32; programmable via software as inputs/outputs in blocks of 8
Pull down resistor	10kΩ
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1 μA
I _{IL}	max. 0.01 mA
Voltage range	-0.5V ... +5.5V
Output current	max. ±35mA per channel, max. ±70mA per block (8 channels) via V _{CC} or GND
Event input	TTL logic
Power up status	All channels as inputs
Input / output FIFO	Size: 511 value pairs Frequency: 100MHz
TiCo	Prozessor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal, 256MiByte DRAM external
Connector	37-pin D-Sub female connector

Fig. 94 – Pro II-DIO-32-TiCo Rev. E: Specification

Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>P2_DigProg</code>
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code>

Function	Instructions
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Set spike filter	<code>P2_Digin_Filter_Init</code>
Query edges of input channels	<code>P2_Digin_Edge</code>
Set and read back output signals	<code>P2_Digout, P2_Digout_Bits</code> <code>P2_Digout_Set, P2_Digout_Reset</code> <code>P2_Digout_Long</code> <code>P2_Get_Digout_Long</code>
Set output signals automatically	<code>P2_Digout_FIFO_Clear</code> <code>P2_Digout_FIFO_Empty</code> <code>P2_Digout_FIFO_Enable</code> <code>P2_Digout_FIFO_Read_Timer</code> <code>P2_Digout_FIFO_Start</code> <code>P2_Digout_FIFO_Write</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `DIO32TiCo.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>DigProg</code>
Query input signals	<code>Digin_Long</code>
Monitor edges of input channels	<code>Digin_FIFO_Enable</code> <code>Digin_FIFO_Read</code> <code>Digin_FIFO_Read_Timer</code> <code>Digin_FIFO_Clear</code> <code>Digin_FIFO_Full</code>
Set spike filter	<code>Digin_Filter_Init</code>
Query edges of input channels	<code>Digin_Edge</code>
Set and read back output signals	<code>Digout, Digout_Bits</code> <code>Digout_Set, Digout_Reset</code> <code>Digout_Long</code> <code>Get_Digout_Long</code>
Set output signals automatically	<code>Digout_FIFO_Clear</code> <code>Digout_FIFO_Empty</code> <code>Digout_FIFO_Enable</code> <code>Digout_FIFO_Read_Timer</code> <code>Digout_FIFO_Start</code> <code>Digout_FIFO_Write</code>
Use LED	<code>Check_LED, Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable, Trigger_Event</code> <code>Event_Config</code>

Programming in TiCoBasic

Programming TiCo access

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer, P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start, P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

5.7.3 Pro II-DIO-32/1-TiCo Rev. E

The digital input/output module Pro II-DIO-32/1-TiCo Rev. E provides 32 programmable digital input and output channels at TTL levels. Each of the channels can be configured as input or output by *ADbasic* instructions. The channels are configured as inputs after power up.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

At the input channels spikes can be suppressed using an adjustable filter. Each channel has its own filter, but filter settings are the same for all channels. After power-up, all filters are disabled.

The module provides in addition the freely programmable *TiCo* processor with 28KiB program memory and 28KiB data memory. The *TiCo* processor has access to all digital input and output channels. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The module may output levels at defined points in time to digital outputs as stand-alone. A FIFO serves as buffer where the user-defined level patterns and points in time are stored.

The Fifo can either be used for edge control of the inputs or for level output at the outputs.

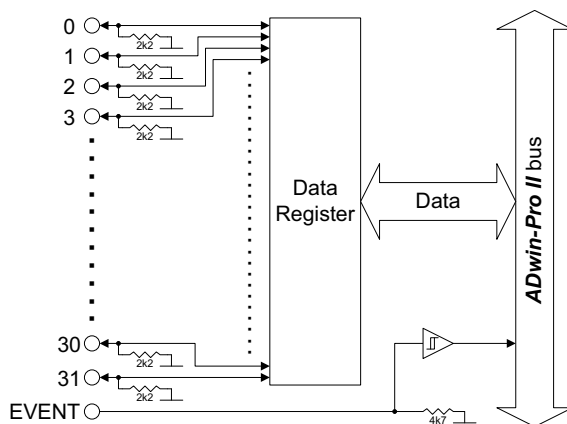


Fig. 95 – Pro II-DIO-32/1-TiCo Rev. E: Block diagram

Edge monitor

Spike filter

TiCo processor

Timed level output

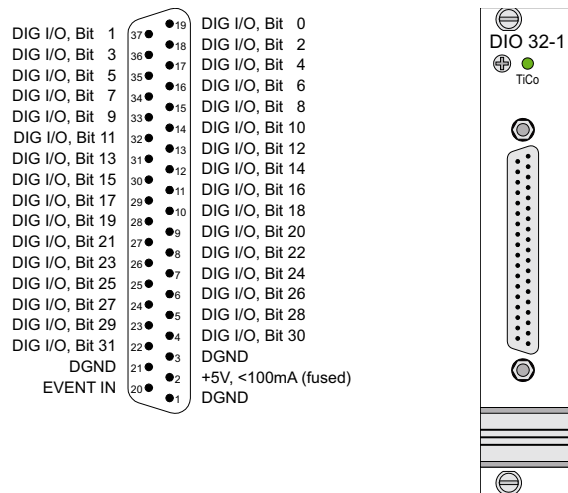


Fig. 96 – Pro II-DIO-32/1-TiCo Rev. E: Front panel and pin assignment

Digital inputs	TTL logic
Input/output channels	32; each programmable via software as input/output
Pull down resistor	10kΩ
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1μA
I _{IL}	max. 0.01mA
Voltage range	-0.5V ... +5.5V
Output current	max. ±32mA per channel via V _{CC} or GND
Event input	TTL logic
Power up status	All channels as inputs
Input / output FIFO	Size: 511 value pairs Frequency: 100MHz
TiCo	Prozessor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal
Connector	37-pin D-Sub female connector

Fig. 97 – Pro II-DIO-32/1-TiCo Rev. E: Specification

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>P2_DigProg</code> <code>P2_DigProg_Bits</code>
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code>

Function	Instructions
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Set spike filter	<code>P2_Digin_Filter_Init</code>
Query edges of input channels	<code>P2_Digin_Edge</code>
Set and read back output signals	<code>P2_Digout, P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Digout_Set</code> <code>P2_Digout_Reset</code> <code>P2_Get_Digout_Long</code>
Set output signals automatically	<code>P2_Digout_FIFO_Clear</code> <code>P2_Digout_FIFO_Empty</code> <code>P2_Digout_FIFO_Enable</code> <code>P2_Digout_FIFO_Read_Timer</code> <code>P2_Digout_FIFO_Start</code> <code>P2_Digout_FIFO_Write</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Config</code> <code>P2_Event_Read</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The folder `C:\ADwin\TiCoBasic\samples_ADwin_ProII` contains additional examples.

The include file `DIO32TiCo.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>DigProg, DigProg_Bits</code>
Query input signals	<code>Digin_Long</code>
Set spike filter	<code>Digin_FIFO_Enable</code> <code>Digin_FIFO_Read</code> <code>Digin_FIFO_Read_Timer</code> <code>Digin_FIFO_Clear</code> <code>Digin_FIFO_Full</code>
Query edges of input channels	<code>Digin_Filter_Init</code>
Set and read back output signals	<code>Digin_Edge</code>
Set output signals automatically	<code>Digout, Digout_Bits</code> <code>Digout_Set, Digout_Reset</code> <code>Digout_Long</code> <code>Get_Digout_Long</code>

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Use LED	<code>Digout_FIFO_Clear</code> <code>Digout_FIFO_Empty</code> <code>Digout_FIFO_Enable</code> <code>Digout_FIFO_Read_Timer</code> <code>Digout_FIFO_Start</code> <code>Digout_FIFO_Write</code>

Use interrupt and event inputs	<code>Check_LED</code> , <code>Set_LED</code>
--------------------------------	---

Setting interrupts and event input	<code>Event_Enable</code> , <code>Trigger_Event</code> <code>Event_Config</code>
------------------------------------	---

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

5.7.4 Pro II-DIO-32-TiCo2 Rev. E

The digital input/output module Pro II-DIO-32-TiCo2 Rev. E provides 32 programmable digital input and output channels at selectable voltage levels. The channels can be configured as blocks of 8 bits as inputs or outputs by *ADbasic* instructions. The channels are configured as inputs after power up.

The nominal voltage level of the digital channels can be set in the range of 1.6V ... 4.7V for groups of 8 channels per software. The table shows typical voltage levels with the appropriate thresholds and output currents.

voltage level	$V_{IL\ max}$	$V_{IH\ min}$	I_{OL}
1.65V	0.6V	1.1V	4mA
2.5V	0.7V	1.7V	8mA
3.3V	0.8V	2.0V	24mA
4.7V	1.4V	3.3V	32mA

Fig. 98 – Pro II-DIO-32-TiCo2 Rev. E: Typical voltage levels

The voltage level of the event input is not adjustable and operates with TTL logic.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 200MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 2048 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

At the input channels spikes can be suppressed using an adjustable filter. Each channel has its own filter, but filter settings are the same for all channels. After power-up, all filters are disabled.

Different from variant [Pro II-DIO-32 Rev. E](#) the module additionally provides the freely programmable *TiCo* processor (type TiCo2) with 128KiB program memory and 512KiB data memory. The *TiCo* processor has access to all digital input and output channels. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The module may output levels at defined points in time to digital outputs as stand-alone. A FIFO serves as buffer where the user-defined level patterns and points in time are stored, maximum 2048 value pairs. The point of time the output is effected can be set with a precision of 5ns.

The FIFO can be used either for edge control of inputs or for output of level patterns at the outputs.

Edge monitor

Spike filter

TiCo processor

Timed level output

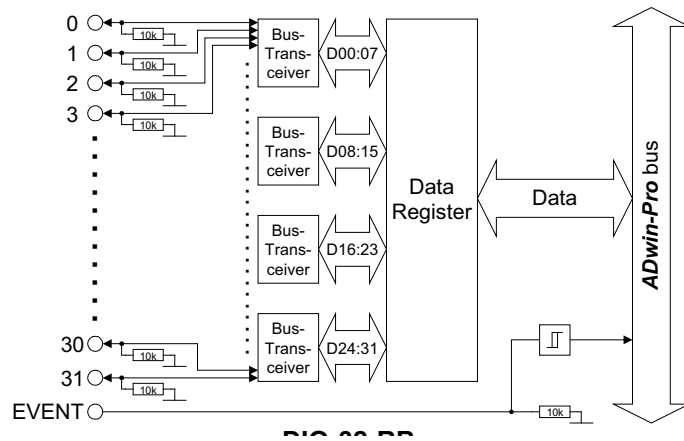


Fig. 99 – Pro II-DIO-32-TiCo2 Rev. E: Block diagram

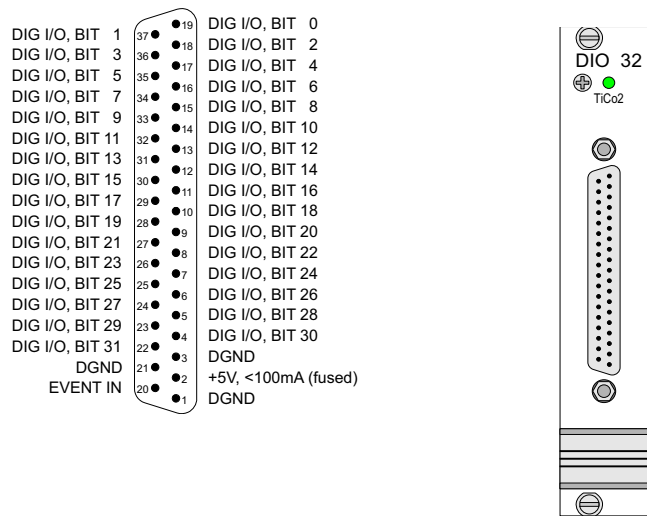


Fig. 100 – Pro II-DIO-32-TiCo2 Rev. E: Front panel and Pin assignment

Input/output channels	32; programmable via software as inputs/outputs in blocks of 8
Pull down resistor	10kΩ
Voltage thresholds	$V_{IL\ max}: 0.3 \times V_{CC}$; $V_{IH\ min}: 0.7 \times V_{CC}$ with V_{CC} = set voltage level
Voltage range	-0.5V ... +5.5V
Output current	max. ±35mA per channel, max. ±70mA per block (8 channels) via V_{CC} or GND
Event input	TTL logic
Power up status	All channels as inputs
Input / output FIFO	Size: 2048 value pairs Frequency: 200MHz
TiCo	Prozessor type: TiCo2 Clock rate: 100MHz Memory size: 128kiB PM internal, 512KiB DM internal
Connector	37-pin D-Sub female connector

Fig. 101 – Pro II-DIO-32-TiCo2 Rev. E: Specification

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>P2_DigProg</code> <code>P2_DigProg_Set_IO_Level</code>
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_Fifo_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Set spike filter	<code>P2_Digin_Filter_Init</code>
Query edges of input channels	<code>P2_Digin_Edge</code>
Set and read back output signals	<code>P2_Digout, P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Get_Digout_Long</code>
Set output signals automatically	<code>P2_Digout_FIFO_Clear</code> <code>P2_Digout_FIFO_Empty</code> <code>P2_Digout_FIFO_Enable</code> <code>P2_Digout_FIFO_Read_Timer</code> <code>P2_Digout_FIFO_Start</code> <code>P2_Digout_FIFO_Write</code>
Synchronize	<code>P2_Sync_All, P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable, P2_Event_Read</code> <code>P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `DIO32TiCo.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>DigProg</code> <code>DigProg_Set_IO_Level</code>
Query input signals	<code>Digin_Long</code>
Monitor edges of input channels	<code>Digin_FIFO_Enable</code> <code>Digin_FIFO_Read</code> <code>Digin_FIFO_Read_Timer</code> <code>Digin_FIFO_Clear</code> <code>Digin_FIFO_Full</code>
Set spike filter	<code>Digin_Filter_Init</code>
Query edges of input channels	<code>Digin_Edge</code>

Programming in ADbasic

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Set and read back output signals	<code>Digout</code> , <code>Digout_Bits</code> <code>Digout_Set</code> , <code>Digout_Reset</code> <code>Digout_Long</code> <code>Get_Digout_Long</code>
Set output signals automatically	<code>Digout_FIFO_Clear</code> <code>Digout_FIFO_Empty</code> <code>Digout_FIFO_Enable</code> <code>Digout_FIFO_Read_Timer</code> <code>Digout_FIFO_Start</code> <code>Digout_FIFO_Write</code>
Use LED	<code>Check_LED</code> , <code>Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable</code> , <code>Trigger_Event</code> <code>Event_Config</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

5.7.5 Pro II-DIO-8-D12 Rev. E

The digital input/output module Pro II-DIO-8-D12 Rev. E provides 20 programmable digital input and output channels, 8 channels with TTL levels, and 12 differential channels. The voltage levels cannot be selected.

The channels with TTL levels can be configured in groups of 4 as inputs or outputs using the instruction `P2_DigProg`. The differential digital channels can be individually configured as inputs or outputs with `P2_DigProg_Bits`. The channels are configured as inputs after power up.

The event input is differential.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 200MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 2048 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

At the input channels spikes can be suppressed using an adjustable filter. Each channel has its own filter, but filter settings are the same for all channels. After power-up, all filters are disabled.

The module provides the freely programmable *TiCo* processor (type TiCo2) with 128KiB program memory and 512KiB data memory. The *TiCo* processor has access to all digital input and output channels. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The module may output levels at defined points in time to digital outputs as stand-alone. A FIFO serves as buffer where the user-defined level patterns and points in time are stored, maximum 2048 value pairs. The point of time the output is effected can be set with a precision of 5ns.

The FIFO can be used either for edge control of inputs or for output of level patterns at the outputs.

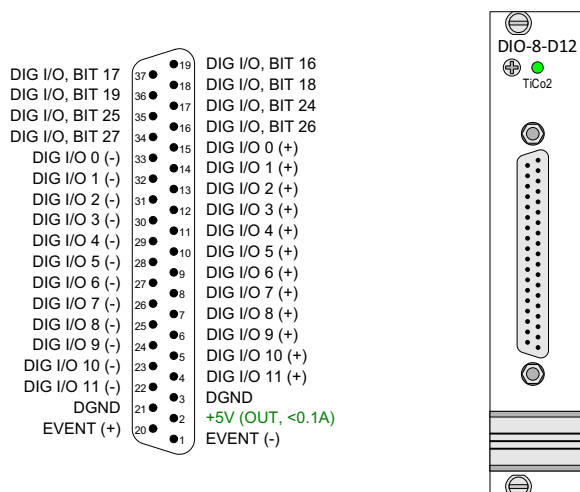


Fig. 102 – Pro II-DIO-8-D12 Rev. E: Front panel and pin assignment

Input/output channels	8 TTL channels (5V) 12 differential channels
-----------------------	---

Fig. 103 – Pro II-DIO-8-D12 Rev. E: Specification

Edge monitor

Spike filter

TiCo processor

Timed level output

Input resistance	TTL inputs: 10 kΩ, pull-down Diff. inputs: 120 Ω
Input/output levels TTL	Voltage range 0 V ... +5 V V _{IH} min. 2 V V _{IL} max. 0,8 V I _{IH} max. 1 μA
Input/output levels differential	-0.5V ... +5.5V compatible to RS422/485 (5V differential, 120 Ω bus terminating resistor)
Output current	max. ±35mA per channel, max. ±70mA per block (4 channels) via V _{CC} or GND
Event input	differential
Power up status	All channels as inputs
Input / output FIFO	Size: 2048 value pairs Frequency: 200MHz
TiCo	Prozessor type: TiCo2 Clock rate: 100MHz Memory size: 128kiB PM internal, 512KiB DM internal
Connector	37-pin D-Sub female connector

Fig. 103 – Pro II-DIO-8-D12 Rev. E: Specification

Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>P2_DigProg</code> <code>P2_DigProg_Bits</code>
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Set spike filter	<code>P2_Digin_Filter_Init</code>
Query edges of input channels	<code>P2_Digin_Edge</code>
Set and read back output signals	<code>P2_Digout</code> , <code>P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Get_Digout_Long</code>
Set output signals automatically	<code>P2_Digout_FIFO_Clear</code> <code>P2_Digout_FIFO_Empty</code> <code>P2_Digout_FIFO_Enable</code> <code>P2_Digout_FIFO_Read_Timer</code> <code>P2_Digout_FIFO_Start</code> <code>P2_Digout_FIFO_Write</code>

Function	Instructions
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `DIO32TiCo.inc` contains instructions for the following functions:

Function	Instructions
Configure input/outputs	<code>DigProg</code> <code>DigProg_Bits</code>
Query input signals	<code>Digin_Long</code>
Monitor edges of input channels	<code>Digin_FIFO_Enable</code> <code>Digin_FIFO_Read</code> <code>Digin_FIFO_Read_Timer</code> <code>Digin_FIFO_Clear</code> <code>Digin_FIFO_Full</code>
Set spike filter	<code>Digin_Filter_Init</code>
Query edges of input channels	<code>Digin_Edge</code>
Set and read back output signals	<code>Digout</code> , <code>Digout_Bits</code> <code>Digout_Set</code> , <code>Digout_Reset</code> <code>Digout_Long</code> <code>Get_Digout_Long</code>
Set output signals automatically	<code>Digout_FIFO_Clear</code> <code>Digout_FIFO_Empty</code> <code>Digout_FIFO_Enable</code> <code>Digout_FIFO_Read_Timer</code> <code>Digout_FIFO_Start</code> <code>Digout_FIFO_Write</code>
Use LED	<code>Check_LED</code> , <code>Set_LED</code>
Use interrupt and event inputs	<code>Event_Enable</code> , <code>Trigger_Event</code> <code>Event_Config</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

5.7.6 Pro II-OPT-16 Rev. E

The input module Pro II-OPT-16 Rev. E provides 16 channels of optically isolated digital inputs. The input voltage range can be set by jumpers (5V, 12V, 24V). The default setting of the input voltage range is 24V. The switching time of only 100ns allows the sampling of high-speed digital inputs.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

Each channel is optically isolated from the system circuitry and from the other inputs. The event-input is optically isolated from the system as well.

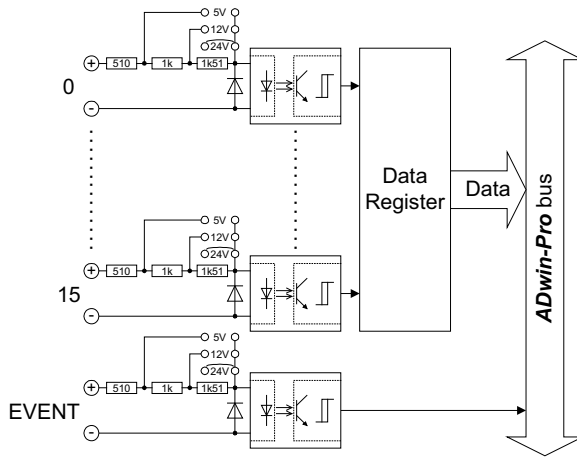


Fig. 104 – Pro II-OPT-16 Rev. E: Block diagram

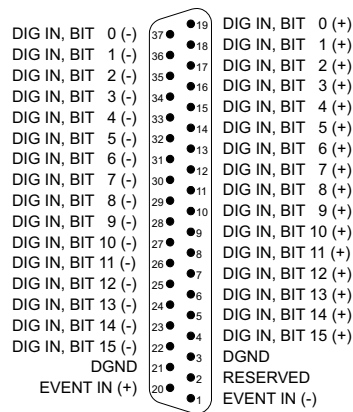


Fig. 105 – Pro II-OPT-16 Rev. E: Pin assignment

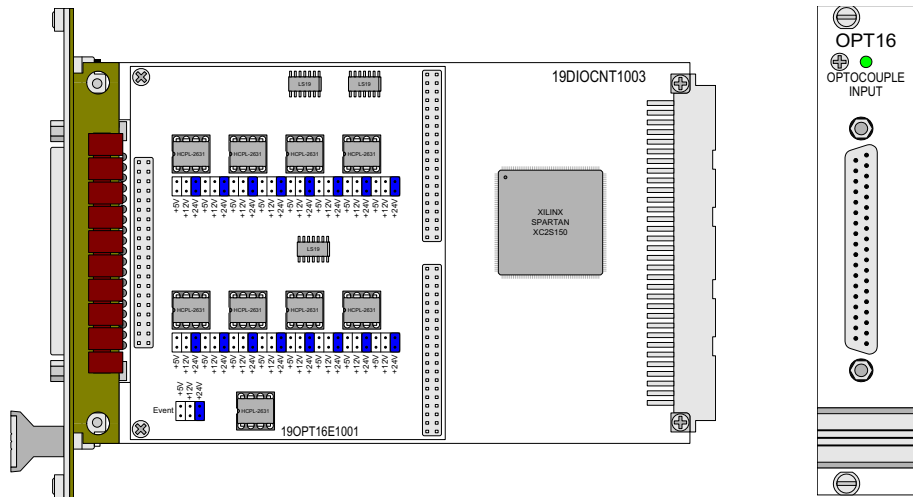


Fig. 106 – Pro II-OPT-16 Rev. E: Board and front panel

Input channels	16		
Event inputs	1		
Input current	typ. 3.5mA / max. 7.5mA		
Input voltage range (selectable via jumpers)	0...5V	0...12V	0...24V
Switching threshold for 0-low	0...0.8V	0...1.6V	0...3.2V
Switching threshold for 1-high	4.5...5V	10...12V	20...24V
Input over-voltage	-5V ... 8V	-5V ... 16V	-5V ... 30V
Switching time	100ns		
Input FIFO	Size: 511 value pairs Frequency: 100MHz		
Isolation	42V channel-channel / channel-GND		
Connector	37-pin D-Sub female connector		

Fig. 107 – Pro II-OPT-16 Rev. E: Specification

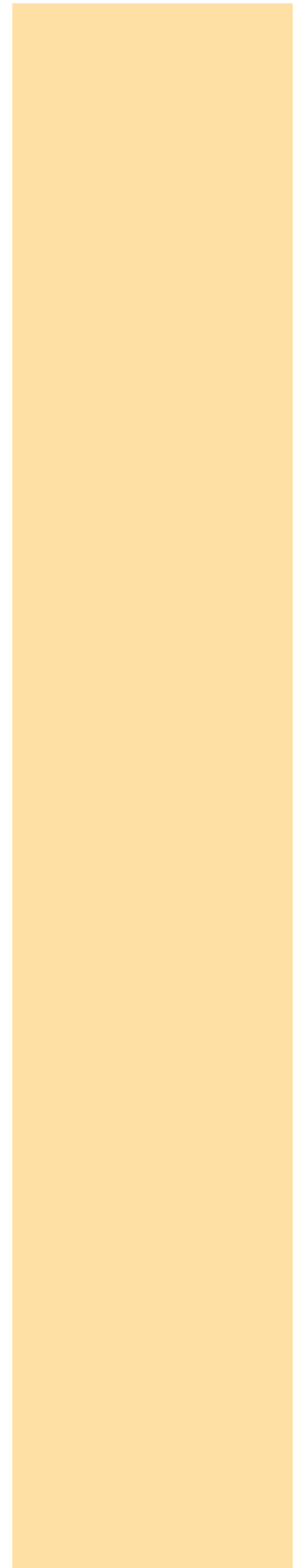
Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Query input signals	<code>P2_Digin_Edge</code> <code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Dig_Read_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>

Function	Instructions
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>



5.7.7 Pro II-OPT-32-24V Rev. E

The input module Pro II-OPT-32-24V Rev. E provides 32 channels of optically isolated digital inputs.

The input voltage range is fixed to 24V, which is also true for the event input. The switching time of only 100ns allows the sampling of high-speed digital inputs.

The module can automatically monitor the edges of input channels, which is performed with a frequency of 100MHz. With every change, the current input levels are saved together with a time stamp in a FIFO; up to 511 of those value pairs (input level and time stamp) can be stored. The FIFO data can be read and processed.

In addition, one can query whether a positive or negative edge has occurred at the input channels.

Each channel is optically isolated from the system circuitry and from the other inputs. The event-input is optically isolated from the system as well.

The channels are single ended and share the same ground potential (ext. GND), which has to be provided on the D-Sub connector.

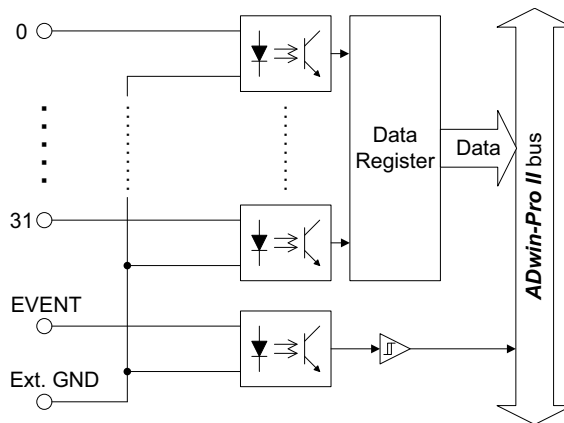


Fig. 108 – Pro II-OPT-32-24V Rev. E: Block diagram

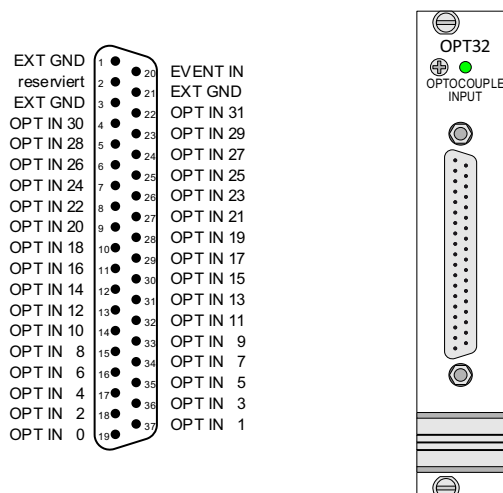


Fig. 109 – Pro II-OPT-32-24V Rev. E: Pin assignment and front panel

Input channels	32 single ended
Event inputs	1
Input current	typ. 3.5mA / max. 7.5mA
Input voltage range (ordering option)	0...24V
Switching threshold for 0-low	0...3.2V
Switching threshold for 1-high	20...24V
Input over-voltage	-5V ... 30V
Switching time	100ns
Input FIFO	Size: 511 value pairs Frequency: 100MHz
Isolation	42V channel-channel / channel-GND
Connector	37-pin D-Sub female connector

Fig. 110 – Pro II-OPT-32-24V Rev. E: Specification

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Query input signals	<code>P2_Digin_Edge</code> <code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Dig_Read_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_Fifo_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

The instructions are described in the *ADbasic* online help and in the Pro Software manual.

Programming

5.7.8 Pro II-REL-16 Rev. E

The Pro II-REL-16 Rev. E output module provides 16 isolated relay outputs. Each channel is isolated from system circuitry and other output channels. The event-output is optically isolated from the system circuitry.

The module is equipped with normally open contacts, as an option also normally closed contacts are available.

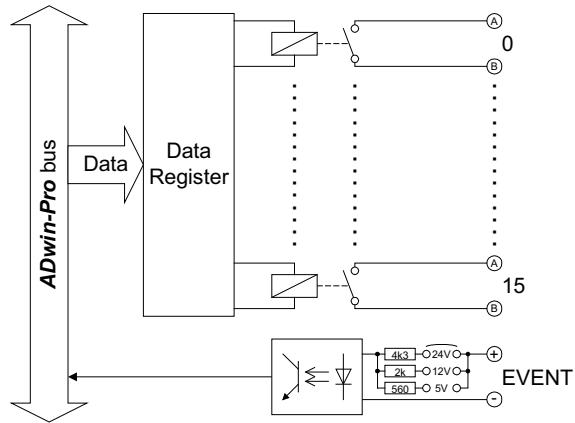


Fig. 111 – Pro II-REL-16 Rev. E: Block diagram

RELAY 0 A	● 37	● 19	RELAY 0 B
RELAY 1 A	● 36	● 18	RELAY 1 B
RELAY 2 A	● 35	● 17	RELAY 2 B
RELAY 3 A	● 34	● 16	RELAY 3 B
RELAY 4 A	● 33	● 15	RELAY 4 B
RELAY 5 A	● 32	● 14	RELAY 5 B
RELAY 6 A	● 31	● 13	RELAY 6 B
RELAY 7 A	● 30	● 12	RELAY 7 B
RELAY 8 A	● 29	● 11	RELAY 8 B
RELAY 9 A	● 28	● 10	RELAY 9 B
RELAY 10 A	● 27	● 9	RELAY 10 B
RELAY 11 A	● 26	● 8	RELAY 11 B
RELAY 12 A	● 25	● 7	RELAY 12 B
RELAY 13 A	● 24	● 6	RELAY 13 B
RELAY 14 A	● 23	● 5	RELAY 14 B
RELAY 15 A	● 22	● 4	RELAY 15 B
DGND	● 21	● 3	DGND
EVENT IN (+)	● 20	● 2	RESERVED
		● 1	EVENT IN (-)

Fig. 112 – Pro II-REL-16 Rev. E: Pin assignment

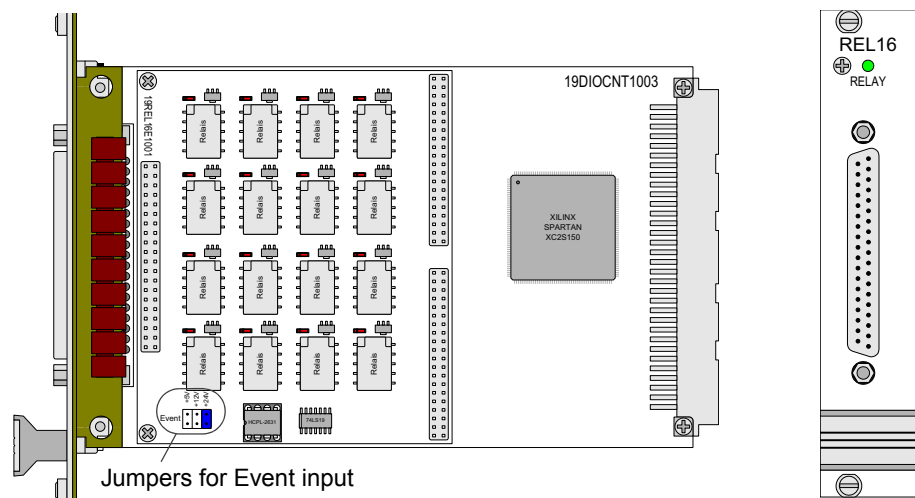


Fig. 113 – Pro II-REL-16 Rev. E: Board and front panel

Output channels	16
Switch voltage	30V AC/DC Maximum
Switch current	max. 500mA per channel
Contact	1 per channel, normally open (optional: normally closed)
Operate time	4ms
Release time	3ms
Bounce time	2ms
Event inputs	1
Isolation	42V channel-channel / chan- nel-GND
Event input voltage	5V, 12V, 24V (selectable via jumpers)
Power up status	low (normally open contact: open / normally closed contact: closed)
Connector	37-pin D-Sub female connector

Fig. 114 – Pro II-REL-16 Rev. E: Specification

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Set and read back output signals	<code>P2_Digout</code> , <code>P2_Digout_Long</code> <code>P2_Digout_Bits</code> <code>P2_Get_Digout_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Dig_Write_Latch</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

Programming

5.7.9 Pro II-TRA-16 Rev. E

The output module Pro II-TRA-16 Rev. E provides 16 channels of isolated transistor outputs. There are 2 module variants:

- Pro II-TRA-16 Rev. E: The outputs switch to V_{CC} .
- Pro II-TRA-16-G Rev. E: The outputs switch to ground.

The switching voltage V_{CC} has to be provided by an external power supply. The channels as well as the event-input are optically isolated from system circuitry.

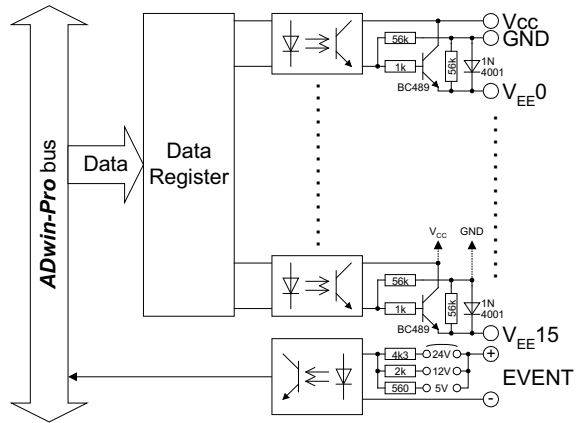


Fig. 115 – Pro II-TRA-16 Rev. E: Block diagram

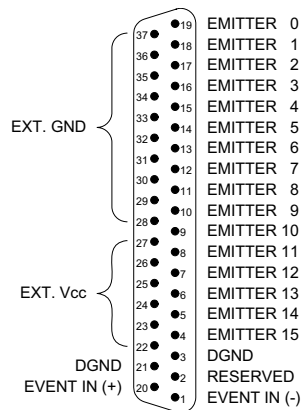


Fig. 116 – Pro II-TRA-16 Rev. E: Pin assignment

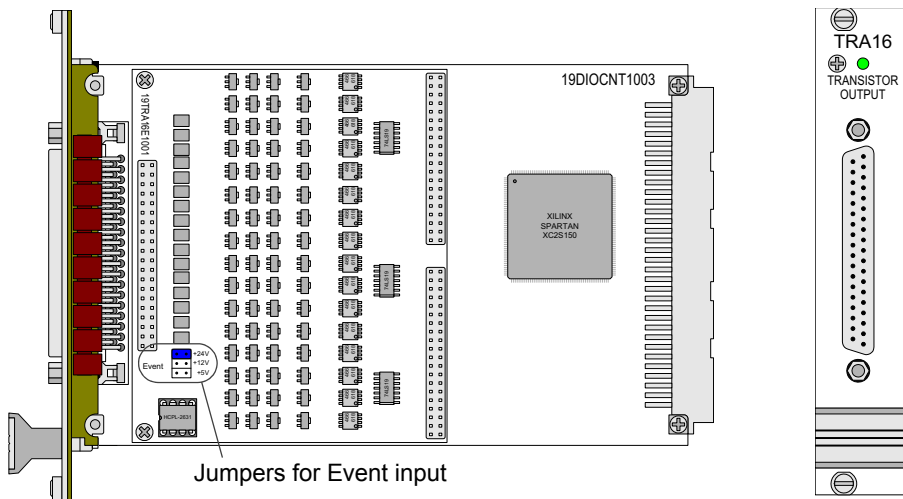


Fig. 117 – Pro II-TRA-16 Rev. E: Board and front panel

Output channels	16
Switching voltage V_{CC}	5...30V DC with external power supply
Switching current	Pro II-TRA-16: 200mA max. per channel Pro II-TRA-16-G: 100mA max. per channel
Voltage drop	0.5V
Switching time	2.5 μ s
Event input	1
Isolation	42V channel-channel / channel-GND
Event input voltage	5V, 12V, 24V (selectable via jumpers)
Power up status	low (GND external)
Connector	37-pin D-Sub female connector

Fig. 118 – Pro II-TRA-16 Rev. E: Specification

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Set and read back output signals	<code>P2_Digout</code> , <code>P2_Digout_Long</code> <code>P2_Digout_Bits</code> <code>P2_Get_Digout_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Dig_Write_Latch</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

Programming

5.7.10 Pro II-PWM-16 Rev. E

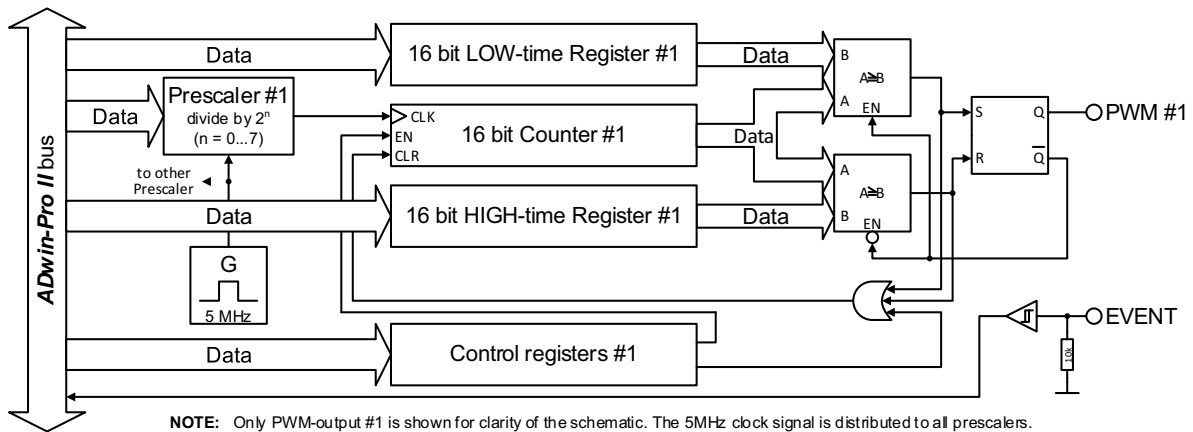
The output module Pro II-PWM-16 Rev. E generates pulse width modulated signals (PWM signals) at 16 outputs. Each (PWM) signal can be configured individually via software; that means, they can be configured separately.

The module is available as version Pro II-PWM-16-I Rev. E, too. On this module, the outputs are optically isolated against the system circuit and against each other. The event input, too, is optically isolated from the system circuit.

The output signals are clocked with a reference clock speed of 100MHz.

The lowest output frequency is about 0.6Hz.

The highest output frequency where the duty cycle can be still defined in 1%-steps, is 1000kHz.



above: Pro II-PWM-16 Rev. E, below: Pro II-PWM-16-I Rev. E

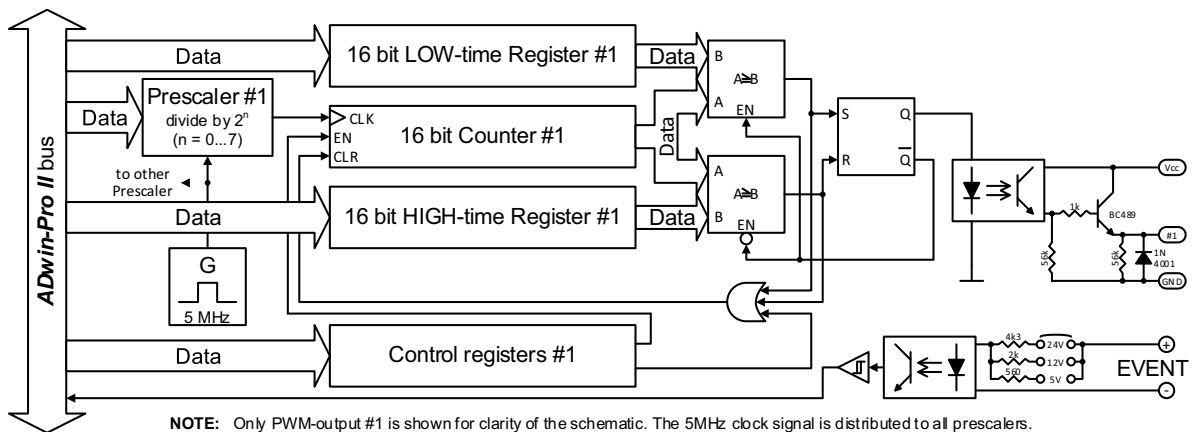


Fig. 119 – Pro II-PWM-16 Rev. E: Block diagram

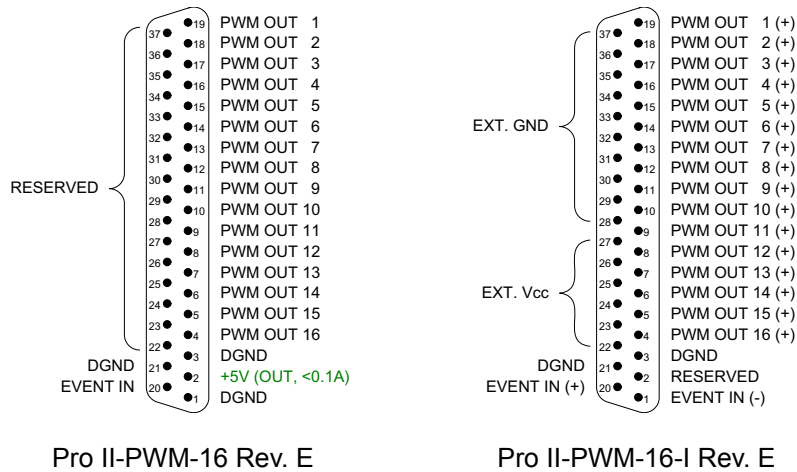


Fig. 120 – Pro II-PWM-16 Rev. E: Pin assignment

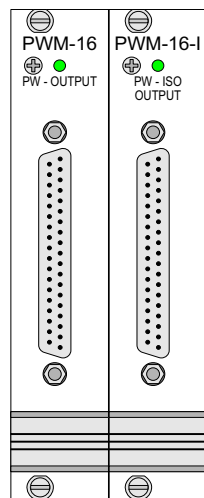


Fig. 121 – Pro II-PWM-16-I Rev. E: Front panels

Output channels	16 PWM channels
Outputs	TTL
Counter / register width	32 bit
Reference clock	100MHz
Event input	1 input, positive TTL
Connector	37-pin D-Sub female connector
Pro II-PWM-16 Rev. E	
V_{OH}	2.4V min.
V_{OL}	0.8V max.
Output current	5mA per channel max.
Isolation	No (see page 132)
Pro II-PWM-16-I Rev. E	
Output voltage	5...30V DC via external power supply
Output current	100mA max. per channel
Voltage drop	0.5V max.

Fig. 122 – Pro II-PWM-16 Rev. E: Spezifikation

Programming

Switching time	without load: 3µs with load: 1...2µs
Event input voltage	5V, 12V, 24V (selectable via jumpers)
Isolation	42V channel-channel / channel-GND

Fig. 122 – Pro II-PWM-16 Rev. E: Spezifikation

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Initialize module	<code>P2_PWM_Init</code> <code>P2_PWM_Standby_Value</code> <code>P2_PWM_Enable</code> <code>P2_PWM_Reset</code>
Output and read back PWM signals	<code>P2_PWM_Write_Latch</code> <code>P2_PWM_Latch</code> <code>P2_PWM_Get_Status</code>
Synchronize	<code>P2_Sync_All</code> , <code>P2_Sync_Enable</code> <code>P2_Sync_Stat</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> <code>P2_Event_Config</code> <code>P2_Event_Read</code>

5.7.11 Pro II-COMP-16 Rev. E

The digital input module Pro II-COMP-16 Rev. E provides 16 channels with each a comparator. Incoming signals are compared to an analog threshold, which is set via software. Depending on whether the signal exceeds or falls below the threshold an appropriate bit is set to 1 or 0.

In addition, the following features are available: symmetrical hysteresis around the threshold, holding of comparator values, a free definable hysteresis (by use of 2 channels with different thresholds) and filtering of spikes.

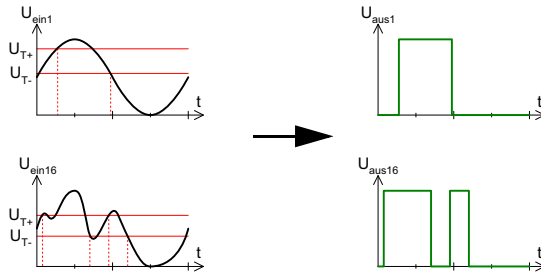


Fig. 123 – Pro II-COMP-16 Rev. E: Comparator principle with hysteresis

The input voltage range and the setting range for the threshold is -1V ... +30V.

You set the threshold digitally via software; from this, a DAC creates an analog signal with 16-bit resolution to be compared with the comparator. On the module there are 4 DAC, so each 4 channels work with the same threshold. The assignment of DAC to channels is as follows:

DAC	Channels
1	1, 3, 5, 7
2	2, 4, 6, 8
3	9, 11, 13, 15
4	10, 12, 14, 16

The comparators run with 50MHz.

The module can automatically monitor the edges of input channels (= signals after comparators), which is performed with a frequency of 100MHz. A positive edge therefore means exceeding, a negative edge falling below the threshold. With every change, the current level status are saved together with a time stamp in a FIFO; up to 511 of those value pairs (level status and time stamp) can be stored. The FIFO data can be read and processed.

The following features can be enabled additionally: Holding a comparator value, setting a (standard) hysteresis around a threshold, and setting a free hysteresis by use of 2 channels with different thresholds.

– Holding a comparator value

As soon as an incoming analog value reaches the threshold, the comparator is disabled for a defined hold time and enabled again afterwards. The hold time can be set to the following values: 1 μ s, 10 μ s, or 100 μ s.

You can set comparator hold for groups of 4 channels (1...4, 5...8, 9...12, 13...16), but not for single channels. Comparator hold and standard hysteresis cannot be combined.

Edge detection

Additional features

- Standard hysteresis
The hysteresis refers to the set threshold. The channel bit is set to 1 if the threshold plus hysteresis is exceeded, and is reset to 0 if the signal falls below the threshold minus hysteresis.
The hysteresis can be set to the following values. Until Rev. E02: 0mV, ± 17 mV, ± 33 mV, ± 55 mV, or ± 100 mV; since Rev. E03: 0mV, ± 34 mV, ± 66 mV, ± 110 mV, ± 200 mV.
You can the hysteresis for groups of 4 channels (1...4, 5...8, 9...12, 13...16), but not for a single channel. Comparator hold and standard hysteresis cannot be combined.
- Free hysteresis
To use other, especially a greater hysteresis than standard hysteresis, you have to combine a channel pair (1,2 / ... / 15, 16). The signal must be connected to both channels. The comparator bit of the channel pair is set to 1, if the signal exceeds the thresholds of both channels. Accordingly, the comparator bit is set to 0 if the signal falls below the thresholds of both channels.
The channels being used for free hysteresis can additionally be combined with comparator hold or standard hysteresis.
- Comparator filter
You can enable a filter, which is located behind the comparators, in order to suppress single spikes. The number of spikes should be small compared to the pulse width of the signal.
The filter settings apply to all channels. Each channel has its own filter. After power-up all filters are disabled.
The filter duration can be set in a range from 20ns to 1.31 ms and should be somewhat longer than the expected width of spikes.
The filter duration determines how long a comparator output must be 1 to set the channel bit to 1. The timing filter increments in clocks of 20ns until the target value is reached; if the comparator output is 0 the counter is decremented. In order to fall below a threshold, the same procedure is processed in reverse.
The filter can be combined with standard hysteresis and free hysteresis, but is not suited for combination with comparator hold.
The following figure shows filtering of 2 example signals, the left one without, the right one with a spike. The filter delays the edge of the resulting signal by the filter duration plus the double length of the spike's length.

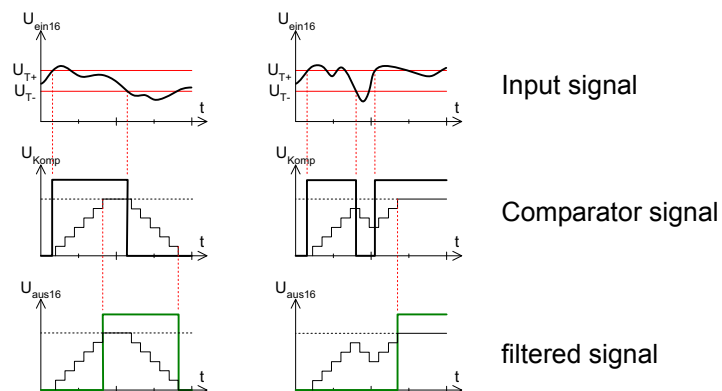


Fig. 124 – Pro II-COMP-16 Rev. E: Comparator filter

You read the comparator bits with instructions for digital inputs like `P2_Digin_Long`.

With comparator hold / standard hysteresis, the bits are assigned to the channels as follows:

Bit no.	31...16	15	14	...	2	1	0
Channel no.	-	16	15	...	3	2	1

The comparator bits for the channel pairs of the free hysteresis are the bits 16...23; the bits are assigned to the channel pairs as follows:

Bit no.	31...24	23	22	...	17	16	15...0
Channel pair.	-	15, 16	13, 14	...	3, 4	1, 2	-

Channel 16 does also work as event input, so there is no separate pin for this function. The signal after the comparator is used as event signal.

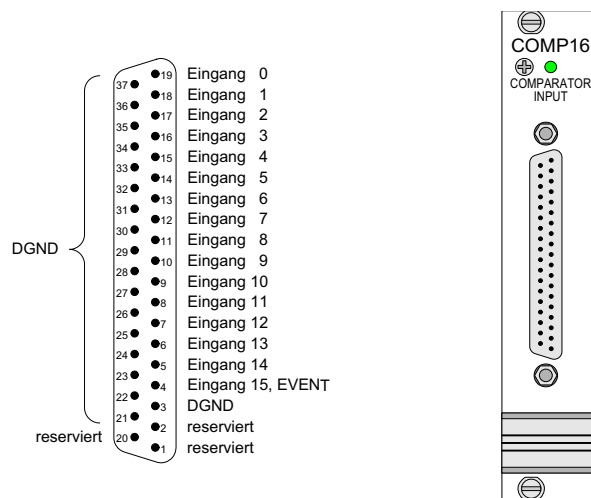


Fig. 125 – Pro II-COMP-16 Rev. E: Pin assignment and front cover

Input channels	16 single-ended inputs each with its own comparator Threshold adjustable for groups of 4 channels each
Event inputs	1
Spannungsfestigkeit	-2V...+32V
Input voltage range	-1V ... +30V
Clock rate	50MHz
Resolution of threshold	16 Bit
DAC settling time	30µs
Input Fifo	Size: 511 value pairs Frequency: 100MHz
Connector	37-pin D-Sub female connector

Fig. 126 – Pro II-COMP-16 Rev. E: Specification

Read comparator bits

Event input

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Do mode and comparator settings	<code>P2_Comp_Init</code> <code>P2_Comp_Filter_Init</code> <code>P2_Comp_Set</code>
Query comparator bits	<code>P2_Digin_Edge</code> <code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code>
Monitor edges of input channels	<code>P2_Digin_FIFO_Enable</code> <code>P2_Digin_FIFO_Read</code> <code>P2_Digin_FIFO_Read_Fast</code> <code>P2_Digin_FIFO_Read_Timer</code> <code>P2_Digin_FIFO_Clear</code> <code>P2_Digin_FIFO_Full</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Use interrupt and event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Read</code> <code>P2_Event_Config</code>

5.7.12 Pro II-CNT-x Rev. E

The module Pro II-CNT-x Rev. E provides 4 configurable multi-purpose counter blocks. Each counter block contains two 32-bit counters: First an up/down counter or four edge evaluation for connection of encoders. Second, a PWM counter to evaluate high and low times, duty cycle, or frequency. Both counters of a block can be operated in parallel.

The following module variants are available:

- *Pro II-CNT-T*: Counter inputs with TTL logic. The module is equipped with a *TiCo* processor.
- *Pro II-CNT-D*: Differential counter inputs; in addition, the counter contains 2 SSI decoders and a *TiCo* processor.
- *Pro II-CNT-I*: Counter inputs are optically isolated. The module is equipped with a *TiCo* processor.

The counter inputs are optically isolated from the system circuit. The event input, too, is optically isolated from the system circuit.

The input voltage range of the counter and event inputs can be set by jumpers to 0...5V, 0...12V or 0...24V. The default setting is 0...24V.

The *TiCo* processor is freely programmable and has access to all counter functions. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The up/down counter of a block can be operated in 2 modes:

- Clock / direction (CLK and DIR signals)

A negative edge at the CLK input is the counting impulse for the 16-bit counter. The DIR signal sets the counting direction, TTL high means a count-up, TTL low means a count-down.

You can invert the signals at the inputs CLK and DIR via software (instruction **P2_Cnt_Mode**) and thus change both the triggering edge and the counting direction.

You can latch the counter values program-controlled or you can influence the counter by an external CLR/LATCH signal.

Depending on the programming, a rising edge at input CLR/LATCH can either clear the counter values (CLR) or latch the counter values (LATCH). This function will only be effective when it is released by the instruction **P2_Cnt_Mode**.

Using the latch function, you can determine the frequency of the measurement by getting the difference of two read latch values, because this difference defines the number of pulses between the two reading processes.

- Four edge evaluation (A and B signals)

The four edge evaluation changes the signals (which should be 90° phase-shifted) of a connected incremental encoder at the inputs A and B to CLK and DIR signals. For this you have to program the inputs correspondingly (see "*ADwin-Pro* System Description, Programming in *ADbasic*").

Since every edge of the A and B signals generates a count impulse, the resolution is increased by factor 4. If the encoder has a reference signal,

Up/down counter

PWM counter

**Exception:
evaluate PWM registers
on your own**

it can be used to clear or latch the counter (after release of the CLR or LATCH input). The counter is cleared when the signals A, B and CLR are on logic "1" (software-selectable: clear, when only the CLR signal is on logic "1").

The PWM counter of the counter block analyzes the signals at the PWM inputs. You can set the input pin as well as the triggering edge with the instruction **P2_Cnt_Mode**.

Please note: On the module *Pro II-CNT-T* the PWM input pins A and B are tied to four edge evaluation, the pins CLK and DIR to clock / direction evaluation.

Via standard instructions the following data can be read directly:

- frequency and duty cycle (with **P2_Cnt_Get_PW**)
- high and low time (with **P2_Cnt_Get_PW_HL**)

There are several registers assigned to each PWM counter being described below. If PWM counters are evaluated with standard instructions mentioned above, no further knowledge is required about PWM registers. Use the evaluation with PWM registers for special solutions only.

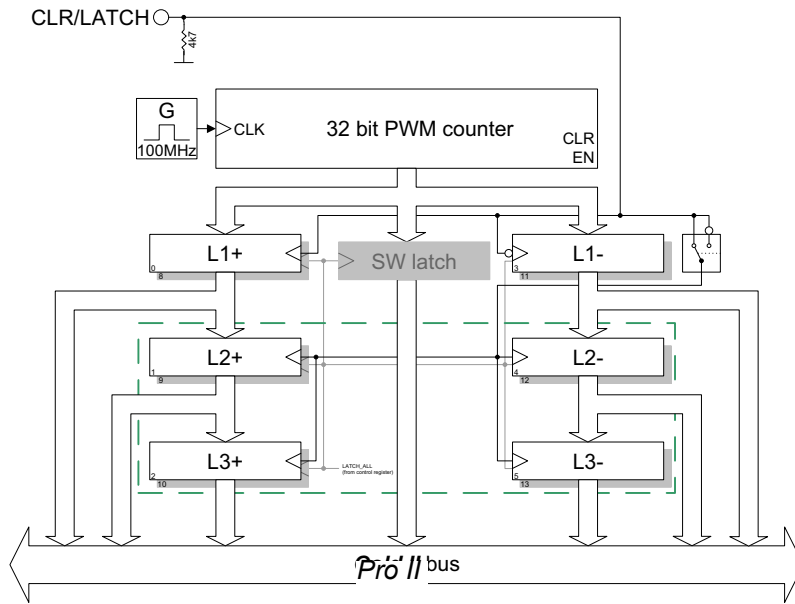
In order to evaluate PWM signals, the counter values of the current and the 2 preceding counter values are stored in latch registers, both for rising and falling edges. In addition, there is a "shadow register" for each of these 6 registers: the values of all 6 registers can at the same moment be copied to the 6 shadow registers to be available for calculations.

Register	Latch	Shadow register
Latch 1 for positive edges (current)	L1+	SL1+
Latch 2 for positive edges	L2+	SL2+
Latch 3 for positive edges	L3+	SL3+
Latch 1 for negative edges (current)	L1-	SL1-
Latch 2 for negative edges	L2-	SL2-
Latch 3 for negative edges	L3-	SL3-

The register values are changed with any edge like this:

- Rising edge:
 - Copy counter value to L1+
 - If rising edge is set as reference edge:
 - Copy register L2+ to L3+
 - Copy register L1+ to L2+
 - Copy register L2- to L3-
 - Copy register L1- to L2-
- Falling edge:
 - Copy counter value to L1-
 - If falling edge is set as reference edge:
 - Copy register L2- to L3-
 - Copy register L1- to L2-
 - Copy register L2+ to L3+
 - Copy register L1+ to L2+

In addition, there is a single latch register where the counter value is copied by software (instruction **P2_Cnt_PW_Latch**).



For any evaluation the PWM registers of levels 2 and 3 are used. First, the register values are copied to the shadow registers with **P2_Cnt_Sync_Latch** and then evaluated.

The calculation depends on the set reference edge:

Parameter	rising edge	falling edge
diagram		
period	$T = L2+ - L3+$	$T = L2- - L3-$
high time	$t_H = L3- - L3+$	$t_H = L2- - L3+$
low time	$t_L = T - t_H = L2+ - L3-$	$t_L = T - t_H = L3+ - L3-$
frequency	$f = 1 / T = 1 / (L2+ - L3+)$	$f = 1 / T = 1 / (L2- - L3-)$
duty cycle	$g = t_H / T = (L3- - L3+) / (L2+ - L3+)$	$g = t_H / T = (L2- - L3+) / (L2- - L3-)$

The module variant *Pro II-CNT-D* provides 2 decoders to connect incremental encoders with SSI interface. The signals are differential and have RS422/485 levels.

A decoder either reads out an individual value (on request) or continuously provides the current value.

Decoder properties can be set via software:

- Clock rates are set with **SSI_Set_Clock** from 6.1kHz ... 12.5MHz.
- Resolution is set with **SSI_Set_Bits** up to 32 bit.

```
REM Exmample: conversion from Gray code into binary code
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

SSI decoder



Event input

```

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
    
```

This input, as far as it has been released, can start an externally triggered **ADbasic** process.

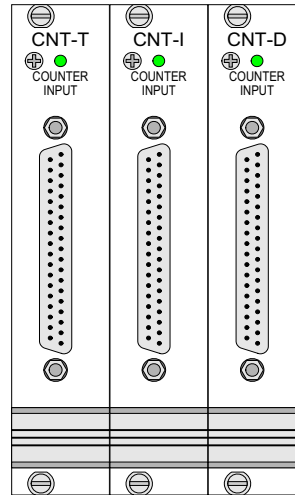


Fig. 127 – Pro II-CNT-x Rev. E: Front panel

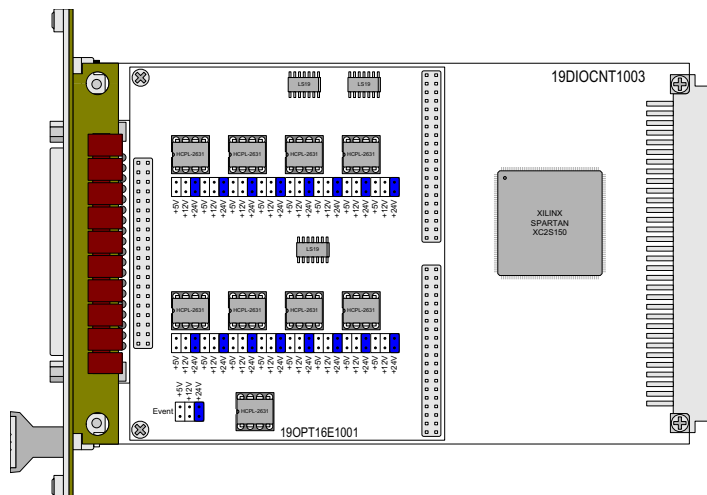
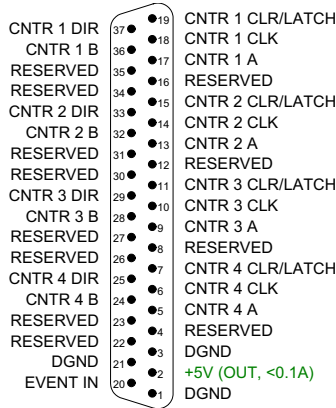
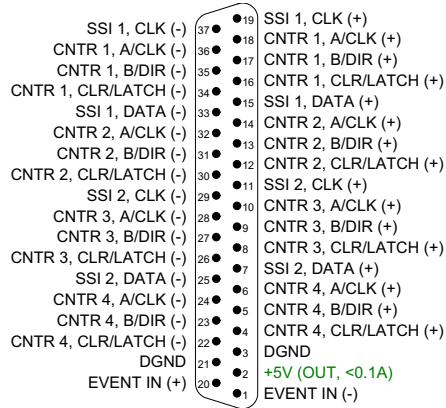


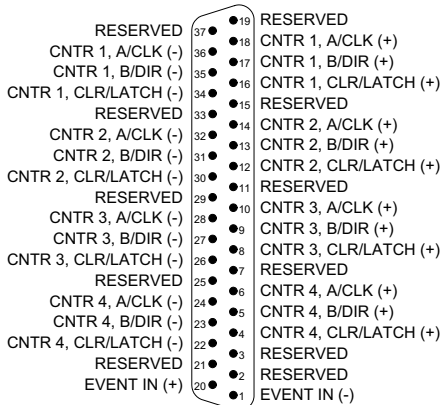
Fig. 128 – Pro II-CNT-I Rev. E: Board with jumpers



Pin assignment Pro II-CNT-T



Pin assignment Pro II-CNT-D



Pin assignment Pro II-CNT-I

Counter	4 multi-purpose counters Cnt-D: 2 SSI decoders in addition
Counter resolution	32 bit
Event input	1
Reference clock	50MHz
Clock frequency four edge evaluation	12.5MHz max. (at 90° phase-shift of the signals)
Clock frequency up/down counter	CNT-T: 25MHz max. CNT-I, CNT-D: 15MHz max.
Reference frequency PWM analysis	100MHz
Connector	37-pin D-Sub female connector
TiCo	Prozessor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal
Power consumption	CNT-T approx. 150mA CNT-D approx. 200mA CNT-I approx. 200mA

Fig. 129 – Pro II-CNT-x Rev. E: General specification

Pro II-CNT-T			
Input / output level	TTL logic		
Event input	TTL logic		
Isolation	none		
Pro II-CNT-D			
Input / output level	compatible to RS422/485 (5V differential, 120 Ω bus terminating resistor)		
Event input	1 differential (singled operation possible)		
Clock frequency SSI decoder (CLK)	6kHz ... 12.5MHz		
Isolation	none		
Pro II-CNT-I			
Input current	typ.3.5mA / max. 7.5mA		
input voltage range (selectable via jumpers)	0...5V	0 ... 12V	0...24V
reliable switching threshold ¹ for 0 (low)	0...0.8V	0...1.6V	0...3.2V
reliable switching threshold ¹ for 1 (high)	4.5...5V	10...12V	20...24V
Input over-voltage	8V	16V	30V
Negative voltage	-5V for all ranges		
Switching time	100ns		
Isolation	42V channel-channel / channel-GND		

Fig. 130 – Pro II-CNT-x Rev. E: Specification

Programming

Programming in ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure counters	<code>P2_Cnt_Enable</code> , <code>P2_Cnt_Mode</code>
Use counters	<code>P2_Cnt_Clear</code> , <code>P2_Cnt_Get_Status</code> <code>P2_Cnt_Latch</code> <code>P2_Cnt_Read</code> , <code>P2_Cnt_Read4</code> <code>P2_Cnt_Read_Latch</code> <code>P2_Cnt_Read_Latch4</code>
Use PWM counters	<code>P2_Cnt_PW_Enable</code> , <code>P2_Cnt_PW_Latch</code> <code>P2_Cnt_Get_PW</code> , <code>P2_Cnt_Get_PW_HL</code> , <code>P2_Cnt_Read_Int_Register</code>
Use SSI decoders (CNT-D only)	<code>P2_SSI_Mode</code> , <code>P2_SSI_Set_Bits</code> <code>P2_SSI_Set_Clock</code> <code>P2_SSI_Set_Delay</code> , <code>P2_SSI_Read</code> <code>P2_SSI_Read2</code> <code>P2_SSI_Start</code> , <code>P2_SSI_Status</code>

1. A low/high signal will reliably be recognized in the indicated voltage ranges. But the switching process can also be effected outside these voltage ranges.

Function	Instructions
Use event inputs	<code>P2_Event_Enable</code> , <code>P2_Event_Config</code> <code>P2_Event_Read</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Synchronize	<code>P2_Sync_All</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `Cnt_TiCo.inc` contains instructions for the functions:

Function	Instructions
Configure counters	<code>Cnt_Enable</code> , <code>Cnt_Mode</code>
Use counters	<code>Cnt_Clear</code> , <code>Cnt_Get_Status</code> <code>Cnt_Latch</code> , <code>Cnt_Sync_Latch</code> <code>Cnt_Read</code> , <code>Cnt_Read_Latch</code> <code>Cnt_Read_Int_Register</code>
Use PWM counters	<code>Cnt_PW_Enable</code> , <code>Cnt_PW_Latch</code> <code>Cnt_Get_PW_HL</code> <code>P2_Cnt_Read_Int_Register</code>
Use SSI decoders (CNT-D only)	<code>SSI_Mode</code> , <code>SSI_Set_Bits</code> <code>SSI_Set_Clock</code> <code>SSI_Set_Delay</code> , <code>SSI_Read</code> <code>SSI_Start</code> , <code>SSI_Status</code>
Use event inputs	<code>Event_Enable</code> , <code>Event_Config</code> <code>Trigger_Event</code>
Use LED	<code>Check_LED</code> , <code>Set_LED</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

Programming in TiCoBasic

Programming TiCo access

5.8 Pro II: Extension and Interface Modules

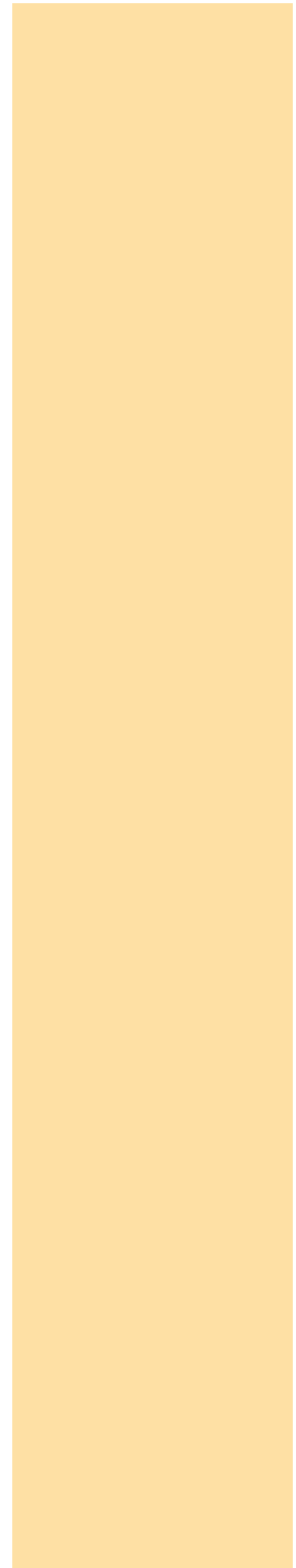
Modul	RTD-8	TC-8-ISO
Revision	E	E
Function	temperature measurement	thermo couple interface
Measuring	2 wire, 3 wire or 4 wire	–
Temp. range	-200°C...+700°C	B: 250°C ... 1820°C E: -200°C...1000°C J: -210°C...1250°C K: -200°C ... 1372°C N: -200°C...1300°C R: -50°C ... 1768°C S: -50°C ... 1768°C T: -270°C ... 400°C
Precision	±0.05...0.1K	±1K
Channels	8	8
Page	149	153

Module	CAN-2	CAN-2-LS	LIN-2	CAN-FD-2
Revision	E	E	E	E
Type	CAN interface, with TiCo1 processor		LIN bus interface	CAN FD interface with TiCo1 processor
Interface type	High speed	Low speed	–	–
Interfaces	2		2	2
Page	158		178	163

Module	Profi-SL	Profinet-IRT
Revision	E	E
Type	Profibus interface	Profinet-IRT interface
Interfaces	1	1
Connectors	DSub	Cu (copper)/ FO (fiber optics)
TiCo processor	–	TiCo1
Page	180	183

Module	RSxxx-2	RSxxx-4	RS422-4	SG-4/18
Revision	E	E	E	E
Type	RSxxx interface, with TiCo1 processor		RS422	Strain gage
Interface type	RS232 RS485	RS232 RS485	RS422	
Interfaces	2	4	4	4
Data exchange rate	0.035... 2304 kBaud	0.035... 2304 kBaud	0.035... 2304 kBaud	–
Page	171		175	155

Module	MIL-1553	ARINC-429	SPI-2-T	SPI-2-D
Revision	E	E	E	E
Type	MIL-1553 interface	ARINC-429 interface, with TiCo1 processor	SPI master / SPI slave, with TiCo1 processor	SPI master / SPI slave, with TiCo1 processor
Signal type	–	–	TTL	differential
Interfaces	2	3	2	2
Data exchange rate	1 MBit/s	100 kBit/s or 12.5 kBit/s	–	–
Page	187	190	213	213



Module	FlexRay-2	EtherCAT-SL	LS bus
Revision	E	E	E
Function	FlexRay interface	EtherCAT interface	LS bus interface
Interfaces	2	1	2
Page	194	196	221

Module	SENT-4	SENT-6	SENT-4-Out
Revision	E	E	E
Type	SENT interface	SENT interface	SENT interface
Interface type	4 inputs	6 inputs	4 outputs
Digital channels	–	4 in, 4 out	4 in, 4 out
Page	199	204	209

5.8.1 Pro II-RTD-8 Rev. E

The module Pro II-RTD-8 Rev. E has 8 inputs for connecting platinum resistance temperature detectors of the type Pt 100, Pt500, Pt1000, or Ni100. The inputs are connected via a multiplexer with an ADC. The maximum possible measurement range is $-200^{\circ}\text{C} \dots +700^{\circ}\text{C}$, depending on the temperature sensor (see data sheets of manufacturer).

There are 2 module variants:

- Pro II-RTD-8 Rev. E: Module inputs on a 37-pin D-Sub connector.
- Pro II-RTD-8-L Rev. E: Module inputs on Lemo connectors. Pin assignment see [fig. 132](#).

Measurements can be done with 2, 3 or 4 wire technique; input circuit see [fig. 131](#). Measuring method and sensor type are set via software.

Measuring methods and wiring between sensor and module are described starting from [page 151](#).

Measurements are performed automatically by a sequence control according to the settings. During a measurement, multiplexers connect the measuring wires with the sensor and disconnect them afterwards. Then, the measurement values are available in the module, so the ADwin CPU only has to read them.

Input signals run through a second order low-pass filter with 25kHz. Additionally, you can use the instruction `P2_RTD_Channel_Config` to filter a single frequency from the digital signals; the measurement value is calculated as mean value of several measurements being done in defined sampling intervals.

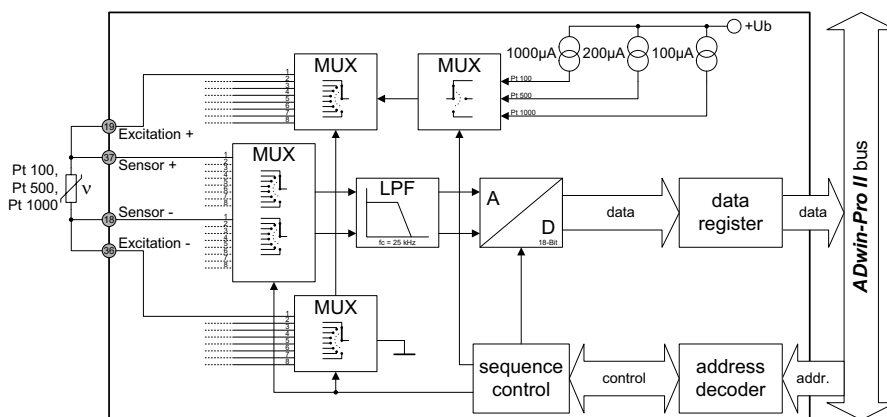


Fig. 131 – Pro II-RTD-8 Rev. E: Block diagram

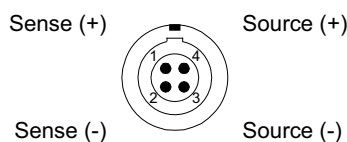


Fig. 132 – Pro II-RTD-8 Rev. E: Pin assignment Lemo connector

Inputs	8
Measurement	2 wire, 3 wire or 4 wire
Multiplexer settling time	100 μs
Input filter	25kHz (2nd order)
Max. measurement range	$-200^{\circ}\text{C} \dots +700^{\circ}\text{C}$

Fig. 133 – Pro II-RTD-8 Rev. E: Specification

Accuracy	PT100: ±0.05K PT500: ±0.1K PT1000: ±0.1K Ni100: ±0.05K
Temp. resolution	0.015K
Drift	±10ppm/K
$I_1 = I_2$	PT100: 1mA PT500: 0.2mA PT1000: 0.1mA Ni100: 1mA
Connector	37-pin D-Sub female connector or Lemo connectors

Fig. 133 – Pro II-RTD-8 Rev. E: Specification

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Range	Instruction
Configure module	<code>P2_RTD_Config</code> <code>P2_RTD_Channel_Config</code>
Access temperature channels	<code>P2_RTD_Start</code> <code>P2_RTD_Status</code> <code>P2_RTD_Read</code> <code>P2_RTD_Read8</code>
Convert measured value	<code>P2_RTD_Convert</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>

Measurement Method

You can choose one of three measurement methods: [2 wire measurement](#), [3 wire measurement](#) or [4 wire measurement](#). 4 wire measurement is recommended because it is the most precise method.

– 2 wire measurement

Please pay attention to a very short connection with low impedance between the Pt sensor and the module input, because the voltage drop gets added to the measured voltage.

This is the reason why this measurement method is in general not to be recommended for precise measurements.

Running a 2 wire measurement on channel *n*, you connect the sensor to the inputs `Excitation +` and `Excitation -`. For channel 1 it would be pins 19 and 36 (see [fig. 134](#)) on the D-Sub connector.

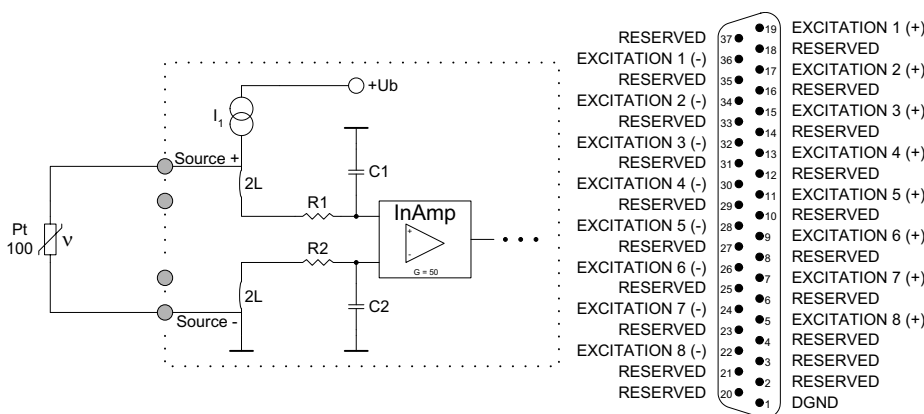


Fig. 134 – Pro II-RTD-8 Rev. E:

Block diagram and pin assignment with 2 wire measurement

– 3 wire measurement

In order to avoid the disadvantages of the 2 wire measurement, the voltage drop in the measurement lines is here compensated by a second voltage source *I2*.

To keep the measurement error as small as possible, the resistance value of the three measurement lines from the Pt sensor to the module input should be identical.

With a 3 wire measurement, you need 2 measurements for each measurement value. Thus, a 3 wire measurement requires double the time of 2 or 4 wire measurement.

Running a 3 wire measurement on channel n , you connect the sensor to the inputs Excitation +, Sensor - and Excitation -. For channel 1 it would be pins 18, 19 and 36 (see fig. 135) on the D-Sub connector.

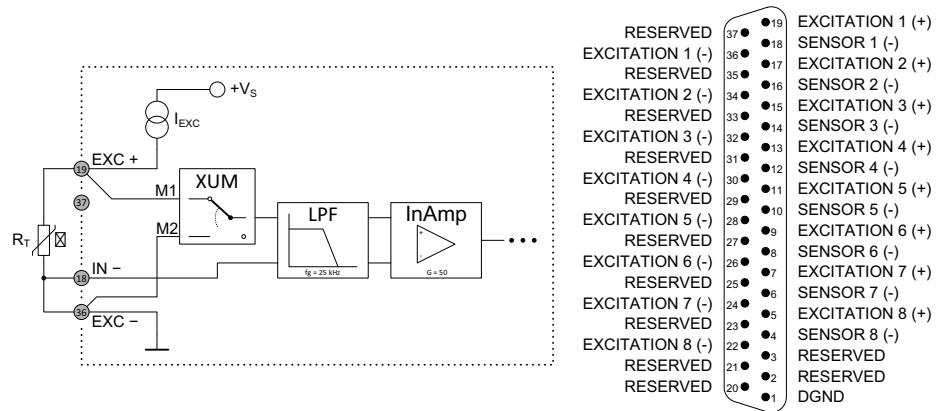


Fig. 135 – Pro II-RTD-8 Rev. E:
 Block diagram and pin assignment with 3 wire measurement

– 4 wire measurement

The voltage drop at the Pt sensor is directly avoided with high impedance at the PCB by the two "sensor" inputs. The resistance of the measurement lines does not have an effect here any longer and need therefore not be compensated.

Running a 4 wire measurement on channel n , you connect the sensor to the inputs Excitation +, Sensor +, Sensor - and Excitation -. For channel 1 it would be pins 18, 19, 36 and 37 (see fig. 136) on the D-Sub connector.

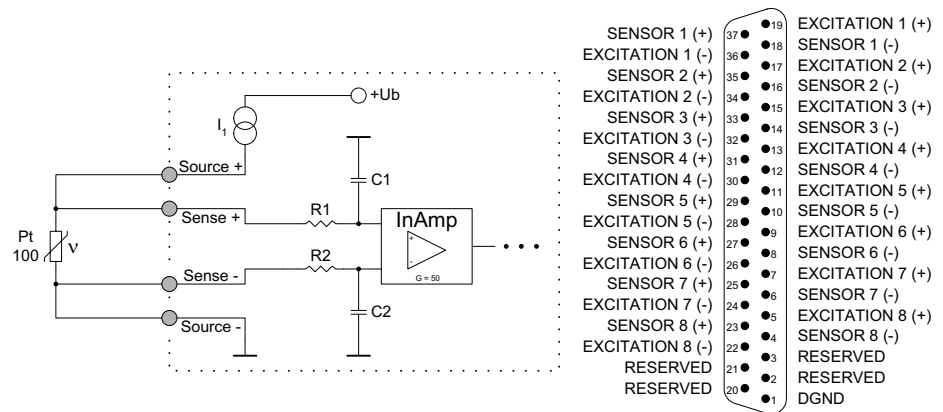


Fig. 136 – Pro II-RTD-8 Rev. E:
 Block diagram and pin assignment with 4 wire measurement

5.8.2 Pro II-TC-8 ISO Rev. E

The module [Pro II-TC-8 ISO Rev. E](#) has 8 inputs for thermocouples and can be operated with thermocouple types B, E, J, K, N, R, S or T. For each channel the thermoelectric voltage (with or without cold junction correction) or the temperature may be queried separately via software.

Each channel is equipped with a separate ADC. The module provides a common cold junction compensation for all temperature inputs.

The jumper position (see [fig. 139](#), left side) sets for each channel separately if the channel potentials are separated from each other:

- Position right: The channel potentials are separated from each other (default).
- Position left: The channel's negative input is connected to ground.

Input signals at the ADCs are digitized at a stepwise adjustable sample rate. As soon as a value is queried via software, the module calculates the thermoelectric voltage or the temperature in °Celsius or °Fahrenheit from the last measurement value. All calculation is based upon the norm IEC 584-1. The thermoelectric voltage without cold junction correction can be returned, too.

The inputs have a Butterworth filter with 5Hz as low-pass. The module can also be ordered without low-pass.

Calibration of the module is performed by the manufacturer. If needed, please send the module to the address given on the back of the cover page.

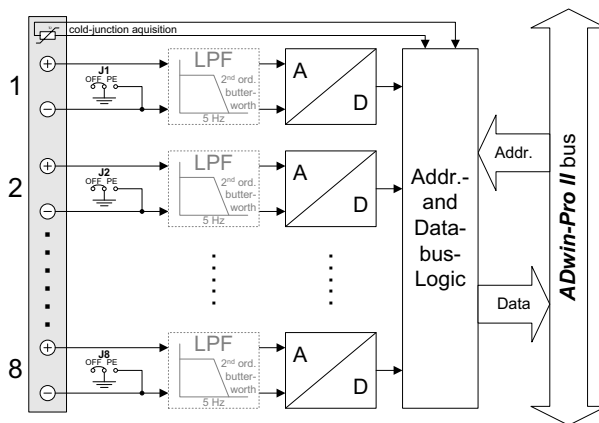


Fig. 137 – Pro II-TC-8 ISO Rev. E: Block diagram

Input channels	8
Sample rate	7Hz ... 3500Hz
Thermocouple types, measuring range and accuracy	B: 250°C ... 1820°C; ±5°C E: -200°C...1000°C; ±1°C J: -210°C...1250°C; ±1°C K: -200°C ... 1372°C; ±1°C N: -200°C...1300°C; ±2°C R: -50°C ... 1768°C; ±3°C S: -50°C ... 1768°C; ±3°C T: -270°C ... 400°C; ±1°C
Resolution	0.1°C
Input resistance	10MΩ
Input filter	5Hz Butterworth
Input over-voltage	±20V

Fig. 138 – Pro II-TC-8 ISO Rev. E: Spezifikation

Offset drift	±30ppm/°K of full scale range
Connector	Omega Subminiature Connector, Type SMP

Fig. 138 – Pro II-TC-8 ISO Rev. E: Specification

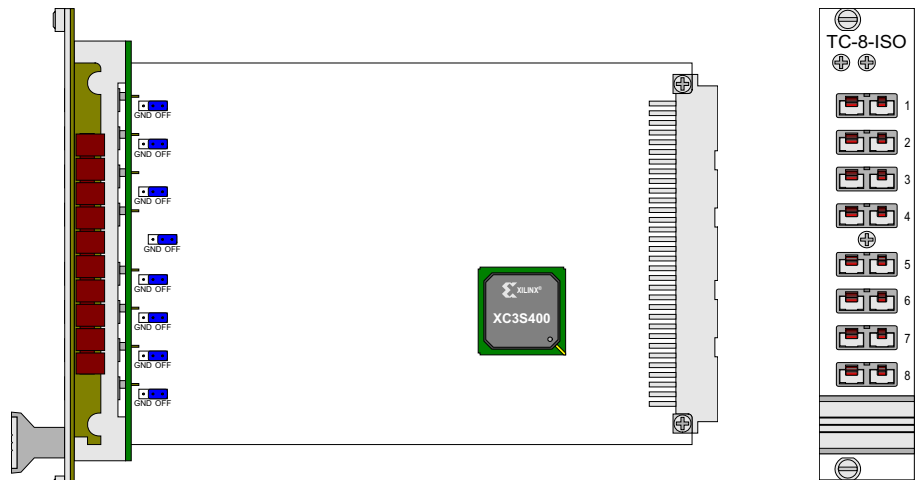


Fig. 139 – Pro II-TC-8 ISO Rev. E: Board and front panel

Programming

The following instructions are used to program the module:

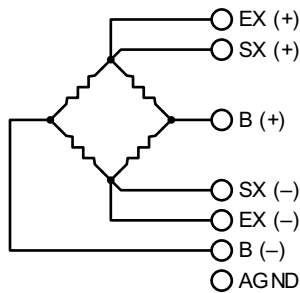
Set sampling rate	<code>P2_TC_Set_Rate</code>
Copy input values to latches	<code>P2_TC_Latch, P2_Sync_All</code>
Read values	<code>P2_TC_Read_Latch</code> <code>P2_TC_Read_Latch4, P2_TC_Read_Latch8</code>

5.8.3 Pro II-SG-4/18 Rev. E

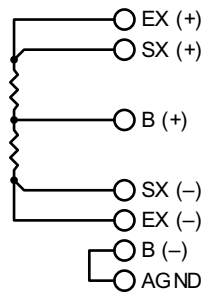
The module [Pro II-SG-4/18 Rev. E](#) provides 4 inputs for strain gage measurement and 2 analog inputs and is suitable for tension and compression. All inputs are differential and connected to an 18-bit ADC via a multiplexer.

Strain gages can be connected with a half bridge or full bridge circuit, and with 6-wire or 4-wire circuit.

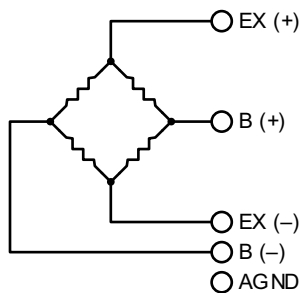
The minimum nominal resistance of strain gage depends on the excitation voltage EX; with EX=9.80V the minimum resistance is 100Ω, with EX=1.25V it is 13Ω.



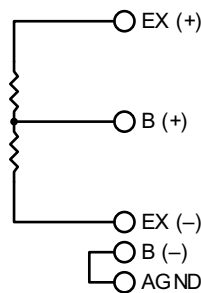
Full bridge, 6-wire



Half bridge, 6-wire



Full bridge, 4-wire

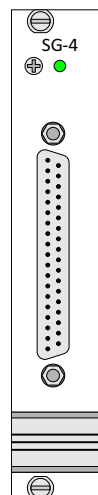


Half bridge, 4-wire

Abb. 140 – Pro II-SG-4/18 Rev. E: Strain gage wiring

The module [Pro II-SG-4/18 Rev. E](#) provides the bridge excitation voltage EX for the strain gage wiring; via software, the excitation voltage can be set separately for each strain gage wiring. From each strain gage wiring, 3 voltages can be measured: excitation voltage EX, sense voltage SX, and bridge voltage B. In addition, 2 analog inputs AIN13/AIN14 can be measured.

All pins are available on a 37-pin D-Sub female connector. All AGND pins are connected internally.



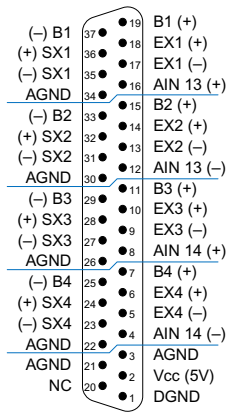


Abb. 141 – Pro II-SG-4/18 Rev. E: Pin assignment

The gain of the bridge voltage can be set in steps (20 ... 640) via software; this also sets the voltage range and the signal resolution. Other signals are measured with gain 1.

Gain	Voltage range	Resolution
20	±500mV	3.84 µV
40	±250mV	1.92 µV
80	±125mV	0.96 µV
160	±62.5mV	0.48 µV
320	±31.25mV	0.24 µV
640	±15.625mV	0.12 µV

To compensate zero offset, the bridge voltage can be adjusted via software **P2_SG_Zero** by up to ±10mV. In addition, the gain factor of strain gage can be calibrated by software; before, you have put a defined strain on strain gage.

Bridge sensitivity depends on the excitation voltage EX and the gain factor:

Voltage EX	min. gain 20	max. gain 640
1.25V	400mV/V	12.50mV/V
2.50V	200mV/V	6.25mV/V
3.75V	133mV/V	4.17mV/V
5.00V	100mV/V	3.13mV/V
6.25V	80mV/V	2.50mV/V
7.50V	66mV/V	2.08mV/V
8.75V	57mV/V	1.79mV/V
9.80V	51mV/V	1.59mV/V

For each channel there are several filters:

Für jeden Kanal sind mehrere Filter vorhanden:

- 1 MHz at the input
- 2 kHz, 2nd order after first amplifier
- 2kHz or 200 Hz (2nd order) after second amplifier, selectable by software

Calibration of the module is performed by the manufacturer. If needed, please send the module to the address given on the back of the cover page.

Strain gage inputs	4 (differentiell)
Analog inputs	2 (differentiell)
Min. nominal resistance of strain gage	acc. to EX: 50Ω... 400Ω
Resolution ADC	18 Bit
ADC conversion time	2μs
Voltage range strain gage	max. ±500mV; min. ±15,625mV
Voltage range analog inputs	±10V
Multiplexer settling time	5μs
Bridge sensitivity	1.59mV/V ... 400mV/V
Input resistance	>10MΩ
Input filter	2kHz or 20kHz
Input over-voltage	±5V
Bridge excitation voltage	max. 25mA per channel short-circuit-proof
Connector	37-pin D-Sub female connector

Abb. 142 – Pro II-SG-4/18 Rev. E: Specification

The following instructions are used to program the module:

Range	Instruction
Initialize	P2_SG_Mode, P2_SG_Init, P2_SG_Zero, P2_SG_Set_Gain
Measure	P2_SG_Start, P2_SG_Wait, P2_SG_Read
Convert resulting value	P2_SG_Convert
Set LEDs	P2_Check_LED, P2_Set_LED

Programming

5.8.4 Pro II-CAN-2 Rev. E

The module Pro II-CAN-2 Rev. E has 2 CAN interfaces, a high speed or a low speed version. The names for the module versions are:

- Pro II-CAN-2 Rev. E: CAN interface high speed
- Pro II-CAN-2-LS Rev. E: CAN interface low speed

Each module is equipped with a freely programmable *TiCo* processor, which has full access to the CAN interfaces. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

The manual is divided into the following sections:

- [CAN Controller](#)
- [TiCo processor](#)
- [Hardware design](#)
- [Message Management](#)
- [Setting the bus frequency](#)
- [Enable Interrupt / Trigger Event](#)
- [Module revisions](#)
- [Programming](#)

CAN Controller

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

Messages sent via CAN bus are data telegrams with up to 8 bytes, which are characterized by so-called identifiers. The CAN controller 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 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.

TiCo processor

The module provides the freely programmable *TiCo* processor with 28 KiB program memory, and 28 KiB data memory. The *TiCo* processor has access to the CAN controller. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Message



Hardware design

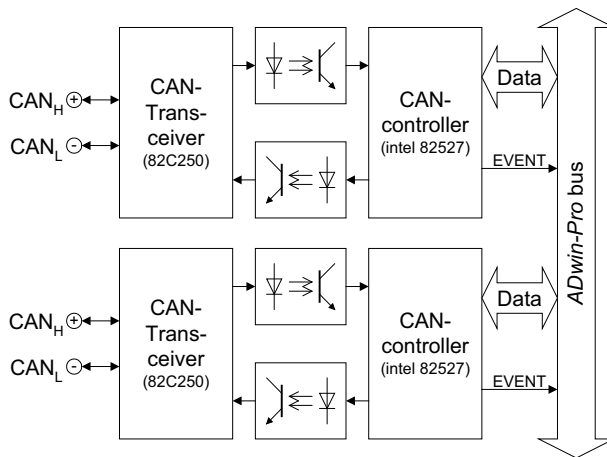


Fig. 143 – Pro II-CAN-2: Block diagram

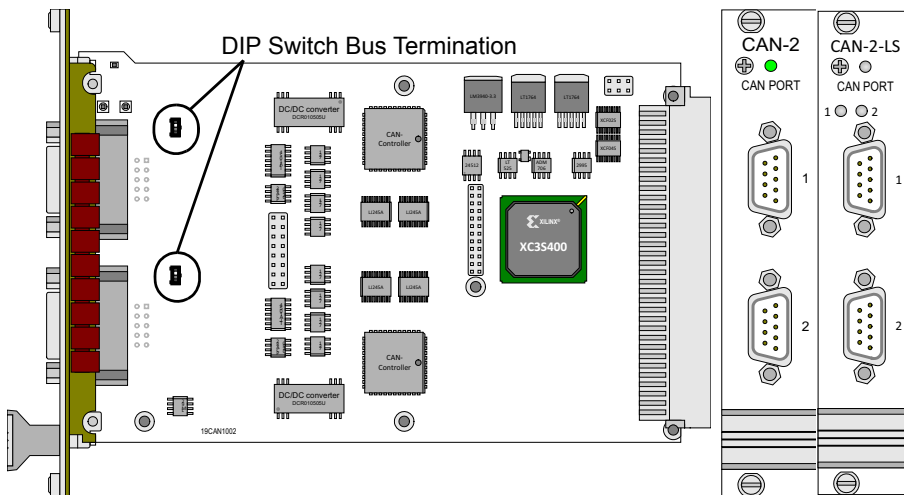


Fig. 144 – Pro II-CAN-2: PCB and front panels

The connections of the CAN bus interface are on the 9-pin D-SUB connector; the pin assignment is shown below.

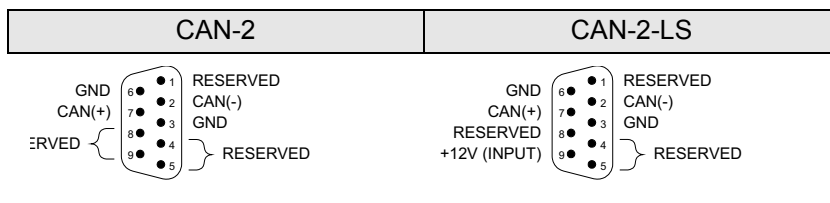


Fig. 145 – Pro II-CAN-2: Pin assignment (male)

The "low speed" version Pro II-CAN-2-LS requires an external power supply of 12V DC. The module needs a power supply for each controller separately.

With low-speed CAN, both DIP switches (see [fig. 144](#)) must be set downward (off-position).

If the CAN module 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).

If a termination is necessary, move the DIP switch (see [fig. 144](#)) upward. CAN nodes, which are not positioned in an end-location, must not be terminated.

**Power supply
(Low speed only)**

**Bus termination
(High speed only)**

Message Management	
Identifier	The CAN controller identifies messages by an identifier; these are parameters in a defined bit length. The parameters $0 \dots 2^{11}-1$ or $0 \dots 2^{29}-1$ result from the bit length.
Message objects	<p>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.</p> <p>Each message object has an identifier, which enables the user to assign a message to a message object.</p>
Transferring messages	In <i>ADbasic</i> , a message is transferred to a message object using the array <code>can_msg []</code> , 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 <code>can_msg []</code> .
Sending messages	<p>Sending a message is made as follows:</p> <ul style="list-style-type: none">– You configure a message object to send and define the identifier of the object (instruction <code>En_Transmit</code>).– Save the message in <code>can_msg []</code>.– Send the message (instruction <code>Transmit</code>). The message in the array <code>can_msg []</code> 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	<p>Receiving a message is made as follows:</p> <ul style="list-style-type: none">– You configure a message object to receive and define the identifier of the object (instruction <code>En_Receive</code>).– 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 <code>can_msg []</code> (instruction <code>Read_Msg</code>) and read out the corresponding identifier. <p>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.</p> <p>The message object 15 has an additional buffer, so that 2 messages can be stored there.</p>
Assigning messages	<p>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:</p> <ul style="list-style-type: none">– 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 <code>111111111100b</code>).

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, just use the instruction `Set_CAN_Baudrate`.

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

In some special cases, it may be better to select configurations other than those set with `Set_CAN_Baudrate`. For this purpose specified registers have to be set with the instruction `Poke`. The structure of the register is described in the controller documentation.

Bus frequency for special cases

Enable Interrupt / Trigger 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.

Module revisions

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E		First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

**Programming in
 TiCoBasic**

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Initialize CAN controller, set Baud rate	<code>P2_INIT_CAN</code> <code>P2_Set_CAN_Baudrate</code>
Transfer message objects	<code>P2_En_Receive</code> <code>P2_Read_Msg</code> <code>P2_En_Transmit</code> <code>P2_Transmit</code>
Set and read registers	<code>P2_Set_CAN_Reg</code> <code>P2_Get_CAN_Reg</code>
Use LED	<code>P2_Check_LED</code> <code>P2_Set_LED</code> <code>P2_CAN_Set_LED</code>
Use interrupt and event inputs	<code>P2_En_Interrupt</code> <code>P2_Event_Enable</code> <code>P2_Event_Config</code> <code>P2_Event_Read</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `CAN_TiCo.inc` contains instructions for the functions:

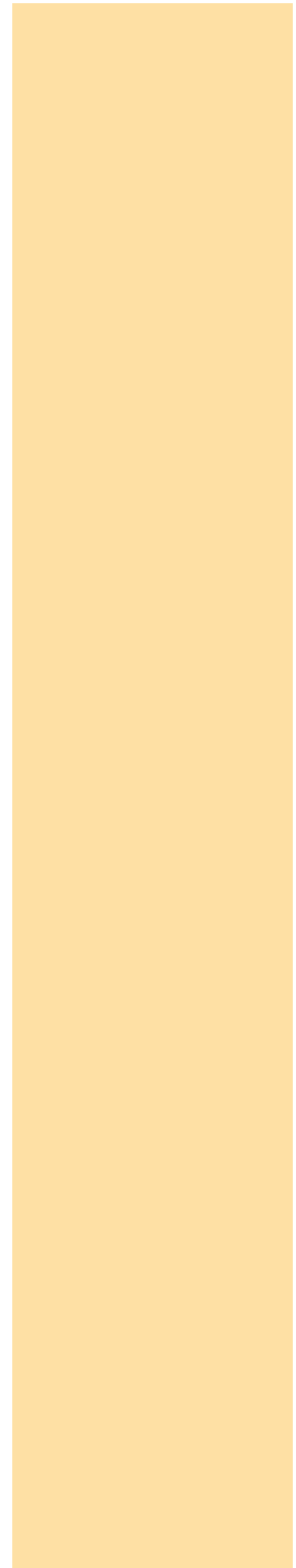
Function	Instructions
Initialize CAN controller, set Baud rate	<code>INIT_CAN</code> <code>Set_CAN_Baudrate</code>
Transfer message objects	<code>En_Receive, Read_Msg</code> <code>Read_Msg_Con</code> <code>En_Transmit, Transmit</code> <code>Transmit_Status</code>
Set and read registers	<code>Set_CAN_Reg, Get_CAN_Reg</code>
Use LED	<code>Check_LED, Set_LED</code> <code>CAN_Set_LED</code>
Use interrupt	<code>En_Interrupt</code>

**Programming TiCo
 access**

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start,</code> <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>

Function	Instructions
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>



5.8.5 Pro II-CAN-FD-2 Rev. E

The module Pro II-CAN-FD-2 Rev. E has 2 CAN FD interfaces.

Each module is equipped with a freely programmable *TiCo* processor, which has full access to the CAN interfaces. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

The manual is divided into the following sections:

- [CAN FD Controller](#)
- [TiCo processor](#)
- [Hardware design](#)
- [Message Management](#)
- [Module revisions](#)
- [Programming](#)

CAN FD Controller

The CAN bus interface is equipped with the Microchip® CAN controller MCP2517FD, which works according to the specification ISO 11898-1. You program the interface with *ADbasic* instructions, which are directly accessing the controller's registers.

The controller supports both CAN 2.0B (high speed) and CAN FD (also mixed mode with CAN 2.0B).

Messages sent via CAN FD bus are data telegrams with up to 64 data bytes, which are characterized by so-called identifiers. The CAN controller supports identifiers with a length of 11 bit, 12 bit, and 29 bit. The communication, that means the management of bus messages, is effected by 32 message objects. Message objects 1...31 run as FIFO (first in, first out) and can be configured as input or output; message object 0 is a transmit queue which transmits messages with lowest identifier first.

The controller can be set to frequencies of up to 8Mbps (data bit rate) or 1 Mbps (arbitration bit rate). With CAN 2.0B the max. frequency is 1 MHz. The CAN bus is galvanically isolated by optocouplers from the *ADwin* system.

TiCo processor

The module provides the freely programmable *TiCo* processor with 28KiB program memory, and 28KiB data memory. The *TiCo* processor has access to the CAN controller. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Hardware design

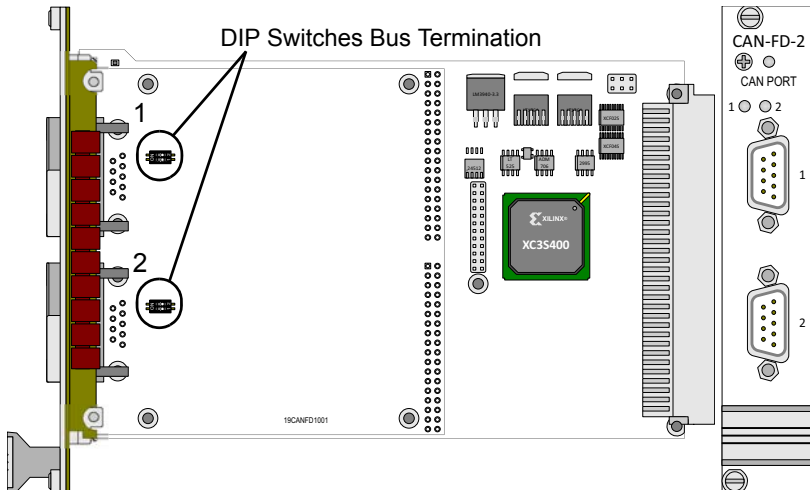


Fig. 146 – Pro II-CAN-FD-2 Rev. E: PCB and front panels

The connections of the CAN FD bus interface are on the 9-pin D-SUB connector; the pin assignment is shown below.

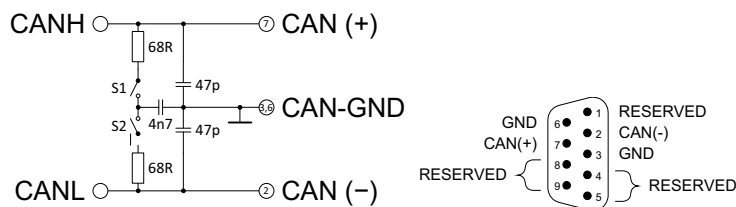


Fig. 147 – Pro II-CAN-FD-2 Rev. E: Block diagram and pin assignment (male)

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

If a termination is necessary, move both DIP switches (see [fig. 146](#)) of the appropriate CAN interface to the left.

Message Management

The CAN controller identifies messages by an identifier; these are parameters in a defined bit length. The parameters $0 \dots 2^{11}-1$, $0 \dots 2^{12}-1$, or $0 \dots 2^{29}-1$ result from the bit length.

The controller stores each message (incoming or outgoing) in one out of 32 message objects. Each message object can store up to 32 messages. The message objects 1...31 are built as FIFO and can either be configured to send (output fifo) or to receive messages (input fifo). In a Fifo (first in, first out), messages are accessed in the order as they have been stored.

The message object 0, if initialized, always runs as transmit queue. The queue processes messages in the order of priority. The message with the lowest identifier gets transmitted first.

Each message object has an identifier, which enables the user to assign a message to a message object. Via a mask (see [Assigning messages](#)), even a group of a identifiers can be assigned to a message object.

In order to control the sending operation, you can initialize a transmit event Fifo. The Fifo can store the message header and an optional timestamp for each sent message.

Bus termination

Identifier

Message objects

Initialization

Message objects require space in the controller memory. Overall, there is 2kB memory which can be distributed to message objects and the transmit event Fifo.

For initialization, a data array is created where all properties of the CAN FD controller (operation mode, bit rate, transmitter delay compensation) as well as properties of used message objects (e.g. size, identifier) are stored. With `P2_CANFD_Init_Controller`, all stored information from the data array are transferred to the appropriate controller registers. Then, the controller is ready for operation.

Sending messages

Sending a message is made as follows:

- You configure a message object to send using `P2_CANFD_Enable_Transmit_Fifo` or `P2_CANFD_Enable_Transmit_Queue`. Then, initialize the controller with `P2_CANFD_Init_Controller` (see [Initialization](#)).
- You transfer a message including the identifier to the controller (`P2_CANFD_Write_TMO`).
- You enable the message object for sending (`P2_CANFD_Transmit_MSG`). As soon as the bus is ready, the message is sent (with the identifier of the message object).

Please note: If messages are ready for sending in several message objects, the message with the lowest identifier is sent first.

While sending, the controller reads back data from the bus in order to enable reactions to conflicts according to the bus protocol. During read back, a delay can occur compared to the sent data; a transmitter delay compensation can be set (`P2_CANFD_Set_TDC`) with controller initialization.

Controlling sent messages

You can define a transmit event fifo in order to control sent messages. The Fifo can store the message header and an optional timestamp for each sent message. You read a message object EFO with `P2_CANFD_Read_EFO`.

If you want to control and distinguish messages with identical header, enter a specific parameter SEQ when sending a message. After sending, you find the parameter in the header being stored in the transmit event fifo.

Receiving messages

Receiving a message is made as follows:

- You configure a message object to receive and define the identifier of the object (`P2_CANFD_Enable_Receive_Fifo`). Then, initialize the controller with `P2_CANFD_Init_Controller` (see [Initialization](#)).
- The controller monitors the CAN bus for incoming messages and stores messages with the appropriate identifier in the input Fifo.
- You read the message object from the input Fifo (`P2_CANFD_Read_RMO`).

If all defined elements of the input Fifo are full, the next incoming message overwrites the oldest stored data, which will be definitely lost. Therefore, pay attention to reading out the data faster than you are receiving them. A data loss can be queried separately.

Assigning messages

The allocation of an incoming message to an input Fifo is automatically controlled by comparing its identifiers. Each input Fifo has a mask which controls this comparison as follows:

- The identifier of the message is bit by bit compared to the identifier of the input Fifo. If the relevant bits are identical, the message is transferred to the input Fifo. Not relevant bits are not compared to each other, that is, the message is transferred to the input Fifo (if it depends on this bit).
- Relevant bits are defined in the mask by setting them to 1.

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

Message ID	ID of the input Fifo			
	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).

Module revisions

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E	04/2018	First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

Programming in ADbasic

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Initialize CAN controller, set Baud rate	<code>P2_CANFD_Init_Datatable</code> <code>P2_CANFD_Init_Controller</code> <code>P2_CANFD_Enable_Receive_Fifo</code> <code>P2_CANFD_Enable_Transmit_Fifo</code> <code>P2_CANFD_Enable_Transmit_Queue</code> <code>P2_CANFD_Enable_Transmit_Event_Fifo</code> <code>P2_CANFD_Set_Baudrate_Nominal</code> <code>P2_CANFD_Set_Baudrate_Data</code> <code>P2_CANFD_Set_SID11</code> <code>P2_CANFD_Set_TDC</code> <code>P2_CANFD_Set_Mode</code>
Check status	<code>P2_CANFD_Get_Fifo_State</code>
Receive message	<code>P2_CANFD_Read_RMO</code> <code>P2_CANFD_Get_Header_Parts</code> <code>P2_CANFD_Get_ID</code> <code>P2_CANFD_Get_ESI</code> <code>P2_CANFD_Get_FDF</code> <code>P2_CANFD_Get_BRS</code> <code>P2_CANFD_Get_RTR</code> <code>P2_CANFD_Get_IDE</code> <code>P2_CANFD_Get_DLC</code>
Transfer and check message	<code>P2_CANFD_Write_TMO</code> <code>P2_CANFD_Transmit_MSG</code> <code>P2_CANFD_Transmit_Multi_MSG</code> <code>P2_CANFD_Read_EFO</code> <code>P2_CANFD_Get_SEQ</code>
Read registers	<code>P2_CANFD_Get_TREC</code> <code>P2_CANFD_Get_BDIAG0</code> <code>P2_CANFD_Get_BDIAG1</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code> <code>P2_CANFD_Set_LED</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `CAN_TiCo.inc` contains instructions for the functions:

Function	Instructions
Initialize CAN controller, set Baud rate	<code>CANFD_Init_Datatable</code> <code>CANFD_Init_Controller</code> <code>CANFD_Enable_Receive_Fifo</code> <code>CANFD_Enable_Transmit_Fifo</code> <code>CANFD_Enable_Transmit_Queue</code> <code>CANFD_Enable_Transmit_Event_Fifo</code> <code>CANFD_Set_Baudrate_Nominal</code> <code>CANFD_Set_Baudrate_Data</code> <code>CANFD_Set_SID11</code> <code>CANFD_Set_TDC</code> <code>CANFD_Set_Mode</code>
Check status	<code>CANFD_Get_Fifo_State</code>
Read message	<code>CANFD_Read_RMO</code> <code>CANFD_Get_Header_Parts</code> <code>CANFD_Get_ID</code> <code>CANFD_Get_ESI</code> <code>CANFD_Get_FDF</code> <code>CANFD_Get_BRS</code> <code>CANFD_Get_RTR</code> <code>CANFD_Get_IDE</code> <code>CANFD_Get_DLC</code>
Transfer and check message	<code>CANFD_Write_TMO</code> <code>CANFD_Transmit_MSG</code> <code>CANFD_Transmit_Multi_MSG</code> <code>CANFD_Read_EFO</code> <code>CANFD_Get_SEQ</code>
Read registers	<code>CANFD_Get_TREC</code> , <code>CANFD_Get_BDIAG0</code> <code>CANFD_Get_BDIAG1</code>
Use LED	<code>Check_LED</code> , <code>Set_LED</code> , <code>CANFD_Set_LED</code>

Programming in TiCoBasic

Programming TiCo access

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start,</code> <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

5.8.6 Pro II-RSxxx Rev. E

The Pro II-RSxxx module has 2 or 4 interfaces of the type RS-232 or RS-485. The type of interface is selected with the instruction `P2_RS485_Send`.

The names for the module versions are:

- Pro II-RSx-2 Rev. E: 2 interfaces RS232/485
- Pro II-RSx-4 Rev. E: 4 interfaces RS232/485

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

Each module is equipped with a freely programmable *TiCo* processor, which has full access to the RSxxx interfaces. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

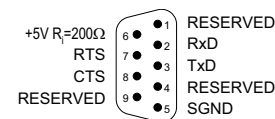
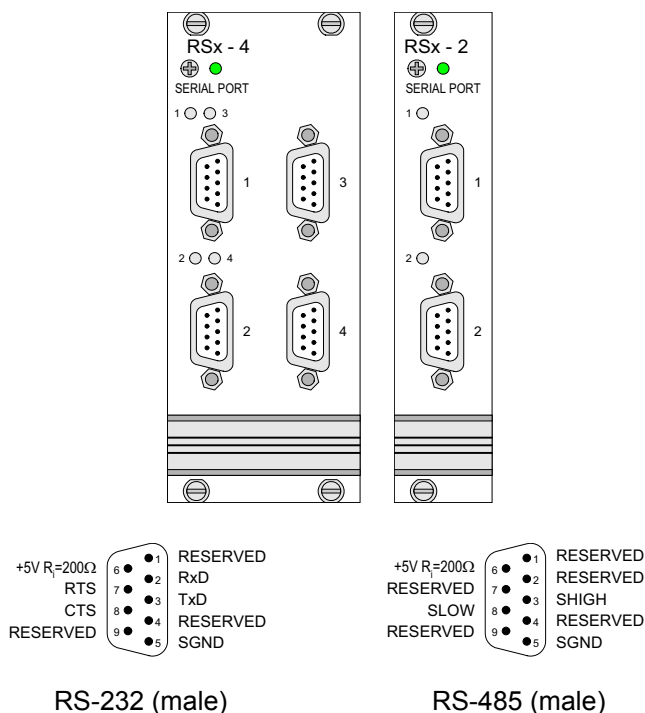
The physical difference between the interface versions is their signal level, which is provided by appropriate drivers on the bus.

The description is divided into the following paragraphs:

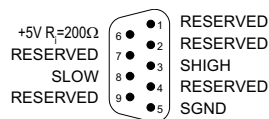
- [Hardware](#)
- [Interface parameters](#)
- [Module revisions](#)
- [Programming](#)

Hardware

These are the front panels and pin assignments of the modules Pro II-RSxxx. The pin assignment is switched together with the interface type.



RS-232 (male)



RS-485 (male)

Fig. 148 – Pro II-RS-xxx: Front panels / pin assignments

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 can be operated in 3 modes:
 1. Without handshake
 2. Software handshake
 3. Hardware handshake (RS232 only).
When using the hardware handshake the signals RTS and CTS must be connected.

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 115.2kBaund, according to the specification.

The Baud rates are derived from the clock rate of the module; the basic clock rate has a frequency of 2.304MHz. Based on this fact, every Baud rate is possible, which can be derived from 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		Baud rate	Divisor	
	dec.	hex.		dec.	hex.
2.304.000	1	0001h	19.200	120	0078h
1.152.000	2	0002h	9.600	240	00F0h
460.800	5	0005h	4.800	480	01E0h
230.400	10	000Ah	2.400	960	03C0h
115.200	20	0014h	1.200	1920	0780h
57.600	40	0028h	600	3840	0F00h
38.400	60	003Ch	300	7680	1E00h

Fig. 149 – Pro II-RS-xxx: Baud rates

Special features of RS485

Contrary to the RS232 and RS422 interface, with RS485 more than 2 participants can communicate with each other. With an RS485 interface, a bus is set up.

Consider the following:

- There is no handshake, because a handshake is only possible between two participants.
- The interface must know if it should write to the bus or get data from the bus (**RS485_SEND**).

Module revisions

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E	11/2007	First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Initialization	<code>P2_RS_INIT, P2_rs_reset</code>
Receiving and transmitting of data	<code>P2_read_fifo, P2_write_fifo</code>
Configure RS485 channel	<code>P2_RS485_Send</code>
Write and read access to the controller register	<code>P2_GET_RS, P2_SET_RS</code>
Use LED	<code>P2_Check_LED, P2_Set_LED</code> <code>P2_RS_Set_LED</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `RS_LIN_TiCo.inc` contains instructions for the following functions:

Function	Instructions
Initialization	<code>RS_INIT, rs_reset</code>
Receiving and transmitting of data	<code>read_fifo, write_fifo</code> <code>Check_Shift_Reg</code>
Configure RS485 channel	<code>RS485_Send</code>
Write and read access to the controller register	<code>GET_RS, SET_RS</code>
Use LED	<code>Check_LED, Set_LED</code> <code>RS_Set_LED</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Programming in ADbasic

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	P2_TDrv_Init P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block P2_Get_TiCo_RingBuffer, P2_Set_TiCo_RingBuffer P2_RingBuffer_Empty P2_RingBuffer_Full
Control <i>TiCo</i> processor	P2_TiCo_Reset, P2_TiCo_Start, P2_TiCo_Stop P2_Get_TiCo_Bootloader_ Status P2_Get_TiCo_Status, P2_Workload
Control <i>TiCo</i> processes	P2_Process_Status P2_TiCo_Get_Processdelay P2_TiCo_Set_Processdelay P2_TiCo_Start_Process P2_TiCo_Stop_Process
Transfer <i>TiCo</i> programs	P2_TiCo_Flash, P2_TiCo_Load

5.8.7 Pro II-RS422-4 Rev. E

The module [Pro II-RS422-4 Rev. E](#) has 4 interfaces of the type RS-422.

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

Each module is equipped with a freely programmable *TiCo* processor, which has full access to the RS422 interfaces. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

The description is divided into the following paragraphs:

- [Hardware](#)
- [Interface parameters](#)
- [Module revisions](#)
- [Programming](#)

TiCo processor

The module provides the freely programmable *TiCo* processor with 28KiB program memory, and 28KiB data memory. The *TiCo* processor is programmed with *TiCoBasic*.

The *TiCo* processor has access to the RS422 controller. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Hardware

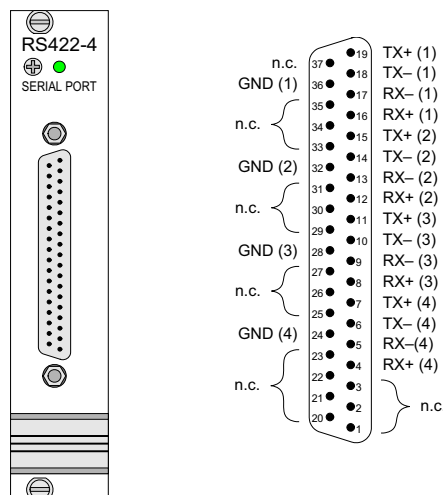


Fig. 150 – [Pro II-RS422-4 Rev. E](#): Front panel and pin assignment

Interface parameters

Initialize the interfaces for RS422 operation first with `P2_RS_Init`.

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 chan-

Handshake

nel, using the controller register. Below the settings are described more detailed:

- Handshake: The interface can be operated in 2 modes:
 1. Without handshake
 2. Software handshake

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.

The Baud rates are derived from the clock rate of the module; the basic clock rate has a frequency of 2.304MHz. Based on this fact, every Baud rate is possible, which can be derived from 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		Baud rate	Divisor	
	dec.	hex.		dec.	hex.
2.304.000	1	0001h	19.200	120	0078h
1.152.000	2	0002h	9.600	240	00F0h
460.800	5	0005h	4.800	480	01E0h
230.400	10	000Ah	2.400	960	03C0h
115.200	20	0014h	1.200	1920	0780h
57.600	40	0028h	600	3840	0F00h
38.400	60	003Ch	300	7680	1E00h

Fig. 151 – Pro II-RS422-4 Rev. E: Baud rates

Module revisions

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E	11/2011	First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Initialization	<code>P2_RS_Init</code> , <code>P2_RS_Reset</code>

Programming in ADbasic

Function	Instructions
Receiving and transmitting of data	<code>P2_Read_Fifo</code> , <code>P2_Write_Fifo</code>
Write and read access to the controller register	<code>P2_Get_RS</code> , <code>P2_Set_RS</code>
Use LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code> <code>P2_RS_Set_LED</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `RS_LIN_TiCo.inc` contains instructions for the following functions:

Function	Instructions
Initialization	<code>RS_Init</code> , <code>RS_Reset</code>
Receiving and transmitting of data	<code>Read_Fifo</code> , <code>Write_Fifo</code> <code>Check_Shift_Reg</code>
Write and read access to the controller register	<code>Get_RS</code> , <code>Set_RS</code>
Use LED	<code>Check_LED</code> , <code>Set_LED</code> <code>RS_Set_LED</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long</code> , <code>P2_Get_Par</code> , <code>P2_Get_Par_Block</code> <code>P2_SetData_Long</code> , <code>P2_Set_Par</code> , <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer</code> , <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>

Programming in
TiCoBasic

Programming **TiCo**
access

5.8.8 Pro II-LIN-2 Rev. E

The module Pro II-LIN-2 Rev. E provides 2 LIN interfaces, which can be independently configured as LIN master or LIN slave.

LIN Interface

The LIN interfaces of the module are implemented according to specification „LIN 2.1“ (Local Interconnect Network) of november 2006. You program the LIN interfaces with *ADbasic* instructions.

LIN is a serial communication protocol on a one-wire bus with a transfer rate of up to 20 KiBit/s. The bus efficiently supports the control of mechatronic nodes in distributed automotive applications.

Messages sent via LIN bus are data packets with up to 8 bytes payload, which are characterized by so-called identifiers. The management of bus messages is effected by 64 message boxes.

The bus concept refers to a single master node with multiple slave nodes. The master controls the total data transfer of the bus: Before each data packet the master sends a header with the identifier of the next data packet. Then, only this bus node will react (which can also be the master node itself), which manages a message box with the given identifier. Thus, this node will send a data packet to or receive a data packet from the LIN bus.

Hardware design

The connections of the LIN bus interfaces are on the 9-pin D-SUB connectors; the pin assignments are shown below.

The operating voltage range of +Bat is 8V...18V.

For each interface there is a programmable LED.

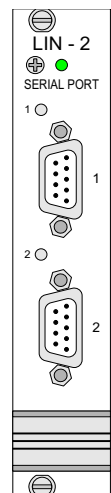
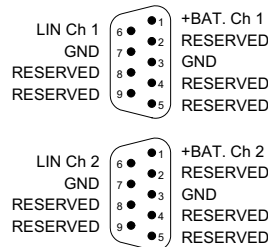


Fig. 152 – Pro II-LIN-2 Rev. E: Pin assignments (male)

Working with LIN bus

The 2 LIN interfaces of the module can be independently configured as LIN master or LIN slave with `P2_LIN_Init_Write`; this also true for the setting of the baudrate. The bus termination will be switched automatically according to the configuration.

The network master timing (NMT) is to be programmed in *ADbasic*.

While configuring a message box with `P2_LIN_Msg_Write` you set the identifier and the sending mode (send, receive) of the message box. Any number of message boxes can be assigned to a LIN interface, but each identifier may be used only once for sending and once for receiving on the LIN bus; otherwise data collisions will occur.

After configuring, a message box will at once be active on the LIN bus, i. e. data packets can be sent or received.

If a LIN interface is configured as master, you use `P2_LIN_Msg_Transmit` to send a header and the identifier of a message box to the LIN bus. The message boxes configured with this identifier will react automatically.

The message box of a LIN master node operates different from a slave node:

- **Master node, send:** The LIN master sends both the header (see `P2_LIN_Msg_Transmit`) and then the data packet of the message box.
- **Master node, receive:** The LIN master sends the header (see `P2_LIN_Msg_Transmit`) on the LIN Bus and waits for the response of the appropriate slave node. The received data packet is stored into the message box.
- **Slave node, send:** The LIN slave waits until the master sends the header with the identifier, which fits to the identifier of the message box. Only then the slave node will its data packet.
- **Slave node, receive:** The slave node waits until the master sends the header with the identifier, which fits to the identifier of the message box. Then the slave receives the data packet and stores it into the message box.

Module Revisions

The differences between the revisions are described below:

Revision	Output date	Previous changes
E1		First version

Programming

The module is comfortably programmed with **ADbasic** instructions. The instructions are described in **ADbasic** online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Initialize and reset LIN interfaces	<code>P2_LIN_Init</code> <code>P2_LIN_Init_Write</code> <code>P2_LIN_Init_Apply</code> <code>P2_LIN_Reset</code>
Query LIN interface version	<code>P2_LIN_Get_Version</code>
Read or send data	<code>P2_LIN_Read_Dat</code>
Send LIN header	<code>P2_LIN_Ch_Read_Cnt</code> <code>P2_LIN_Msg_Read_Status</code> <code>P2_LIN_Msg_Write</code> <code>P2_LIN_Msg_Transmit</code>
Set module LED	<code>P2_Check_LED</code> <code>P2_Set_LED</code> <code>P2_LIN_Set_LED</code>

5.8.9 Pro II-Profi-SL Rev. E

The module Pro II-Profi-SL Rev. E provides a fieldbus node with the functionality of a Profibus slave. All settings are done via software.

Functions description

After power-on the fieldbus node must be initialized. The initialization determines the station address (slave node address) on the profibus as well as the size of the input and output areas.

There is a range each for data input and data output; each range has a maximum size of 76 bytes. Please note, that the terms "input" and "output" are used as the fieldbus controller sees them.

You set the number and length of input and output areas separately.

Hardware

The pin assignment of the 9-pin DSUB connector refers to DIN E 19245, part 3.

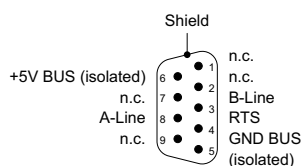


Fig. 153 – Pro II-Profi-SL Rev. E: Pin assignment

The Profibus has to be terminated at its physical beginning and at the end of its segments by an active terminator.

If required, you have to add the terminator yourself at the appropriate data lines of the fieldbus node or use an appropriate connector with integrated terminator.

Above and below the DSUB connector there are two LEDs, which display the operation status of the fieldbus node: operation mode (OP) and interface status (ST).

LED	Status	Meaning
OP	off	Offline or no power.
	green	Fieldbus node online, data exchange.
	flashing green	Fieldbus node online, status clear.
	flashing red, 1 flash	Error: Input/output configuration does not fit to master configuration.
	flashing red, 2 flashes	Error in Profibus configuration.
ST	off	Offline or no power.
	green	initialized.
	flashing green	initialized, diagnostic event(s) present.
	red	Exception error.

Fig. 154 – Pro II-Profi-SL Rev. E: Meaning of LEDs

Projecting the Profibus

You are projecting the Profibus with a configuration tool suitable for the bus master. The following process description uses a Profibus master of the Hilscher company and the appropriate program SyCon.

The process description is valid for other configuration tools, correspondingly. Look for the exact process description of bus projection in the documentation of the configuration tool.

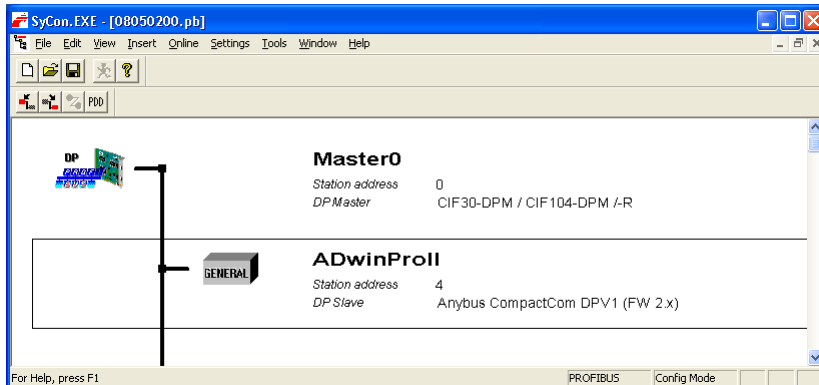
- Copy or import the GSD file `hmsb1811.gsd` of the fieldbus node from `C:\ADwin\Fieldbus\Profibus` into the source directory of the configuration tool.

The configuration tool loads all required information about the new slave from the appropriate GSD file; the file content is determined by EN 50170. Afterwards, the slave can be accessed by any master.

Copy the GSD file

- In the configuration tool, add the Slave, i.e. the fieldbus node to the Profibus by selecting the GSD file `hmsb1811.gsd`. The station address must equal the address used for *ADbasic* initialization with `P2_Init_Profibus`.

Afterwards the bus could be structured as below:



- Configure number and length of input and output data of input and output data in the fieldbus node memory one by one.

Please note the following rules:

- The terms "input" and "output" have reverse meanings in *ADbasic* (slave) and in the configuration tool (master).
- Configure the outputs (as seen from the master) first and then the inputs. If there are inputs initialized in *ADbasic*, you have to configure outputs as correspondent in the configuration tool.
- Number and length of data ranges must equal the data used for *ADbasic* initialization with `P2_Init_Profibus`.
- You may use only a single data length for each input data and output data. Input and output data can be set to a length of 1, 2, 4 or 8 Byte (2 byte = 1 word).

The following example line in *ADbasic* configures the slave with 2 inputs of 1 byte and 3 outputs of 1 byte.

```
Par_31 = P2_Init_Profibus(2, 2, 1, 3, 1, conf_Arr, Data_1)
```

To configure the slave correctly in the configuration tool you have to set the 2 outputs first and the 3 inputs afterwards (1 byte each). The graphic below shows this example configuration:

Module	Inputs	Outputs	In/Out	Identifier
Input 1 byte	1 Byte			0x90
Input 1 word	1 Word			0xD0
Input 2 words	2 Word			0xD1
Input 4 words	4 Word			0xD3
Output 1 byte		1 Byte		0xA0
Output 1 words		1 Word		0xE0

Module Revisions

The differences between the revisions are described below:

Revision	Output date	Previous changes
E1	July 2008	First version

Integrate the Slave

Configure the Slave

**Operating modes of
 fieldbus node**

Programming with ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Initialize station address and data ranges	<code>P2_Init_Profibus</code>
Read and write data	<code>P2_Run_Profibus</code>

Initialization must be run with low priority since it takes some seconds; if it were a process with high priority, the PC interrupts the communication after a time (time-out). For the same reason, reading and writing data should be run with low priority.

Specifications

The fieldbus node is in agreement with the European Standard EN 50170, Volume 2. This norm is provided by the Profibus user organization:

Profibus Nutzerorganisation e.V.
 Haid-und-Neu-Str. 7
 76131 Karlsruhe, Germany
 Phone: +49-72196-58590
 Fax : +49-72196-58589
 Order number: 0.042

The following table shows the operating modes, the fieldbus node supports and its behavior:

Operating mode	Behavior
Operate	The Profibus slave is part of the cyclic data exchange. Input data are transferred to the master via bus and output data are made ready for the master to transfer them.
Clear	The inputs are updated and the outputs are set to zero.
Stop	The slave is no longer part of the bus communication.

Fig. 155 – Pro-Profi-SL Rev. E: Operating modes

5.8.10 Pro II-Profinet Rev. E

The module Pro II-Profinet Rev. E provides a fieldbus node with the functionality of a Profinet-IRT slave. All settings are done via software.

The module is available with different connectors:

- *Pro II-Profinet-IRT-Cu* Rev. E: Interface with copper cable, 2 sockets RJ-45, customary connectors.
- *Pro II-Profinet-IRT-FO* Rev. E: Interface with optical fiber, 2 duplex sockets SC-RJ (fiber optics).

Functions description

After power-on the fieldbus node must be initialized. The initialization determines the size of the input and output areas.

There is a range each for data input and data output; each range has a maximum size of 1280 bytes. Please note, that the terms "input" and "output" are used as the fieldbus master sees them.

During initialization, you set the number and size of input and output areas separately. Nevertheless, during operation only one size can be used.

TiCo processor

The module provides a freely programmable *TiCo* processor, which has full access to the fieldbus node, as does the *ADwin* CPU. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

The *TiCo* processor (type *TiCo1*) has a clock frequency of 50MHz and is equipped with 56KiB memory: 28KiB PM, 28KiB DM.

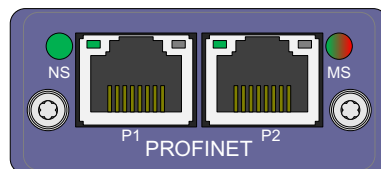
If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

Hardware

The connectors are connected to standard plugs:

- Ethernet plugs RJ-45 (IRT-Cu)

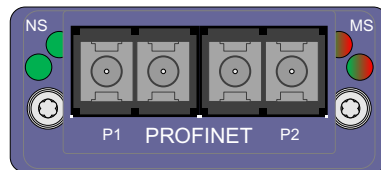
Left and right of the RJ-45 connectors, there are two LEDs, which display the operation status of the profinet node: network status (NS) and interface status (MS).



In each connector there is a LINK LED.

- Fiber duplex plugs SC-RJ (IRT-FO)

Left and right of the connectors, there are four LEDs. The half concealed LEDs display the operation status of the profinet node: network status (NS) and interface status (MS).



The LEDs more outside display the LINK status for the plug.

LED	Status	Meaning
NS	off	Offline: No power or no connection with IO controller.
	green	Online (RUN): Field bus node online, IO Controller in RUN state.
	green, 1 flash	Online (STOP): Field bus node online, IO Controller in STOP state or IO data bad. IRT synchronization not finished.
	green, blinking	Blink: Used by engineering tools to identify the node on the network.
	red	Fatal event: Major internal error. Indication is combined with a red MS LED.
	red, 1 flash	Station name error: Station name is not set.
	red, 2 flashes	IP address error: IP address is not set.
	red, 3 flashes	Configuration error: Expected identification differs from real identification.
MS	off	Not initialized: No power or module in SETUP or NW_INIT state.
	green	Normal operation.
	green, 1 flash	Diagnostic events present.
	red	Device in state EXCEPTION.
	red	Fatal event: Major internal error. Indication is combined with a red NS LED.
	red / green alternating	Firmware update. Do not power off the device. Turning the device off during this phase could cause permanent damage.
LINK	off	No link.
	green	Ethernet link established, no communication.
	green flickering	Ethernet link established, communication present.

Fig. 156 – Profinet: Meaning of LEDs

Projecting the Profinet

You are projecting the Profinet bus with a configuration tool suitable for the bus master. The following process description uses a Profinet master of the Siemens company and the appropriate program *SIMATIC-Manager*.

The process description is valid for other configuration tools, correspondingly. Look for the exact process description of bus projection in the documentation of the configuration tool.

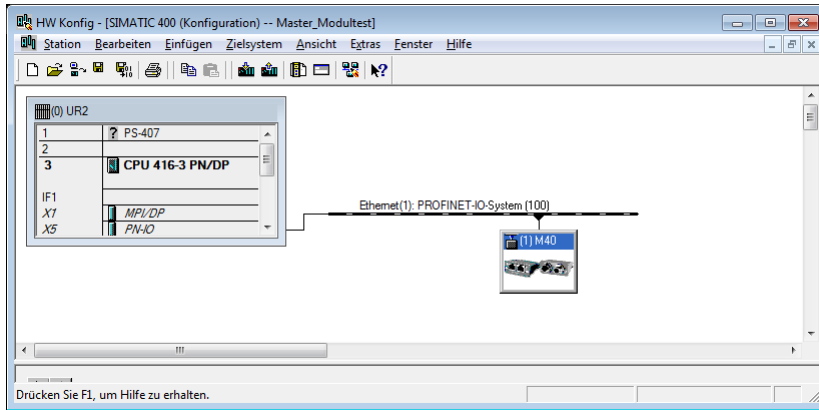
- In the program *SIMATIC-Manager*, install the GSD file *GSDML-V2.33-HMS-ABCC40-PIR-ADwin-20180620.xml* of the fieldbus node from *C:\ADwin\Fieldbus\Profinet*.

The configuration tool loads all required information about the new slave from the appropriate XML file; the file content is determined by EN 50170. Afterwards, the slave can be accessed by any master.

- In the configuration tool, add the Slave, i.e. the fieldbus node to the Profinet.

Afterwards the bus could be structured as below:

Install the GSD file



- Configure number and length of input and output data of input and output data in the fieldbus node memory one by one.

Please note the following rules:

- The terms "input" and "output" have reverse meanings in *ADbasic* (slave) and in the configuration tool (master).
If there are inputs initialized in *ADbasic*, you have to configure outputs as correspondent in the configuration tool.
- Number and length of data ranges must equal the data used for *ADbasic* initialization with `P2_Init_ProfinetIO`.
- The data range sizes for input and output can be set individually. Data ranges can be configured in one of the following sizes (1 double word = 4 byte):
1, 2, 4, ... 64 double words; 64, 128, ... 320 double words.
The blocks of 64 double words are numbered and may only be configured in ascending order IN0...IN5 / OUT0...OUT5.

The following example line in *ADbasic* configures the slave with an input range and an output range of 320 double words each (=1280 bytes):

```
P2_Init_ProfinetIO(module, 320, 320, work_arr)
```

To configure the slave correctly in the configuration tool, you have to set 5 blocks of 256 bytes (=64 double words) in both input range and output range. Please note the required order of block configuration:

Steckplatz	Baugruppe	Bestellnummer	E-Adresse	A-Adresse	Diagnoseadresse
0	M40	ABCC40.P1R			16378*
X7	Interface				16377*
P1	Port 1				16376*
P2	Port 2				16375*
1	IN0064UINT32		0..255		
2	IN1064UINT32		256..511		
3	IN2064UINT32		512..767		
4	IN3064UINT32		768..1023		
5	IN4064UINT32		1024..1279		
6	OUT0064UINT32			0..255	
7	OUT1064UINT32			256..511	
8	OUT2064UINT32			512..767	
9	OUT3064UINT32			768..1023	
10	OUT4064UINT32			1024..1279	
11					

Configure the Slave

Example configuration

**Operating modes of
 fieldbus node**

Programming with ADbasic

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Reset, initialize data ranges	<code>P2_Init_ProfinetIO</code>
Read and write data, message handling	<code>P2_Run_ProfinetIO</code>

Specifications

The fieldbus node is in agreement with the Standard IEC 61158 (Profinet). This norm is provided by the Profibus user organization:

ProfibusNutzerorganisation e.V.
 Haid-und-Neu-Str. 7
 76131 Karlsruhe, Germany
 Phone: +49-72196-58590
 Fax : +49-72196-58589
www.profibus.com

The following table shows the operating modes, the fieldbus node supports and its behavior:

Operating mode	Behavior
Setup	Interface initialization.
Wait	Slave waits for bus start by master.
Active	Profinet slave is part of the cyclic data exchange.
Error	Error. Profinet slave is not part of the cyclic data exchange.
Exception	Major internal error. Profinet slave is not part of the cyclic data exchange.

Fig. 157 – Profinet: Operating modes

5.8.11 Pro II-MIL-1553 Rev. E

The module Pro II-MIL-1553 Rev. E provides a multi-function interface for the MIL-STD-1553 bus. Currently, each MIL interface can be configured as bus monitor in SMT 16-bit mode (simple monitoring terminal). Other modes as bus controller, remote terminal, or other bus monitor modes can be configured by setting appropriate device registers.

MIL-STD-1553 Interface

The MIL interfaces of the module are implemented according to specification MIL-STD-1553B. The module is based on the MIL multi-terminal HI-6130.

You program the SMT mode of the MIL interfaces with *ADbasic* instructions. For other modes, set the appropriate registers using the separate documentation "HI-6130 / MIL-STD-1553 / BC/MT/RT Multi-Terminal Device" by Holt Integrated Circuits Inc.

MIL-STD-1553 is a serial data bus, commonly used in spacecraft on-board data handling subsystems, both military and civil. The bit transfer rate is 1.0 megabit per second.

The bus concept refers to a single Bus Controller (BC) with multiple Remote Terminals (RT) as slave nodes. Additionally, there may also be one or more Bus Monitors (BM), which do not take part in data transfers, and are only used to capture or record data.

All transmissions onto the data bus are accessible to the BC and all connected RTs. Messages consist of one or more 16-bit words (command word, data words, or status word). The 16 bits comprising each word are transmitted using Manchester code.

The BC controls the total data transfer of the bus, using commands from the BC to the RTs to receive or transmit.

Before each data packet the BC sends a header with the identifier of the next data packet. Then, only this bus node will react (which can also be the master node itself), which manages a message box with the given identifier. Thus, this node will send a data packet to or receive a data packet from the MIL bus.

An application or function in the subsystem behind the RT interface (e.g. RT1) writes the data that is to be transmitted into a specific (transmit) sub-address.

If configured as bus monitor in SMT mode, the module stores message commands, terminal responses and message data of the MIL bus.

The SMT can utilize 16-bit time tags with a range of clocking options.

After power-up, the SMT records every message on the MIL bus. You can set filters to select messages for monitoring. The filters are based on each command's RT address and subaddress, and the transmit/receive bit status.

The SMT monitor stores a message in 2 separate buffers:

- Command buffer: contains 4 words for each MIL message.
 - Command word of the BC that initiated the message. For an RT-RT message, the Receive command word is stored here; Transmit command word 2 is the first stored word in the data buffer.
 - Pointer to the words of the message in the data buffer; not required, since the *ADbasic* instruction returns the message data.
 - Time tag (16 bit) of the message.

MIL bus

SMT mode

- SMT block status word, additional information about message status, the used bus A/B and possibly occurred errors.
- Data buffer: Data words of the message.
 The number of words for each message depends on the MIL message type, ranging from zero (broadcast mode command without data) to 35 words (for a 32 data word RT-RT command).

The information of the recently stored message can be read using **P2_MIL_SMT_Message_Read**. The instruction returns the message data in 2 arrays: array 1 contains 4 words of the command buffer, array 2 contains the data words of the data buffer.

You can only read the most recent message information. As soon as a new MIL message is completely received and stored, any previous message information is lost.

Hardware design

The front panel provides 2 concentric 3-lug triax bayonet jacks, which are referred to as "Bus A" and "Bus B". For each connector there are 2 programmable LEDs.

The board contains several DIP switches (S1...S8, S13), which are required only for the use of the interface as bus controller or remote terminal. With SMT mode, all DIP switches should be set to OFF position.

Non suitable DIP switch settings can cause hardware damage. If you need to change the settings please contact the support of Jäger Messtechnik; you find the address on the inner side of the cover page of the manual.

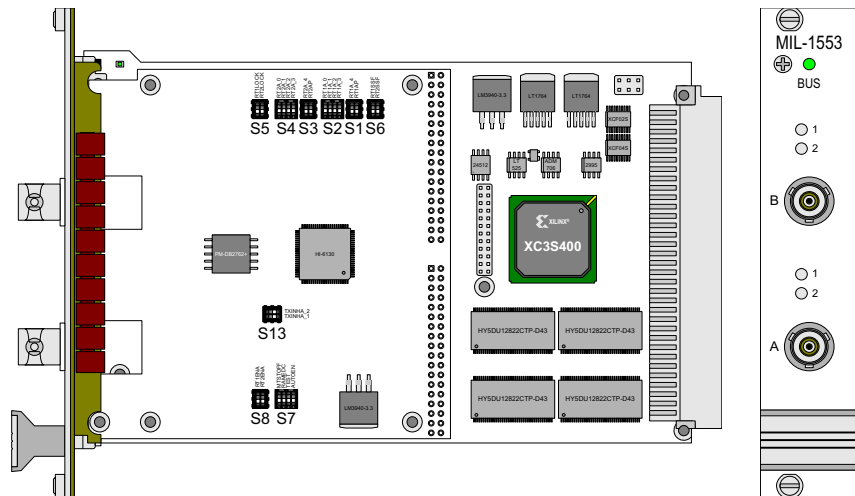


Fig. 158 – Pro II-MIL-1553 Rev. E: PCB and front panel

DIP switch no.	DIP	setting in SMT mode	DIP switch no.	DIP	setting in SMT mode
S5	RT1LOCK	OFF	S6	RT1SSF	OFF
S5	RT2LOCK	OFF	S6	RT2SSF	OFF
S2	RT1A_0	OFF	S4	RT2A_0	OFF
S2	RT1A_1	OFF	S4	RT2A_1	OFF
S2	RT1A_2	OFF	S4	RT2A_2	OFF
S2	RT1A_3	OFF	S4	RT2A_3	OFF
S1	RT1A_4	OFF	S3	RT2A_4	OFF
S1	RT1AP	OFF	S3	RT2AP	OFF

DIP switch no.	DIP	setting in SMT mode	DIP switch no.	DIP	setting in SMT mode
S8	RT1ENA	OFF	S13	TXINHA_1	OFF
S8	RT2ENA	OFF	S13	TXINHA_2	OFF
S7	MTSTOFF	OFF			
S7	RAMEDC	OFF			
S7	TEST	OFF			
S7	AUTOEN	OFF			

You find a description of the DIP switches in the separate documentation "HI-6130 / MIL-STD-1553 / BC/MT/RT Multi-Terminal Device" by Holt Integrated Circuits Inc.

Module Revisions

The differences between the revisions are described below:

Revision	Output date	Previous changes
E1	Dec. 2012	First version

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Initialize and reset MIL bus interfaces	<code>P2_MIL_Reset</code> <code>P2_MIL_SMT_Init</code>
SMT: Read bus message	<code>P2_MIL_SMT_Message_Read</code>
Set SMT filters	<code>P2_MIL_Set_All_Filters</code> <code>P2_MIL_Set_Filter</code>
Set module LEDs	<code>P2_Check_LED</code> <code>P2_Set_LED</code> <code>P2_MIL_Set_LED</code>
Set / read register of the MIL interface	<code>P2_MIL_Set_Register</code> <code>P2_MIL_Get_Register</code>

5.8.12 Pro II-ARINC-429 Rev. E

The module Pro II-ARINC-429 Rev. E provides an interface for the ARINC 429 serial bus with one transmitter and 2 receivers. Transmitter and receivers can be run in parallel at the same time.

ARINC 429 interface

The interface of the module is implemented according to specification ARINC 429. The module is based on the interface HI-3582A by Holt Integrated Circuits Inc.

You program the ARINC interface with *ADbasic* or *TiCoBasic* instructions.

ARINC 429 is a two-wire serial bus and the most widely used data bus standard for aviation. The bus operates with one transmitter and up to 20 receivers. The bit transfer rate is 100 kBit/s (high speed) or 12.5 kBit/s (low speed).

ARINC 429 uses a self-clocking, self-synchronizing data bus protocol; Tx (transmit) and Rx (receive) are on separate ports. The physical connection wires are twisted pairs, named A and B.

Data can be transmitted in one direction only—simplex communication—with bi-directional transmission requiring two channels or buses. The devices, line replaceable units (LRU), are most commonly configured in a star or bus-drop topology. Each LRU may contain multiple transmitters and receivers communicating on different buses.

Data words on the bus are 32 bits in length, and most messages consist of a single data word. The transmitter constantly transmits either 32-bit data words or the NULL state while receivers are monitoring the bus messages.

An ARINC 429 data word consists of 32 bits in 5 fields:

- label, 8 bits
- source/destination identifier (SDI), 2 bits
- data, 19 bits
- sign/status matrix (SSM), 2 bits
- parity, 1 bit

Module features

The module Pro II-ARINC-429 Rev. E provides one transmitter and 2 receivers, each of which using its own Fifo. Each Fifo can store 32 words maximum.

The module Pro II-ARINC-429 Rev. E provides the following features:

- Transmitter parity check

The transmitter parity check can be disabled. If enabled, the parity check can set to even parity or odd parity.

Please note, that module receivers will always check for odd parity. Receiver parity check cannot be disabled.

- Label matching

Each of the 2 receivers can be programmed with up to 16 labels. If enabled, a receiver will only accept ARINC messages, which match one of the programmed labels. Only accepted messages are copied to the receiver fifo.

Label matching can be enabled for each receiver independently.

– SDI decoding

Each of the 2 receivers can be programmed with an SDI value. If SDI decoding is enabled, a receiver will only accept ARINC messages, which match the set SDI value. Only accepted messages are copied to the receiver fifo.

SDI decoding can be enabled for each receiver independently.

SDI decoding and label matching can be enabled independently. If both are enabled, an incoming value must match both conditions to be accepted.

Hardware design

The front panel provides a 25-pin D-Sub plug.

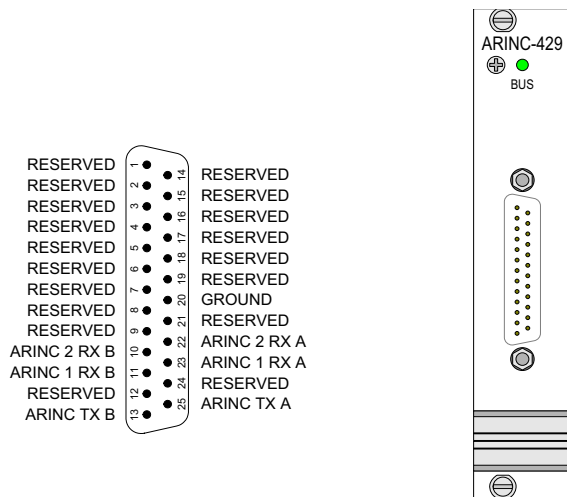


Fig. 159 – Pro II-ARINC-429 Rev. E: Pinout and front panel

Function	ARINC 429 interface with 1 transmitter, 2 receivers
Bit transfer rate	Selectable: 100 kBit/s (high speed) or 12.5 kBit/s (low speed)
Interface based on	HI-3582A by Holt Integrated Circuits Inc.
TiCo	Processor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal.
Connector	25-pin D-Sub female connector

Fig. 160 – Pro II-ARINC-429 Rev. E: Specification

Module Revisions

Revision	Output date	Previous changes
E1	Mar. 2013	First version

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

Programming in ADbasic

**Programming in
 TiCoBasic**

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Reset and configure the ARINC-429 bus interface	<code>P2_ARINC_Reset</code> <code>P2_ARINC_Config_Transmit</code> <code>P2_ARINC_Config_Receive</code> <code>P2_ARINC_Set_Labels</code>
Transmit a message	<code>P2_ARINC_Transmit_Enable</code> <code>P2_ARINC_Transmit_Fifo_Full</code> <code>P2_ARINC_Transmit_Fifo_Empty</code> <code>P2_ARINC_Write_Transmit_Fifo</code> <code>ARINC_Create_Value32</code>
Receive a message	<code>P2_ARINC_Receive_Fifo_Empty</code> <code>P2_ARINC_Read_Receive_Fifo</code> <code>ARINC_Split_Value32</code>
Set module LEDs	<code>P2_Check_LED</code> <code>P2_Set_LED</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `ARINC_TiCo.inc` contains instructions for the following functions:

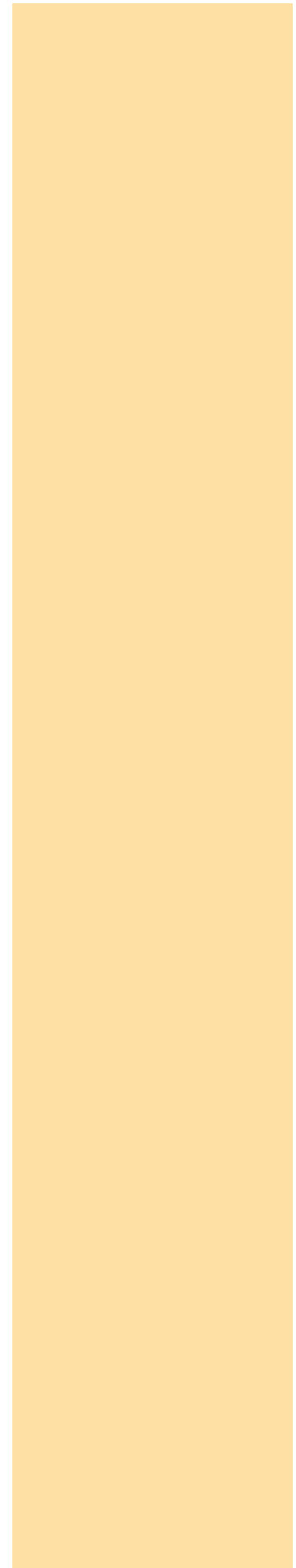
Function	Instructions
Reset and configure the ARINC-429 bus interface	<code>ARINC_Reset</code> <code>ARINC_Config_Transmit</code> <code>ARINC_Config_Receive</code> <code>ARINC_Set_Labels</code>
Transmit a message	<code>ARINC_Transmit_Enable</code> <code>ARINC_Transmit_Fifo_Full</code> <code>ARINC_Transmit_Fifo_Empty</code> <code>ARINC_Write_Transmit_Fifo</code> <code>ARINC_Create_Value32</code>
Receive a message	<code>ARINC_Receive_Fifo_Empty</code> <code>ARINC_Read_Receive_Fifo</code> <code>ARINC_Split_Value32</code>
Set module LEDs	<code>Check_LED</code> <code>Set_LED</code>

**Programming TiCo
 access**

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>

Function	Instructions
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset</code> , <code>P2_TiCo_Start</code> , <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_Status</code> <code>P2_Get_TiCo_Status</code> , <code>P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash</code> , <code>P2_TiCo_Load</code>



5.8.13 Pro II-FlexRay-2 Rev. E

The module Pro II-FlexRay-2 Rev. E is equipped with 2 FlexRay interfaces, each interface represents a complete FlexRay bus with 2 channels.

The module can be configured to start a FlexRay bus without any other bus members. To do so, both FlexRay interfaces have to run as coldstarter nodes and the interfaces have to be combined to a FlexRay bus via DIP switches.

You can also set the bus termination for each interface and channel via DIP switch.

The description of the FlexRay module is divided into the following sections:

- FlexRay Controller
- Hardware
- Module Revisions
- Programming

FlexRay Controller

The module Pro II-FlexRay-2 is equipped with two FlexRay controllers MFR4310 from FreeScale® and runs according to the "FlexRay Communications System Protocol Specification V2.1". You program the interfaces with *ADbasic* instructions, which directly access the controller registers.

For configuration and status display of the FlexRay controllers you use the appropriate registers. Here you can set all FlexRay parameters as e.g. bus speed, bus timing, etc.

You find more information on the web site <http://www.freescale.com> in the following documentations from FreeScale®:

- Engineering Bulletin EB683: MFR4310 and MFR4310 differences
- Data sheet MFR4300: MFR4300 FlexRay Communication Controller

Hardware

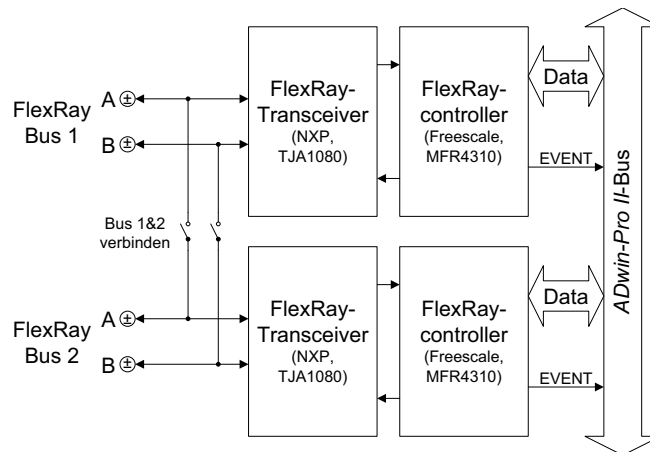


Fig. 161 – Pro II-FlexRay-2 Rev. E: Block diagram

The connections of the FlexRay interfaces are provided on a 9-pole DSUB connector; the pin assignment is shown below.

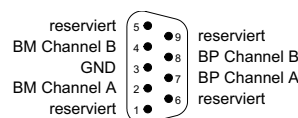


Fig. 162 – Pro II-FlexRay-2: Pin assignment

If an interface runs as physical end of a FlexRay bus, the bus must be terminated at this interface with the appropriate DIP switch (see fig. 163); the channels A and B can be terminated separately. Since both channels are run differential, you always have to switch both (!) DIP switches for each channel to the right for termination.

If the FlexRay module is not located at the end of the bus you may not terminate the interface.

Both FlexRay interfaces can be combined to a startable FlexRay cluster via DIP switches. For combination of the channels A and B you always have to switch both (!) DIP switches for each channel upwards (see fig. 163).

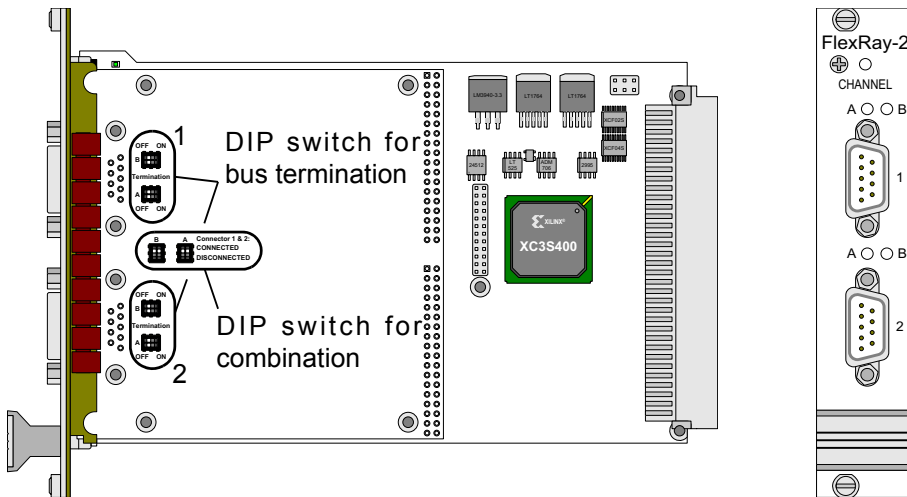


Fig. 163 – Pro II-FlexRay-2 Rev. E: PCB and front panel

Module Revisions

The differences between the revisions are described below:

Revision	Output date	Previous changes
E1	08 / 2009	First version

Programming

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Initialize a FlexRay interface	<code>P2_FlexRay_Init</code>
Reset a FlexRay controller	<code>P2_FlexRay_Reset</code>
Query interface version	<code>P2_FlexRay_Get_Version</code>
Set and read controller register	<code>P2_FlexRay_Read_Word</code> <code>P2_FlexRay_Write_Word</code>
Set module LED	<code>P2_Check_LED</code> <code>P2_Set_LED</code> <code>P2_FlexRay_Set_LED</code>

5.8.14 Pro II-EtherCAT-SL Rev. E

The module Pro II-EtherCAT-SL Rev. E provides a fieldbus node with the functionality of an EtherCAT slave. All settings are done via software.

Functions description

After power-on you must initialize the fieldbus node in *ADbasic*. The initialization determines the size of the input and output areas.

There is a range each for data input and data output; each range has a size of 16 Longs or 64 bytes. Please note, that the terms "input" and "output" are used as the fieldbus controller sees them.

Hardware

The interface has a plug connector of type RJ45 for both data input (IN) and data output (OUT). Each connector has a LED "Link / Activity" top right, which displays the operating status of the node in the EtherCAT bus. The two other LEDs (at the bottom of the plug) have no function.

Underneath the connectors, there are LEDs displaying the status of the EtherCAT state machine (RUN) and the occurrence of communication errors (ERR).



LED	Status	Meaning
Link / Activity	off	Offline (or no power).
	green	Fieldbus node online, no data exchange.
	green, flickering	Fieldbus node online, with data exchange.
RUN	off	Status INIT: interface being initialized (or no power).
	blinks green	Status PRE-OP: Interface has contact to bus master.
	flashes green once	Status SAFE-OP: Interface can read data from the bus, but not send.
	green	Status OP: Interface is completely ready, inputs and outputs are active.
	red	Status EXCEPTION.
ERR	off	No error (or no power).
	blinks red	Invalid configuration.
	flashes red once	Local error in the interface; EtherCAT status has been changed.
	flashes red twice	Application watchdog timeout.
	red	Critical communication error.

Fig. 164 – Pro II-EtherCAT-SL Rev. E: Meaning of LEDs

If both LEDs *RUN* and *ERR* turn red, a serious error has occurred in the interface. Please inform the support of Jäger Messtechnik; you find the address on the inner side of the cover page of the manual.

Projecting the EtherCAT bus

You are projecting the EtherCAT bus with a configuration tool suitable for the bus master. The following process description uses the program "TwinCAT System Manager" of the Beckhoff company as EtherCAT bus master.

The process description is valid for other configuration tools, correspondingly. Look for the exact process description of bus projection in the documentation of the configuration tool.

- Copy the description file *ADwin-EtherCAT.xml* of the fieldbus node from *C:\ADwin\Fieldbus\EtherCAT* into the root directory of the configuration tool.

Upon start-up, the configuration tool loads the required information about the new slave from the appropriate description file.

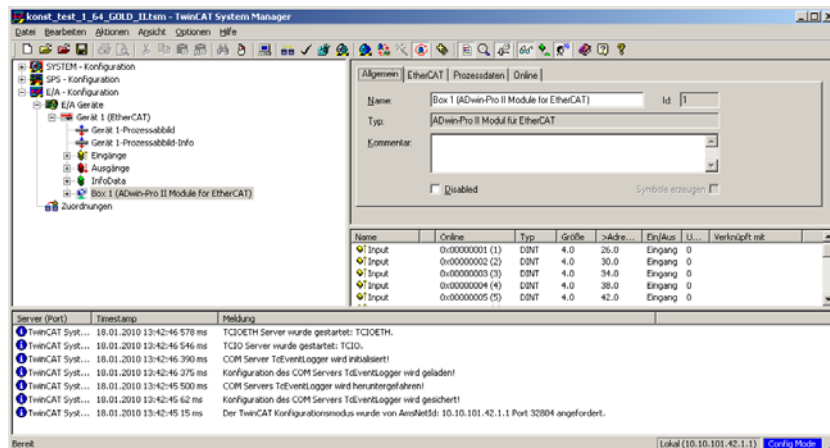
- Add the *ADwin*-EtherCAT slave as bus member to the EtherCAT bus.

Using TwinCAT System Manager, you mark the EtherCAT master and select the menu entry *Scan boxes* from the context menu (right mouse click).

A list of all current bus members will be displayed.

- Select the *ADwin*-EtherCAT slave from the list; now the slave is confirmed as bus member.

Afterwards the bus could be structured as below:



- Configure the *ADwin*-EtherCAT slave in an *ADbasic* program using the instruction `P2_ECAT_Init`.

Though you can configure the slave from the configuration tool as well, the configuration in *ADbasic* has to be processed nevertheless—using the very same settings.

- Read the configuration into the configuration tool.

Using the TwinCAT System Manager, you mark the *ADwin*-EtherCAT slave and clicken the button *Load PDO Info* from the device.

The slave configuration is in any case as follows: 16 DINT (4 byte) as inputs and 16 DINT (4 byte) as outputs.

Module revisions

The difference between the revisions is described below:

Revision	Release date	Previous changes
E	12/2009	First version

The module revision is to be found on the front cover.

Programming with *ADbasic*

The module is comfortably programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Area	Instructions
Initialize station address and data ranges	<code>P2_ECAT_Init</code>
Get current interface version and operation mode	<code>P2_ECAT_Get_Version</code> <code>P2_ECAT_Get_State</code>
Read and write data in groups of 16 longs.	<code>P2_ECAT_Read_Data_16L</code> <code>P2_ECAT_Write_Data_16L</code>

**Operating modes of
EtherCAT node**

Initialization must be run with low priority since it takes some seconds; if it were a process with high priority, the PC interrupts the communication after a time (time-out). For the same reason, reading and writing data should be run with low priority.

Specifications

The fieldbus node is in agreement with the international standard IEC 61158 and IEC 61784-2. More information is provided by the EtherCAT user organization:

EtherCAT Technology Group
Ostendstraße 196
D-90482 Nürnberg
Tel.: +49 9115405620
Fax : +49 9115405629
<http://www.ethercat.org/>

The following table shows the operating modes, the EtherCAT node supports and its behavior:

Operating mode	Behavior
Init	The EtherCAT slave is being initialized by the bus master.
PreOp	The interface is part of the data exchange, inputs and outputs are not active.
SafeOp	The interface can receive data, outputs are not active.
Op	The interface is completely ready; inputs and outputs are active.

Fig. 165 – EtherCAT: Operating modes

5.8.15 Pro II-SENT-4 Rev. E

The module Pro II-SENT-4 Rev. E provides 4 input channels using the SENT protocol (Single Edge Nibble Transmission) and analyzes the incoming SENT signals on all 4 channels automatically.

The description of the SENT module is divided in the following sections:

- [Hardware design](#)
- [Working with SENT](#)
- [Modul revision](#)
- [Programming](#)

SENT channels	4 inputs
Signal voltage	Low level < 0.5V, High level > 4.1V
Supply voltage	5V
Clock tick range	3µs...90µs
Connector	37-pin D-Sub female connector

Fig. 166 – Pro II-SENT-4 Rev. E: Specification

Hardware design

The module Pro II-SENT-4 Rev. E works according to the specification SAE J2716. You find further information on the web site <http://standards.sae.org> of SAE International®.

The connections of the SENT interface are on the 37-pin D-SUB connector; the pin assignment is shown below.

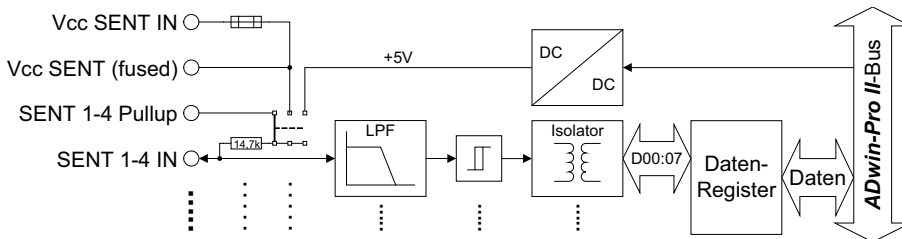


Fig. 167 – Pro II-SENT-4 Rev. E: Block diagram

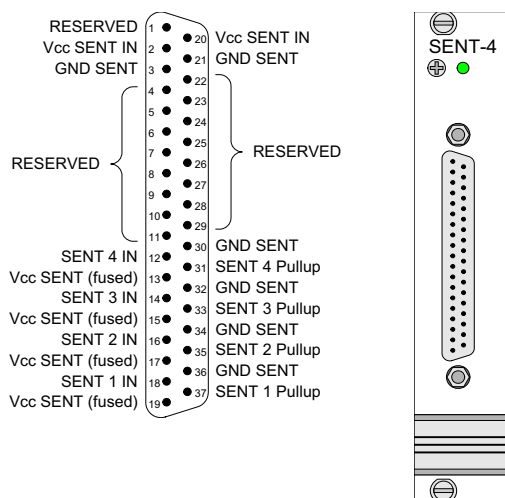


Fig. 168 – Pro II-SENT-4 Rev. E: Pin assignment and front panel

Supply voltage

Via the pins V_{CC} SENT In you can insert a supply voltage 5 Volt of up to 400mA; the pins V_{CC} SENT In are connected to each other. The voltage is provided on the 4 voltage output pins V_{CC} SENT (fused) and also can be used as pull-up-voltage (see below). A fuse of 500mA is placed between voltage inputs and outputs.

Please note: In rev. E01, the supply voltage circuitry contains a resistor 51k Ω , since rev. E02 a resistor 14.7k Ω .

Pull-up signal

The SENT channels can connected to a pull-up-signal in different ways. You select the connection by setting a jumper on the circuit board:

- A DC/DC converter on the module provides a common pull-up voltage of 5 Volt.
- The external voltage supply V_{CC} SENT In is also used as common pull-up voltage.
- External input of pull-up voltage via the inputs SENT 1...4 Pullup. There is a separate input for each SENT channel.

The pull-up voltages shall be 5 Volt.

On the printed circuit board there are 3 spare jumpers (see fig. 169, bottom left), if one of the pull-up-jumpers gets lost.

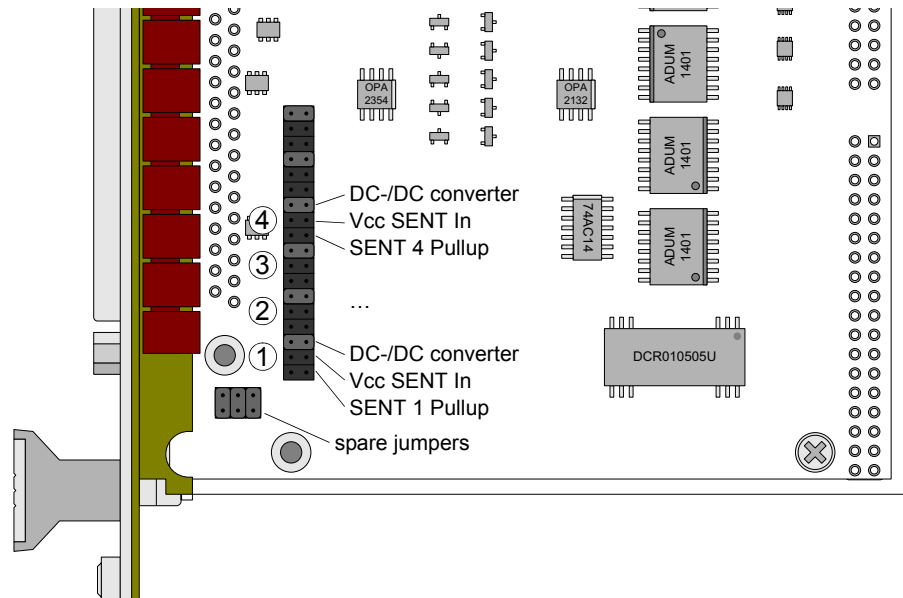


Fig. 169 – Pro II-SENT-4 Rev. E: Jumper positions on the printed circuit board

Working with SENT

The SENT protocol works as point-to-point connection, which transfers signal values from a sensor to a controller. A SENT message consists of 9 or 10 pulses:

- 1 calibration pulse for synchronization
- 1 nibble pulse (1 nibble = 4 bit): Status and communication, contains amongst others 1...2 bits of the serial message.
- 3 nibble pulses: first 12-bit value of the sensor
- 3 nibble pulses: second 12-bit value of the sensor
- 1 nibble pulse: CRC checksum of the data pulses
- optionally 1 pause pulse

Terminology

Checking timeout

You can query whether an input channel continuously receives SENT messages or not: As long as the return value of `P2_SENT_Get_Msg_Counter` keeps changing, new SENT messages are being received on the input channel.

CRC mode

With `P2_SENT_Set_CRC_Implementation`, the CRC check can be switched from calculation algorithm "legacy" to "recommended".

The calculation algorithm refers to both signals 1 and 2 as well as to the serial message.

Setting sensor type

With `P2_SENT_Set_Sensor_Type`, the expected sensor type of a SENT signal is set. The module then checks the incoming signals whether they also apply to the criteria of the selected sensor type, according to SENT specification. If not, an error ID can be queried.

At the moment, the sensor types "throttle position sensor" and "single secure sensor" are supported.

Modul revision

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E01	06 / 2012	First version
E02	10 / 2013	Change of input circuitry from "Legacy SENT System Interface Circuit Topology" (SENT Spec. Fig. 6.3.3) to "Recommended Sent System Interface Circuit Topology" (SENT Spec. Fig. 6.3.4)
E03	11 / 2013	Piggyback card with wiring variants.

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

The interface version is not visible from outside but can be queried by software:

Software version	Output date	Changes to previous version
10	06 / 2012	First official version
11	07 / 2012	Setting of CRC mode.
12	08 / 2012	Setting the sensor type of a channel according to SENT specification.
13	08 / 2012	Setting pause pulse of a SENT signal.
14	09 / 2013	Read all serial messages at once.
16	09 / 2013	Reset collected serial messages.
17	11 / 2013	New latch buffer.
18	01 / 2014	Changes in latch buffer.
19	04 / 2014	Update of include files.
20	11 / 2014	Performance optimized.

Programming

The module is programmed comfortably with *ADbasic* instruction. The instructions are described in the Pro II software manual and in the online help of *ADbasic*.

Please note: For some commands you have to check with `P2_SENT_Command_Ready`, whether the SENT interface is ready to process the next command. Such commands are marked in the following table with *.

The include file `ADwinPro_All.inc` contains the following instructions:

Bereich	Befehle
Query interface version	<code>P2_SENT_Get_Version</code>
Initialize interface	<code>P2_SENT_Init</code>
Command processing	<code>P2_SENT_Command_Ready</code>
Read SENT parameter	<code>P2_SENT_Get_ChannelState</code> <code>P2_SENT_Get_ClockTick</code> <code>P2_SENT_Get_PulseCount</code>
Set SENT parameter	<code>P2_SENT_Set_CRC_Implementation*</code> <code>P2_SENT_Set_Detection*</code> <code>P2_SENT_Set_ClockTick*</code> <code>P2_SENT_Set_PulseCount*</code> <code>P2_SENT_Set_Sensor_Type*</code>
Read SENT signals (fast channel)	<code>P2_SENT_Get_Fast_Channel1</code> <code>P2_SENT_Get_Fast_Channel2</code> <code>P2_SENT_Get_Fast_Channel_CRC_OK</code>
Read SENT serial message (slow channel)	<code>P2_SENT_Get_Serial_Message_Id</code> <code>P2_SENT_Get_Serial_Message_Data</code> <code>P2_SENT_Get_Serial_Message_CRC_OK</code> <code>P2_SENT_Get_Serial_Message_Array</code> <code>P2_SENT_Clear_Serial_Message_</code> <code>Array</code>
Collect SENT message via latch buffer	<code>P2_SENT_Request_Latch*</code> <code>P2_SENT_Check_Latch</code> <code>P2_SENT_Get_Latched_Data</code>
Check for timeout	<code>P2_SENT_Get_Msg_Counter</code>
Set module LED	<code>P2_Check_LED, P2_Set_LED</code>

5.8.16 Pro II-SENT-6 Rev. E

The module Pro II-SENT-6 Rev. E provides 6 input channels using the SENT protocol (Single Edge Nibble Transmission) and analyzes the incoming SENT signals on all 4 channels automatically. Additionally there are 4 digital inputs and 4 digital outputs with TTL levels.

The description of the SENT module is divided into the following sections:

- Hardware design
- Working with SENT
- Digital channels
- Modul revision
- Programming

SENT Interface	
SENT channels	6 inputs
Signal voltage	Low level < 0.5V, High level > 4.1V
Supply voltage	5V
Clock tick range	3µs...90µs
Digital Channels	
Digital channels	4 inputs, 4 outputs; TTL logic
Pull-Down resistor	10kΩ
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1µA
I _{IL}	max. 0.01 mA
Spannungsbereich	-0.5V ... +5.5V
Ausgangsstrom	max. ±35mA per channel, max. ±70 mA per block (4 channels) via V _{CC} or GND
General	
Connector	37-pin D-Sub female connector

Hardware design

The module Pro II-SENT-6 Rev. E works according to the specification SAE J2716. You find further information on the web site <http://standards.sae.org> of SAE International®.

The connections of the SENT interface are on the 37-pin D-SUB connector; the pin assignment is shown below.

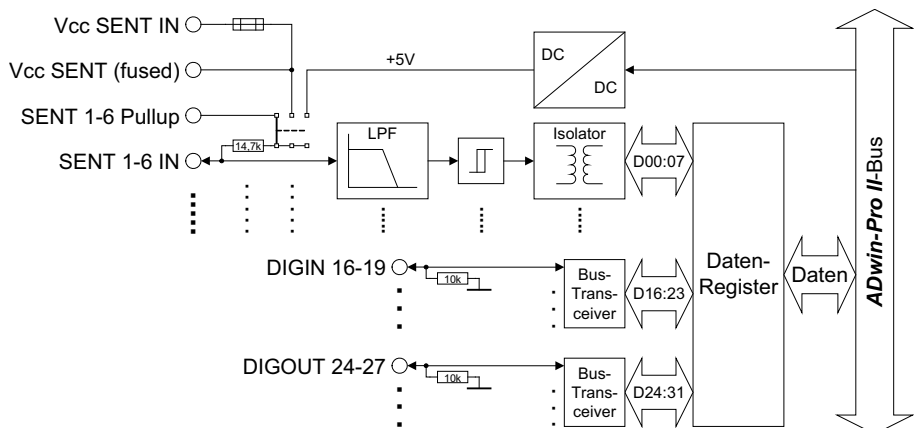


Fig. 171 – Pro II-SENT-6 Rev. E: Block diagram

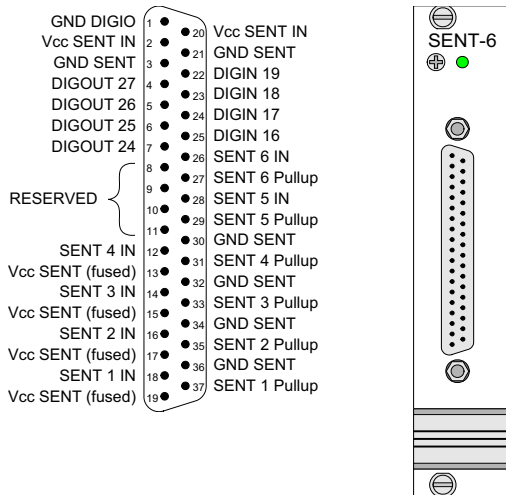


Fig. 172 – Pro II-SENT-6 Rev. E: Pin assignment and front panel

Via the connected pins $V_{CC\ SENT\ In}$, you can insert a supply voltage 5 Volt of up to 400mA. The voltage is provided on the 4 voltage output pins $V_{CC\ SENT\ (fused)}$ and also can be used as pull-up-voltage (see below). A fuse of 500mA is placed between voltage inputs and outputs.

The SENT channels can be connected to a pull-up-signal in different ways. You select the connection by setting a jumper on the circuit board:

- A DC/DC converter on the module provides a common pull-up voltage of 5 Volt.
- The external voltage supply $V_{CC\ SENT\ In}$ is also used as common pull-up voltage.
- External input of pull-up voltage via the inputs $SENT\ 1...6\ Pullup$. There is a separate input for each SENT channel.

The pull-up voltages shall be 5 Volt.

On the printed circuit board there are 3 spare jumpers (see fig. 173, bottom left), if one of the pull-up-jumpers gets lost.

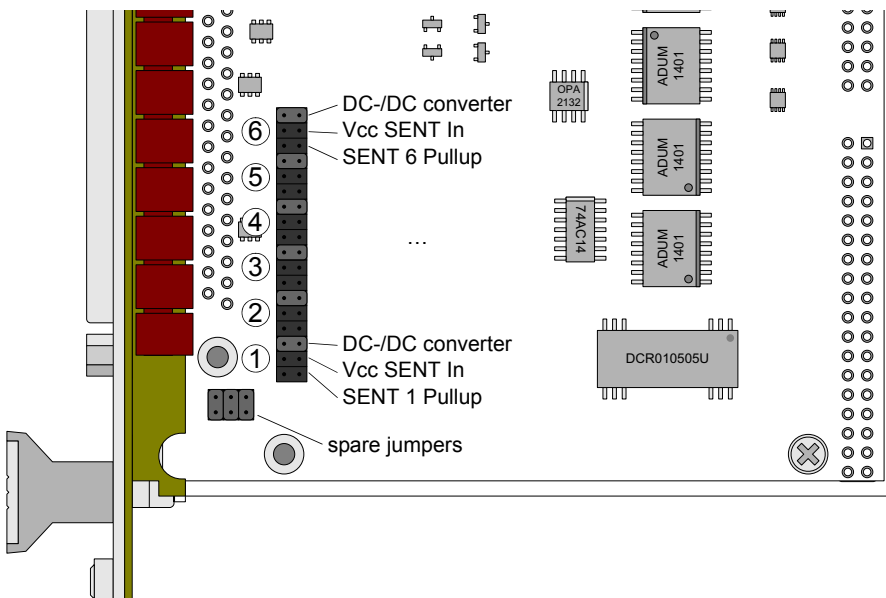


Fig. 173 – Pro II-SENT-4 Rev. E: Jumper positions on the printed circuit board



Supply voltage

Pull-up signal

Terminology	<h3>Working with SENT</h3> <p>The SENT protocol works as point-to-point connection, which transfers signal values from a sensor to a controller. A SENT message consists of 9 or 10 pulses:</p> <ul style="list-style-type: none">– 1 calibration pulse for synchronization– 1 nibble pulse (1 nibble = 4 bit): Status and communication, contains amongst others 1...2 Bits of the serial message.– 3 nibble pulses: first 12-bit value of the sensor– 3 nibble pulses: second 12-bit value of the sensor– 1 nibble pulse: CRC checksum of the data pulses– optionally 1 pause pulse <p>The length of pulses and SENT messages is measured in clock ticks. The clock tick can have a length of 3µs...90µs.</p> <p>A serial message consists of an ID, a data value and a CRC checksum. The sending format (standard, enhanced 1, enhanced 2) determines the number of bits for ID and data value. Serial messages are sent distributed to several SENT messages, only 1...2 bits per message.</p> <p>Several (up to 32) serial messages form a message set. The message set is transferred cyclically.</p>
Detection mode	<p>After power-up all input channels are set to detection mode, where incoming SENT signals are being analyzed. As soon as the module detects the clock tick and number of pulses of SENT messages, it sets the appropriate input channel to read mode.</p> <p>Alternatively, you can set the number of pulses and the clock tick of an input channel manually via software. With setting the clock tick, the input channel is set to read mode without further checks.</p> <p>You can switch an input channel from read mode back to detection mode via software. This can be useful after changing the SENT sensor.</p>
Read mode	<p>With read mode, the module evaluates the SENT signal of the input channels continuously and provides the contained data to be read:</p> <ul style="list-style-type: none">– Signals 1 and 2 (fast channel) and the result of the CRC check. A possible pause pulse of the SENT signal is automatically detected.– ID and data value of a serial message (slow channel) and the result of the CRC check. <p>The following serial message formats are detected:</p> <ul style="list-style-type: none">• short serial message format: 4-bit ID + 8-bit data value.• enhanced serial message format: 8-bit ID + 12-bit data value or 4-bit ID + 12-bit data value. <p>You can either read single serial messages consecutively or a complete message set at once. A message set contains the recently received data value and the number of received messages for each ID.</p> <ul style="list-style-type: none">– Only on request the module collects all data of the next SENT message in a latch buffer, from where you can afterwards read the data set. This ensures that all data belong to the same SENT message. <p>The data set contains amongst others all 8 nibbles as well as the time when the SENT message was received.</p>
Checking timeout	<p>You can query whether an input channel continuously receives SENT messages or not: As long as the return value of <code>P2_SENT_Get_Msg_Counter</code> keeps changing, new SENT messages are being received on the input channel.</p>

With `P2_SENT_Set_CRC_Implementation` the CRC check can be switched from calculation algorithm "legacy" to "recommended".

The calculation algorithm refers to both signals 1 and 2 as well as to the serial message.

With `P2_SENT_Set_Sensor_Type` the expected sensor type of a SENT signal is set. The module then checks the incoming signals whether they also apply to the criteria of the selected sensor type, according to SENT specification. If not, an error ID can be queried.

At the moment, the sensor types "throttle position sensor" and "single secure sensor" are supported.

Digital channels

There are 4 digital inputs (DIGIN 16...19) and 4 digital outputs (DIGOUT 24...27) on the 37-pin D-Sub connector; pin assignment see [fig. 172](#).

The channel numbers 0...15, 20...23, and 28...31 are not available.

Please note: While using digital channels, no instructions for input fifo or output fifo are allowed. Otherwise the SENT interface will work incorrectly.

Modul revision

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E03	11 / 2013	First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

The interface version is not visible from outside but can be queried by software:

Software version	Output date	Changes to previous version
17	11 / 2013	First official version
18	1 / 2014	Changes in latch buffer.
19	04 / 2014	Update of include files.
20	11 / 2014	Performance optimized.

Programming

The module is programmed comfortably with *ADbasic* instruction. The instructions are described in the Pro II software manual and in the online help of *ADbasic*.

Please note: For some commands you have to check with `P2_SENT_Command_Ready`, whether the SENT interface is ready to process the next command. Such commands are marked in the following table with *.

The include file `ADwinPro_All.inc` contains the following instructions:

Function	Instructions
SENT Interface	
Query interface version	<code>P2_SENT_Get_Version</code>
Initialize interface	<code>P2_SENT_Init</code>
Command processing	<code>P2_SENT_Command_Ready</code>
Read SENT parameter	<code>P2_SENT_Get_ChannelState</code> <code>P2_SENT_Get_ClockTick</code> <code>P2_SENT_Get_PulseCount</code>

CRC mode

Setting sensor type

Function	Instructions
Set SENT parameter	P2_SENT_Set_CRC_Implementation* P2_SENT_Set_Detection* P2_SENT_Set_ClockTick* P2_SENT_Set_PulseCount* P2_SENT_Set_Sensor_Type*
Read SENT signals (fast channel)	P2_SENT_Get_Fast_Channel1 P2_SENT_Get_Fast_Channel2 P2_SENT_Get_Fast_Channel_CRC_OK
Read SENT serial message (slow channel)	P2_SENT_Get_Serial_Message_Id P2_SENT_Get_Serial_Message_Data P2_SENT_Get_Serial_Message_CRC_OK P2_SENT_Get_Serial_Message_Array P2_SENT_Clear_Serial_Message_Array
Collect SENT message via latch buffer	P2_SENT_Request_Latch* P2_SENT_Check_Latch P2_SENT_Get_Latched_Data
Check for timeout	P2_SENT_Get_Msg_Counter
Digital Channels	
Query digital channels	P2_Digin_Long P2_Digin_Edge
Use latch register of digital channels	P2_Dig_Latch P2_Dig_Read_Latch P2_Dig_Write_Latch P2_Sync_All
Set and read back digital out- puts	P2_Digout, P2_Digout_Bits P2_Digout_Long P2_Get_Digout_Long P2_Digout_Set, P2_Digout_Reset
Set module LED	P2_Check_LED, P2_Set_LED

5.8.17 Pro II-SENT-4-Out Rev. E

The module Pro II-SENT-4-Out Rev. E provides 4 output channels using the SENT protocol (Single Edge Nibble Transmission). Additionally there are 4 digital inputs and 4 digital outputs with TTL levels.

The description of the SENT module is divided in the following sections:

- [Hardware design](#)
- [Working with SENT](#)
- [Digital channels](#)
- [Modul revision](#)
- [Programming](#)

SENT Interface	
SENT channels	4 outputs
Signal voltage	Low level < 0.5V, High level > 4.1V
Clock tick range	3µs...90µs
Digital Channels	
Digital channels	4 inputs, 4 outputs; TTL logic
Pull-Down resistor	10kΩ
V _{IH}	min. 2V
V _{IL}	max. 0.8V
I _{IH}	max. 1µA
I _{IL}	max. 0.01mA
Spannungsbereich	-0.5V ... +5.5V
Ausgangsstrom	max. ±35mA per channel, max. ±70mA per block (4 channels) via V _{CC} or GND
General	
Connector	37-pin D-Sub female connector

Fig. 174 – Pro II-SENT-4-Out Rev. E: Specification

Hardware design

The module Pro II-SENT-4-Out Rev. E works according to the specification SAE J2716. You find further information on the web site <http://standards.sae.org> of SAE International®.

The connections of the SENT interface are on the 37-pin D-SUB connector; the pin assignment is shown below.

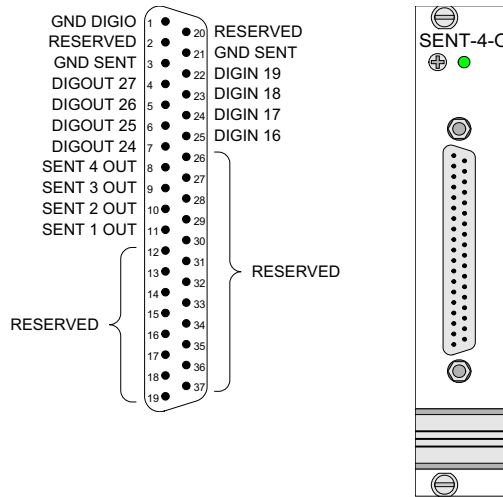


Fig. 175 – Pro II-SENT-4-Out Rev. E: Pin assignment and front panel

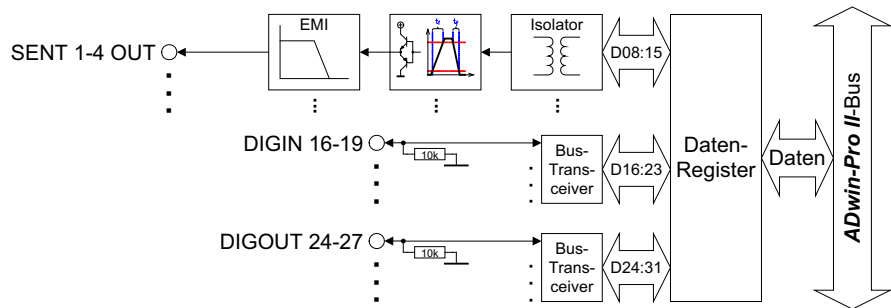


Fig. 176 – Pro II-SENT-4-Out Rev. E: Block diagram

Working with SENT

The SENT protocol works as point-to-point connection, which transfers signal values from a sensor to a controller. A SENT message consists of 9 or 10 pulses:

- 1 calibration pulse for synchronization
- 1 nibble pulse (1 nibble = 4 bit): Status and communication, contains amongst others 1...2 Bits of the serial message.
- 3 nibble pulses: first 12-bit value of the sensor
- 3 nibble pulses: second 12-bit value of the sensor
- 1 nibble pulse: CRC checksum of the data pulses
- optionally 1 pause pulse

The length of pulses and SENT messages is measured in clock ticks. The clock tick can have a length of 3µs...90µs.

A serial message consists of an ID, a data value and a CRC checksum. The sending format (standard, enhanced 1, enhanced 2) determines the number of bits for ID and data value. Serial messages are sent distributed to several SENT messages, only 1...2 bits per message.

Several (up to 32) serial messages form a message set. The message set is transferred cyclically.

The module distinguishes two operation modes for sending:

- Continuous mode: The module continuously sends the currently defined SENT message.

Terminology

Operation modes

You can change the SENT message via software: Signals 1/2 (fast channel), serial message and reserved bits. The CRC checksums are calculated automatically.

- FIFO mode: You fill a FIFO array with SENT messages (max. 400), the module reads the FIFO array, and outputs the SENT messages consecutively. As soon as the FIFO runs empty the SENT channel is disabled.

You prepare the bit order of SENT messages on your own. This refers in particular to the bits of the serial messages.

For the output of SENT messages the following parameters can be set for each channel:

- Clock tick
- Fixed total length of the SENT message via pause pulse or variable total length without pause puls
- Calculation algorithm "Legacy" or "Recommended" of the CRC checksum. The algorithm refers to both the signals 1 and 2 as well as to the serial message.
- Length of low level at the start of a pulse.
- Sending format of serial messages:
 - short serial message format: 4-bit ID + 8-bit data value.
 - enhanced serial message format: 8-bit ID + 12-bit data value or 4-bit ID + 16-bit data value.

You can query how much SENT messages have been sent on an output channel.

Digital channels

There are 4 digital inputs (DIGIN 16...19) and 4 digital outputs (DIGOUT 24 ... 27) on the 37-pin D-Sub connector; pin assignment see [fig. 175](#).

The channel numbers 0...15, 20...23, and 28...31 are not available.

Please note: While using digital channels, no instructions for input fifo or output fifo are allowed. Otherwise the SENT interface will work incorrectly.

Modul revision

Differences between revisions are described below. The revision number is located on the front panel.

Revision	Output date	Changes to previous revision
E03	11 / 2013	First version

Different revision characters mean different module characteristics and are documented separately. A counter number is added for internal purpose.

The interface version is not visible from outside but can be queried by software with `P2_SENT_Get_Version`:

Software version	Output date	Changes to previous version
1	11 / 2013	First official version
2	1 / 2014	Better handling in FIFO mode.
3	2 / 2014	Enhanced restart of SENT channels.
4	3 / 2014	Optimized performance.
5	3 / 2014	Enhanced PWM output.

Output parameters

Programming

The module is programmed comfortably with *ADbasic* instruction. The instructions are described in the Pro II software manual and in the online help of *ADbasic*.

Please note: For some commands you have to check with **P2_SENT_Command_Ready**, whether the SENT interface is ready to process the next command. Such commands are marked in the following table with *.

The include file `ADwinPro_All.inc` contains the following instructions:

Function	Instructions
SENT Interface	
Query interface version	<code>P2_SENT_Get_Version</code>
Initialize interface	<code>P2_SENT_Init</code>
Command processing	<code>P2_SENT_Command_Ready</code>
Initialize output channels	<code>P2_SENT_Set_Output_Mode</code> <code>P2_SENT_Get_Output_Mode</code> <code>P2_SENT_Config_Output*</code> <code>P2_SENT_Config_Serial_Messages*</code> <code>P2_SENT_Invert_Channel</code>
Enable / disable channels	<code>P2_SENT_Enable_Channel</code>
Continuous mode: Set SENT messages	<code>P2_SENT_Set_Fast_Channel1</code> <code>P2_SENT_Set_Fast_Channel2</code> <code>P2_SENT_Set_Reserved_Bits</code> <code>P2_SENT_Clear_Serial_Message_Array*</code> <code>P2_SENT_Set_Serial_Message_Pattern*</code> <code>P2_SENT_Set_Serial_Message_Data*</code>
Fifo mode: Set SENT messages	<code>P2_SENT_Fifo_Empty</code> <code>P2_SENT_Fifo_Clear*</code> <code>P2_SENT_Set_Fifo*</code>
Query number of messages	<code>P2_SENT_Get_Msg_Counter</code>
Digital Channels	
Query digital channels	<code>P2_Digin_Long</code> <code>P2_Digin_Edge</code>
Use latch register of digital channels	<code>P2_Dig_Latch</code> <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code> <code>P2_Sync_All</code>
Set and read back digital outputs	<code>P2_Digout, P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Get_Digout_Long</code> <code>P2_Digout_Set, P2_Digout_Reset</code>
Set module LED	<code>P2_Check_LED, P2_Set_LED</code>

5.8.18 Pro II-SPI-2 Rev. E

The module Pro II-SPI-2 Rev. E provides two independent SPI interfaces with master or slave functionality as well as several digital channels.

The module is available in 2 variants:

- ADwin-Pro II-SPI-2-T: SPI signals use TTL levels.
- ADwin-Pro II-SPI-2-D: SPI signals are differential.

Alternatively, the module can also be used as digital module with only digital inputs and outputs (similar to Pro II-DIO32-TiCo Rev. E).

Each module is equipped with a freely programmable *TiCo* processor, which has full access to the CAN interfaces. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

The manual is divided into the following sections:

- [TiCo processor](#)
- [SPI operation](#)
- [Operation mode digital module](#)
- [Hardware](#)
- [Specification](#)
- [Module revisions](#)
- [Programming](#)

TiCo processor

The module provides the freely programmable *TiCo* processor with 28KiB program memory, and 28KiB data memory. The *TiCo* processor is programmed with *TiCoBasic*.

The *TiCo* processor has—as well as the *ADwin* CPU—access to the SPI controller. Find more information about use and programming of the *TiCo* processor in the *TiCoBasic* manual.

If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the *ADwin-Pro II* system.

SPI operation

The two SPI interfaces can be operated in the following operation modes:

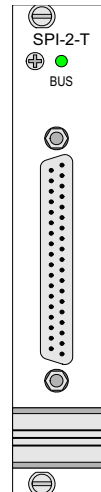
- Both interfaces as SPI master
- Both interfaces as SPI slave
- Interface 1 as SPI master, interface 2 as SPI slave
- For the use as digital module see [Operation mode digital module](#), page 215.

The operation mode is set with the software instruction `P2_SPI_Mode`.

The interfaces are each connected to the SPI bus via the lines `SCLK`, `SS`, `DATAIN` and `DATAOUT`. The pin assignments for the various operation modes are shown on [page 216](#).

In SPI mode, the module provides several digital channels:

- SPI-2-T: 10...22 channels with TTL levels, according to operation mode.
- SPI-2-D: 8 channels with TTL levels and 4 differential channels.



SPI signals

The SPI signals of the module ADwin-Pro II-SPI-2-T use 5 V TTL levels, the signals of the module ADwin-Pro II-SPI-2-D are differential.

The SPI interfaces run with a bus frequency of max. 12.5 MHz.

In theory, an unlimited number of members can be connected to the SPI bus while there has to be exactly one SPI master. The master creates the clock signal on the `SPI CLK` line and selects via a `Slave Select (SS)` line the slave he will communicate with. If the master pulls `SS` to the appropriate TTL level, the slave is activated, listens to `SPI MOSI` and sends its data to `SPI MISO` with the clock rate of `SPI CLK`, normally within the same data transfer.

Thus, an SPI message is transferred from master to slave (MOSI: master out, slave in) and another SPI message from slave to master (MISO: master in, slave out). The data transfer can happen in both directions at the same time.

The length of SPI messages can be configured freely, the maximum length of an SPI message is 64 bits (= 8 Byte).

With each clock signal, a bit of the SPI message is transferred. A common data byte requires 8 clock periods to be completely transferred. You can configure if a data transfer starts with the most (MSB) or the least significant bit (LSB). The data transfer is completed when the master permanently deactivates the slave select line.

You can insert a waiting time into the clock signal `SCLK` of the SPI master. As well you can insert a time delay for reading the MISO signal.

Generally, the slave select signal `SS out` of the master ends a defined time before the start of the clock signal `SCLK` and ends by the same time after the end of the clock signal. You can extend this time interval by software.

Interface SPI master

The module Pro II-SPI-2 Rev. E provides the following lines / signals for the communication as SPI master. The numbers 1 and 2 refer to the numbers of the interfaces:

Signal	Description
SCLK1 out SCLK2 out	Outgoing clock signal (SPI clock) being created by the appropriate master. The maximum SPI clock rate is 12.5 MHz. The real maximum clock rate depends on the used cable length.
SS1 out SS2 out	Outgoing slave select signal to activate connected SPI slaves for the currently valid SPI message. The single slave select line <code>SS out</code> is used in automatic activation mode; with manual activation further slave select lines can be used.
DATAOUT1 DATAOUT2	Data line from master to slave: Master Out, Slave In (MOSI).
DATAIN1 DATAIN2	Data line from slave to master: Master In, Slave Out (MISO).

Interface SPI slave

The module Pro II-SPI-2 Rev. E provides the following lines / signals for the communication as SPI slave. The numbers 1 and 2 refer to the numbers of the interfaces:

Signal	Description
SCLK1 in SCLK2 in	Incoming clock signal (SPI Clock) being being created by the appropriate master. The maximum SPI clock rate is 12.5 MHz. The real maximum clock rate depends on the used cable length.
SS1 in SS2 in	Incoming slave select signal to activate the SPI slave for the currently valid SPI message.
DATAIN1 DATAIN2	Data line from master to slave: Master Out, Slave In (MOSI).
DATAOUT1 DATAOUT2	Data line from slave to master: Master In, Slave Out (MISO).

At the input and at the output of the SPI slave any data runs through a FIFO ring buffer. Both input and output Fifo can hold up to 18 values of 32 bits.

When the slave receives a message from the SPI master through SPI MOSI, the slave starts sending the next message from the output Fifo through SPI MOSI to the master. Sending is done within the same data transfer.

As soon as an incoming message is received completely, the SPI slave stores the message in the input Fifo. You can read messages from the input Fifo via software. The Fifo can buffer up to 18 messages of 32-bit length; if the SPI message length is defined greater than 32 bits, only 9 messages (using each 2 Fifo values) can be buffered.

The output Fifo of the SPI slave can also buffer 18 messages of 32-bit length or 9 messages of more than 32-bit length. Please make sure to write new messages into the output Fifo in time so the Fifo does not run empty. If and as long as the output Fifo contains no more messages, the SPI slave sends the recently sended message again.

Operation mode digital module

With all operation modes, the module provides 8 digital channels with TTL levels (see specification). The signal type of other digital channels differs between module variants Pro II-SPI-2-T (TTL levels) and Pro II-SPI-2-D (differential signals), see below.

In operation mode digital module (see **P2_SPI_Mode**), the module runs without SPI interfaces, i.e. all channels run as digital inputs or outputs.

The module provides an EVENT input. Via the trigger input EVENT an external signal can trigger a process cycle of the ADwin CPU, which is processed immediately and completely (see ADbasic manual).

Pro II-SPI-2-T (TTL level)

With SPI operation modes, some pins not being used for SPI signals are available as additional digital inputs or outputs. These pins are already fixed to be running as inputs or outputs.

In operation mode digital module, 32 digital inputs and outputs with TTL levels are available (similar to digital module Pro II-DIO 32-TiCo). The channels can be configured in groups of 8 as inputs or outputs using the instruction **P2_DigProg**. Please note the appropriate software documentation and the online help.

Pro II-SPI-2-D (differential signals)

In operation mode digital module, 8 digital channels with TTL levels and 12 differential channels are available. With SPI operation modes, the 8 digital TTL channels and 4 differential channels are available.

The channels with TTL levels can be configured in groups of 4 as inputs or outputs using the instruction **P2_DigProg**. The differential digital channels can be individually configured as inputs or outputs with **P2_DigProg_Bits**. The channels are configured as inputs after power up.

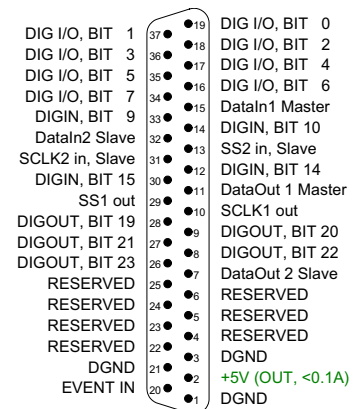
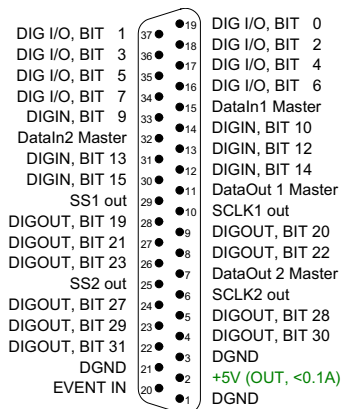
Please note, that instructions for digital channels of the module Pro II-SPI-2-D are valid for both channels with TTL signals and channels with differential signals. As an example **P2_Digout_Long** will set differential signals with bits 0...11 and TTL signals with bits 16...19 / 24...27 (as far as they were configured as outputs). The bits 12...15, 20...23, and 28...31 are not used here.

Hardware

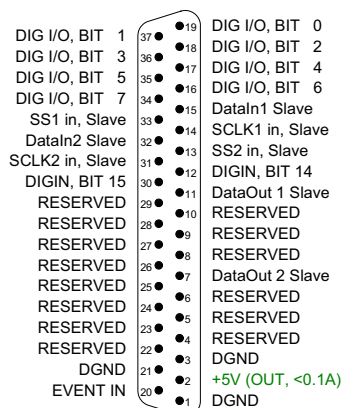
The pin assignment depends on the current operation mode of the module (**P2_SPI_Mode**).

If you have problems with shielding of unused DataIn/DataOut lines, it may help to connect the pins to logical 0 or DGND.

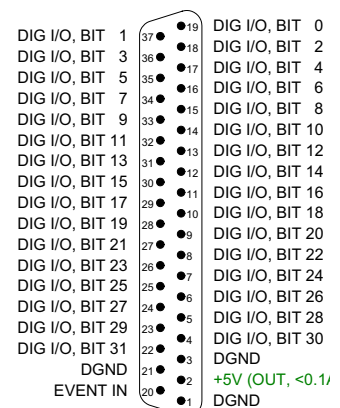
Pro II-SPI-2-T



SPI-2-T: Master 1 + Master 2



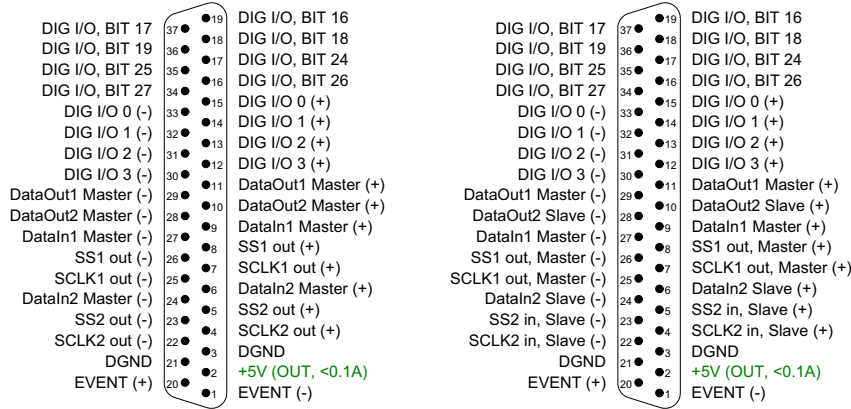
SPI-2-T: Master 1 + Slave 2



SPI-2-T: Slave 1 + Slave 2

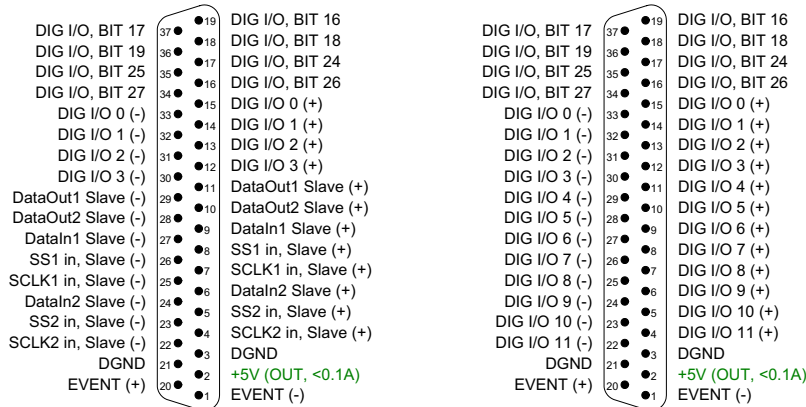
SPI-2-T: Digital

Pro II-SPI-2-D



SPI-2-D: Master 1 + Master 2

SPI-2-D: Master 1 + Slave 2



SPI-2-D: Slave 1 + Slave 2

SPI-2-D: Digital

Specification

Inputs/outputs	Pro II-SPI-2-T: SPI channels with TTL logic (5V) 8...32 dig. I/O, event input Pro II-SPI-2-D: SPI channels with diff. signals 8 dig. I/O TTL 4...12 diff. dig. I/O, diff. event input
Input resistance	TTL inputs: 10 kΩ, pull-down Diff. inputs: 120 Ω
V _{IH} min.	2 V
V _{IL} max.	0,8 V
I _{IH} max.	1 μA
Voltage range	0 V ... +5 V
Output current	max. ±35 mA per channel, max. ±70 mA per SPI interface
Input / output FIFO	Size: 511 value pairs Frequency: 100MHz
TiCo	Prozessor type: TiCo1 Clock rate: 50MHz Memory size: 28kiB PM internal, 28kiB DM internal
Connector	37-pin D-Sub female connector

Module revisions

The differences between the revisions are described below: The revision number to be found on the front cover.

Programming in ADbasic

Revision	Output date	Previous changes
E01	01 / 2013	Erst-Version

Different revisions numbers mean different module properties and are documented separately. Any additional numbering of the revision is used for internal purposes.

Programming

The module is comfortably programmed with *ADbasic* instructions. Instructions are described in *ADbasic* online help and in the Pro II Software manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Function	Instructions
Configure SPI interface	<code>P2_SPI_Mode</code> <code>P2_SPI_Config</code> <code>P2_SPI_Master_Config</code> <code>P2_SPI_Slave_Config</code>
SPI master: special configuration	<code>P2_SPI_Master_Set_Clk_Wait</code>
SPI master: provide output data	<code>P2_SPI_Master_Set_Value32</code> <code>P2_SPI_Master_Set_Value64</code>
SPI master: Set slave select line	<code>P2_Digout</code> , <code>P2_Digout_Bits</code> <code>P2_Digout_Long</code>
SPI master: Start data transfer	<code>P2_SPI_Master_Start</code>
SPI master: read status	<code>P2_SPI_Master_Status</code>
SPI master: Read message	<code>P2_SPI_Master_Get_Value32</code> <code>P2_SPI_Master_Get_Value64</code> <code>P2_SPI_Master_Get_Static_Input</code>
SPI slave: Use input/output Fifo	<code>P2_SPI_Slave_OutFifo_Write</code> <code>P2_SPI_Slave_OutFifo_Empty</code> <code>P2_SPI_Slave_InFifo_Full</code> <code>P2_SPI_Slave_InFifo_Read</code> <code>P2_SPI_Slave_Clear_Fifo</code>
Configure digital TTL channels	<code>P2_DigProg</code>
Configure digital differential channels	<code>P2_DigProg_Bits</code> (Pro II-SPI-2-D only)
Query input signals	<code>P2_Digin_Long</code>
Use latch register	<code>P2_Dig_Latch</code> , <code>P2_Dig_Read_Latch</code> <code>P2_Dig_Write_Latch</code> , <code>P2_Sync_All</code>
Control edges at input channels	<code>P2_Dig_Fifo_Mode</code> <code>P2_Digin_Fifo_Enable</code> <code>P2_Digin_Fifo_Read</code> <code>P2_Digin_Fifo_Read_Fast</code> <code>P2_Digin_Fifo_Read_Timer</code> <code>P2_Digin_Fifo_Clear</code> <code>P2_Digin_Fifo_Full</code>
Query input edge status	<code>P2_Digin_Edge</code>

Function	Instructions
Set and read back output signals	<code>P2_Digout</code> , <code>P2_Digout_Bits</code> <code>P2_Digout_Long</code> <code>P2_Digout_Reset</code> <code>P2_Digout_Set</code> <code>P2_Get_Digout_Long</code>
Set output signals automatically	<code>P2_Digout_Fifo_Clear</code> <code>P2_Digout_Fifo_Empty</code> <code>P2_Digout_Fifo_Enable</code> <code>P2_Digout_Fifo_Read_Timer</code> <code>P2_Digout_Fifo_Start</code> <code>P2_Digout_Fifo_Write</code>
Set module LED	<code>P2_Check_LED</code> , <code>P2_Set_LED</code>
Set event input	<code>P2_Event_Enable</code> , <code>P2_Event_Config</code> <code>P2_Event_Read</code>

The module can be programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `SPI_TiCo.inc` contains instructions for the functions:

Function	Instructions
Configure SPI interface	<code>SPI_Mode</code> <code>SPI_Config</code> <code>SPI_Master_Config</code> <code>SPI_Slave_Config</code>
SPI master: special configuration	<code>SPI_Master_Set_Clk_Wait</code>
SPI master: provide output data	<code>SPI_Master_Set_Value32</code> <code>SPI_Master_Set_Value64</code>
SPI master: Set slave select line	<code>Digout</code> , <code>Digout_Bits</code> <code>Digout_Long</code>
SPI master: Start data transfer	<code>SPI_Master_Start</code>
SPI master: read status	<code>SPI_Master_Status</code>
SPI master: Read message	<code>SPI_Master_Get_Value32</code> <code>SPI_Master_Get_Value64</code> <code>SPI_Master_Get_Static_Input</code>
SPI slave: Use input/output Fifo	<code>SPI_Slave_OutFifo_Write</code> <code>SPI_Slave_OutFifo_Empty</code> <code>SPI_Slave_InFifo_Full</code> <code>SPI_Slave_InFifo_Read</code> <code>SPI_Slave_Clear_Fifo</code>
Configure digital TTL channels	<code>DigProg</code>
Configure digital differential channels	<code>DigProg_Bits</code> (Pro II-SPI-2-D only)
Query input signals	<code>Digin_Long</code>
Use latch register	<code>Dig_Latch</code> , <code>Dig_Read_Latch</code> <code>Dig_Write_Latch</code>

Programming in TiCoBasic

Programming TiCo access

Function	Instructions
Control edges at input channels	<code>Dig_Fifo_Mode</code> <code>Digin_Fifo_Enable</code> <code>Digin_Fifo_Read</code> <code>Digin_Fifo_Read_Timer</code> <code>Digin_Fifo_Clear</code> <code>Digin_Fifo_Full</code>
Query input edge status	<code>Digin_Edge</code>
Set and read back output signals	<code>Digout, Digout_Bits</code> <code>Digout_Long</code> <code>Digout_Reset</code> <code>Digout_Set</code> <code>Get_Digout_Long</code>
Set output signals automatically	<code>Digout_Fifo_Clear</code> <code>Digout_Fifo_Empty</code> <code>Digout_Fifo_Enable</code> <code>Digout_Fifo_Read_Timer</code> <code>Digout_Fifo_Start</code> <code>Digout_Fifo_Write</code>
Set module LED	<code>Check_LED, Set_LED</code>
Set event input	<code>Event_Enable, Event_Config</code> <code>Trigger_Event</code>

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	<code>P2_TDrv_Init</code> <code>P2_GetData_Long, P2_Get_Par,</code> <code>P2_Get_Par_Block</code> <code>P2_SetData_Long, P2_Set_Par,</code> <code>P2_Set_Par_Block</code> <code>P2_Get_TiCo_RingBuffer,</code> <code>P2_Set_TiCo_RingBuffer</code> <code>P2_RingBuffer_Empty</code> <code>P2_RingBuffer_Full</code>
Control <i>TiCo</i> processor	<code>P2_TiCo_Reset, P2_TiCo_Start,</code> <code>P2_TiCo_Stop</code> <code>P2_Get_TiCo_Bootloader_</code> <code>Status</code> <code>P2_Get_TiCo_Status, P2_Workload</code>
Control <i>TiCo</i> processes	<code>P2_Process_Status</code> <code>P2_TiCo_Get_Processdelay</code> <code>P2_TiCo_Set_Processdelay</code> <code>P2_TiCo_Start_Process</code> <code>P2_TiCo_Stop_Process</code>
Transfer <i>TiCo</i> programs	<code>P2_TiCo_Flash, P2_TiCo_Load</code>

5.8.19 Pro II-LS-2 Rev. E

The module [Pro II-LS-2 Rev. E](#) provides 2 LS-bus interfaces on 9-pin D-SUB connectors (female).

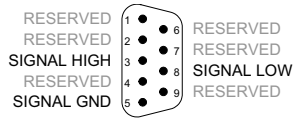


Fig. 177 – Pro II-LS-2 Rev. E: Pin assignment

The LS bus is a bi-directional serial bus with 5MHz clock rate (Low Speed). The bus is an in-house design to access external modules. The first module available is HSM-24V, which can process 24 Volt signals on 32 digital channels.

The bus is set up as line connection, i.e. the ADwin interface and up to 15 LS bus modules are connected to each other via two-way links. The last module of the LS bus must have the bus termination activated. The maximum bus length is 5m.

The LS bus modules are programmed with *ADbasic* instructions, which are sent via the LS bus interface of the ADwin system. The instructions for the LS bus module are described in the manual of the LS bus module and in the online help.

The variants Pro II-Aln-8/18-xxx-TiCo additionally provide a freely programmable *TiCo* processor with 28KiByte data memory and 28KiByte program memory, which has access to all inputs of the module. Find more information about use and programming of the *TiCo* processor in the manual *TiCoBasic*.

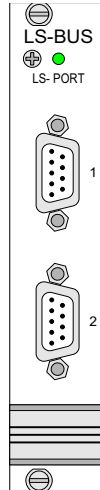
If you store a *TiCoBasic* program in the *TiCo* bootloader, the program is automatically loaded into the *TiCo* processor and started on power-up. Thus, the module can run on its own and independently from the CPU module of the ADwin-Pro II system.

On the module, you can only program the LED using the instructions `P2_Check_LED`, `P2_Set_LED`.

LS bus modules are programmed with *ADbasic* instructions. The instructions are described in *ADbasic* online help and in the HSM-24V manual.

The include file `ADwinPro_All.inc` contains instructions for the following functions:

Range	Instructions
Initialize a module on the LS-bus.	<code>P2_LS_DIO_Init</code>
Configure digital channels of module HSM-24V as inputs or outputs.	<code>P2_LS_DigProg</code>
Query state of digital channels on module HSM-24V (only with 1 modul on LS-bus).	<code>P2_LS_Dig_IO</code> , <code>P2_LS_Digin_Long</code>
Output value	<code>P2_LS_Digout_Long</code> , <code>P2_LS_Get_Output_Status</code>
Initialize the watchdog of the LS bus.	<code>P2_LS_Watchdog_Init</code> , <code>P2_LS_Watchdog_Reset</code>



TiCo processor

Programming in ADbasic

**Programming in
 TiCoBasic**

On the module, you can only program the LED using the instructions **Check_LED**, **Set_LED**.

LS bus modules are programmed with *TiCoBasic* instructions. The instructions are described in *TiCoBasic* online help.

The include file `AInTiCo.inc` contains instructions for the following functions:

Range	Instructions
Initialize a module on the LS-bus.	LS_DIO_Init
Configure digital channels of module HSM-24V as inputs or outputs.	LS_DigProg
Query state of digital channels on module HSM-24V (only with 1 modul on LS-bus).	LS_Dig_IO, LS_Digin_Long
Output value	LS_Digout_Long, LS_Get_Output_Status
Initialize the watchdog of the LS bus.	LS_Watchdog_Init, LS_Watchdog_Reset

**Programming TiCo
 access**

To access the *TiCo* processor from the ADwin CPU the following *ADbasic* instructions are defined in the include file `ADwinPro_All.inc`. The instructions are described in *ADbasic* online help and in the *TiCoBasic* manual.

Function	Instructions
Data exchange with the <i>TiCo</i> processor via global variables	P2_TDrv_Init P2_GetData_Long, P2_Get_Par, P2_Get_Par_Block P2_SetData_Long, P2_Set_Par, P2_Set_Par_Block P2_Get_TiCo_RingBuffer, P2_Set_TiCo_RingBuffer P2_RingBuffer_Empty P2_RingBuffer_Full
Control <i>TiCo</i> processor	P2_TiCo_Reset, P2_TiCo_Start, P2_TiCo_Stop P2_Get_TiCo_Bootloader_Status P2_Get_TiCo_Status, P2_Workload
Control <i>TiCo</i> processes	P2_Process_Status P2_TiCo_Get_Processdelay P2_TiCo_Set_Processdelay P2_TiCo_Start_Process P2_TiCo_Stop_Process
Transfer <i>TiCo</i> programs	P2_TiCo_Flash, P2_TiCo_Load

6 Calibration

6.1 General information

The digital-to-analog (DAC) and analog-to-digital (ADC) converters of the **ADwin** systems are calibrated in factory. According to the regulations for measurement accuracy, we recommend calibrating the systems in regular time intervals.

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 activities, while recognizing and avoiding any possible dangers.

(Definition of qualified personnel as per VDE 105 and ICE 364).

This product documentation and all documents referred to, have always to be available and to be strictly 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.

The following tools are necessary to calibrate the system:

- a reference voltage source with an accuracy of:
 - 10 μ V for calibration of 18-bit converters
 - 30 μ V for calibration of 16-bit converters
 - 100 μ V for calibration of 14-bit converters
- a digital multi meter with an accuracy of:
 - 10 μ V for calibration of 18-bit converters
 - 30 μ V for calibration of 16-bit converters
 - 100 μ V for calibration of 14-bit converters
- connecting cables from the input/outputs to the reference voltage and to the measurement device

Caution: Risk of electric shock.

ADwin-Pro systems have a power supply device, which gives access to high-voltage lines and connectors if the system is open. The ventilation slots are wide enough to pass through an alignment tool of 2.5 mm (=0.1inch).

Calibrate the system only when it is closed!

Do not pass any conductive objects through the ventilation slots!



Qualified personnel

Availability of the documents



Tools

Voltage range

Allocating digits to voltage

Zero offset

Least Significant bit V_{LSB}

Gain k_V

Allocating the bits

DAC

6.2 Calculation basis

The standard voltage range of the analog inputs/outputs of the **ADwin** systems is -10V ... +10V (bipolar 20 Volt).

The 65536 (2^{16}) digits are allocated to the corresponding voltage ranges of the ADC and DAC in such a manner that the value for

- 0 (zero) digits corresponds to the maximum negative voltage.
- 65535 digits correspond to the maximum positive voltage.

The value for 65536 digits, exactly 10 Volt, is therefore just beyond the measurement range, therefore you get for the 16-bit AD or DA conversion a maximum voltage value of 9.999695 Volt, and for the 18-bit AD conversion a value of 9.999923706 Volt.

In bipolar settings, this results in a zero offset, called offset in the following text. The offset has the value $V_{OFF}=-10V$.

The value V_{LSB} defines the voltage, which corresponds to the least significant bit. The value in the standard setting is

- with 18-bit converters: $20V \cdot 2^{-18} = 76.294 \mu V$
- with 16-bit converters: $20V \cdot 2^{-16} = 305.175 \mu V$
- with 14-bit converters: $20V \cdot 2^{-14} = 1220.7 \mu V$

When using Pro-AIn modules with programmable gain arrays (PGA), you can amplify the input voltage by factors 2, 4 and 8. Thus, the measurement range gets smaller by the corresponding gain factor k_V .

Please pay attention to the fact that also the interference signals are amplified when using applications with $k_V > 1$. These can be reduced by programming digital filters in **ADbasic**.

In order to get the same allocation of bits during measurements with 14-bit ADC as with a 16-bit ADC, the converted value is presented left-aligned in the lower word (16 bit) with the 14-bit ADC. The least 2 significant bits are always 0.

Bit no.	31...2 4	23...1 6	15...6	5...2	1...0
con- tent	0	18-bit value in bits 6...23		0	0
	0	0	16-bit value in bits 0...15		
	0	0	14-bit value in bits 2...15		0
	upper word		lower word		

Fig. 178 – Bit allocation with different resolutions

The 16384 digits of a 14-bit ADC are mapped to the 65535 digits of a 16 bit ADC. Therefore, 4 digits of the 16-bit ADC correspond to one digit of the 14 bit ADC.

For DAC use the formula:

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

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

For ADC use the formula:

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

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

Tolerance range

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 your hardware manual.

- The integral non-linearity (INL) defines the deviation from the ideal wave form covering the whole input voltage range.
- The differential non-linearity (DNL) defines the deviation from the ideal value of the quantization level.

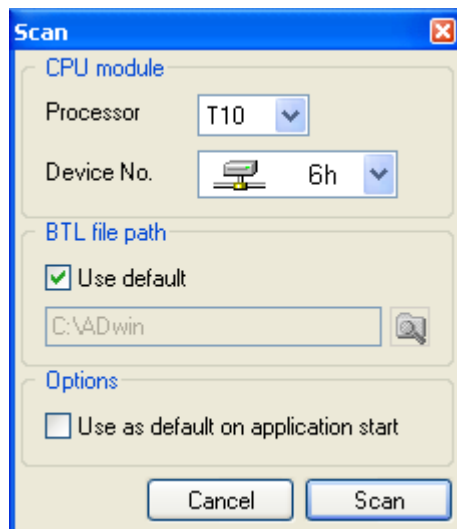
6.3 Calibrating a module

At first define the voltage range of the module.

Calibration has to be made when the system reaches its operating temperature. 30 minutes after power-up of the system, the operating temperature is reached, provided the system has a (room) temperature of approx. 20...25°C before power-up.

Please note the general information in [chapter 6.1](#).

Call the program `ADpro.exe` from the Windows start menu under "Programs\ADwin". The program requires Microsoft .NET Framework 2.0. The dialog window `Scan` opens.



Please note: The next step will stop all processes and reset all module settings!

Enter the data for the **ADwin** system to be calibrated. The button `Scan` starts a connection to the **ADwin** system and reads system information. The program `ADpro.exe` will initialize the **ADwin** system, i.e. it stops and deletes running processes.

If your **ADwin** system has booted successfully, the window "ADwin - ADpro" opens.

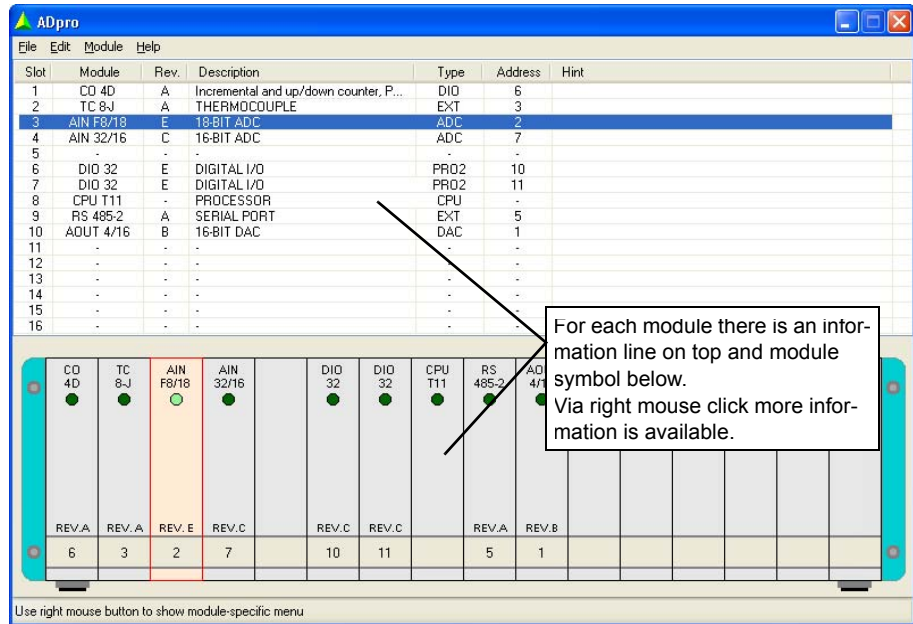
ADC

INL

DNL

Initializing the hardware





Calibration

In the window ADpro, click on the module to calibrate and select the menu entry Calibration in the Module menu. If the Calibration entry is not displayed in the menu, the selected module cannot be calibrated.

Connect the measurement device and the reference voltage source.

Follow the information displayed in the dialog window. Please note the difference of analog input modules with and without multiplexer:

- with the AIN modules the calibration of the ADC is effected via the input channel 1.
- with the AIN-F modules the connected channel is selected in the window "Input channel".

7 Accessories

7.1 LEMO Cable Sets for ADwin-Pro Systems

1-pole

Pro-CS-1	4 cables with 200mm (7.8 inch) and 4 cables with 400mm (15.7 inch)
Pro-CS-2	4 cables with 400mm (15.7 inch) and 4 cables with 800mm (31.5 inch)
Pro-CS-3	4 cables with 1000mm (39.4 inch) and 4 cables with 1500mm (59 inch)
Pro-CS-4	4 cables with 5000mm (196.8 inch)
Pro-CS-5	8 cables 400 mm (15.7 inch)
Pro-CS-6	8 cables 1000 mm (39.4 inch)
Pro-CS-7	8 cables 2000 mm (78.7 inch)

All cables with a male LEMO connector on each end.

2-pole

Pro-CS-8	4 cables with 2000mm (78.7 inch): LEMO connector 2-pole - cable - non-assembled end
Pro-CS-9	4 cables 1000 mm (39.4 inch) and 4 LEMO connectors (female, loose) for front panel assembly
Pro-CS-10	4 cables 500 mm (19.7 inch) and 4 LEMO connectors (female, loose) for front panel assembly
Pro-CS-11	4 cables 2000 mm (78.7 inch) and 4 LEMO connectors (female, loose) for front panel assembly

If not stated otherwise, all cables have a male 2-pole LEMO connector on each end.

7.2 LEMO Adapter sets

Pro-AS-1	4 adapters: LEMO female connectors to BNC connectors (male)
Pro-AS-3	4 LEMO Y connectors (male to double female)
Pro-AS-4	4 adapters: LEMO female connector to LEMO female connector
Pro-AS-5	4 terminators: 50 Ω , LEMO female connector
Pro-AS-6	4 cable adapters, length 4" / 150 mm: LEMO female connector to BNC male connector
Pro-AS-7	4 cable adapters, length 4" / 1000 mm: LEMO female connector to BNC male connector
Pro-AS-8	4 cable adapters, length 4" / 2000 mm: LEMO female connector to BNC male connector
Pro-AS-9	4 cable adapters, length 4" / 1000 mm: LEMO female connector to BNC female connector
Pro-AS-10	4 cable adapters, length 4" / 2000 mm: LEMO female connector to BNC female connector

7.3 Cables / Terminal blocks for OPT-16 and TRA-16

ADwin-Cable-1	Extension cable 1000 mm, shielded, for 37-pin DSub connectors; one side female, other side male
ADwin-Cable-2	Extension cable 500 mm, shielded, for 37-pin DSub connectors; one side female, other side male
ADwin-Cable-3	Extension cable 250 mm, shielded, for 37-pin DSub connectors; one side female, other side male
ADwin-AT-37M	Terminal block for 37-pin DSub connector

7.4 Enclosure Mounting

Pro-Mount	Mounting plate (to be bolted with the enclosure handles) to attach a Pro I enclosure e.g. in a 19" rack.
Pro II-Mount	Mounting angle (to be screwed to the enclosure side panels) to attach a Pro II enclosure e.g. in a 19" rack.

7.5 Reference addresses

7.5.1 LEMO Connectors

Pro modules are equipped with the following LEMO connectors:

- Male connectors / female connectors of series 00 NIM-CAMAC, 1-pole
 - Cable connector: Type FFS
 - Built-in female connector: Type ERN
- Male connectors / female connectors of series 00 Multi-Contact, 2-pole
 - Cable connector: Type FGG
 - Built-in female connector: Type EGG
- Pt100/RTD-8 modules: Male connectors / female connectors of series 0B:
 - Cable connector: Type FGG
 - Built-in female connector: Type EGG

Manufacturer of LEMO connectors:

LEMO S.A.	Tel.: +41 21 695 16 00
Chemin de Champs-Courbes 28	Fax: +41 21 695 16 01
P.O. Box 194	E-Mail: info@lemo.com
CH-1024 Ecublens, Switzerland	Internet: www.lemo.com

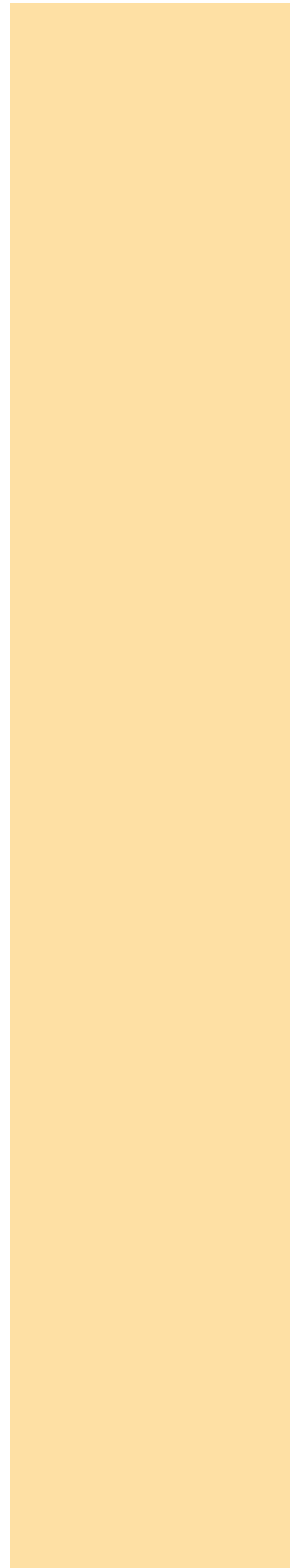
7.5.2 Power Supply Pro-Mini

The plug connector for external power supply of the casing Pro-Mini is manufactured by Phoenix Contact GmbH:

Combicon plug component, pitch 5.0mm, Type MSTB 2,5/ 3-STF;
order no. 1786844 (as of Dec. 2005)

Manufacturer of the connector:

Phoenix Contact GmbH & Co. KG	Tel.: +49 5235 300
Flachsmarktstraße 8	Fax: +49 5235 341 200
D-32825 Blomberg	E-Mail: info@phoenixcontact.com
	Internet: www.phoenixcontact.com



Annex

A.1 RoHS Declaration of Conformity

The RoHS directive 2011/65/EU of the European Union on the restriction of the use of certain hazardous substances in electrical and electronic equipment (RoHS directive) has become operative as from 3rd January, 2013.

The following substances are involved:

- Lead (Pb)
- Cadmium (Cd)
- Hexavalent chromium (Cr VI)
- Polybrominated biphenyls (PBB)
- Polybrominated diphenyl ethers (PBDE)
- Mercury (Hg)
- Bis(2-ethylhexyl) phthalate (DEHP)
- Benzyl butyl phthalate (BBP)
- Dibutyl phthalate (DBP)
- Diisobutyl phthalate (DIBP)

The product line **ADwin-Pro II** complies with the requirements of the RoHS directive in all delivered variants.

A.2 List of Modules

The list of modules is sorted by page numbers.

Pro-CPU-T11-ENET;	14
Pro-CPU-T12;	17
Pro II-Boot with Storage Device;	19
Pro II-Boot;	22
Pro II-MIO-4 Rev. E;	24
Pro II-MIO-4-ET1 Rev. E;	31
Pro II-MIO-D12 Rev. E;	44
Pro II-Aln-8/18 Rev. E;	53
Pro II-Aln-32/18-D Rev. E;	57
Pro II-Aln-16/18-C Rev. E;	61
Pro II-Aln-8/18-8B Rev. E;	64
Pro II-Aln-16/18-8B Rev. E;	67
Pro II-Aln-F-4/14 Rev. E;	70
Pro II-Aln-F-8/14 Rev. E;	74
Pro II-Aln-F-4/16 Rev. E;	78
Pro II-Aln-F-8/16 Rev. E;	82
Pro II-Aln-F-4/18 Rev. E;	86
Pro II-Aln-F-8/18 Rev. E;	89
Pro II-AOut-4/16 Rev. E;	93
Pro II-AOut-8/16 Rev. E;	96
Pro II-AOut-1/16 Rev. E;	99
Pro II-DIO-32 Rev. E;	105
Pro II-DIO-32-TiCo Rev. E;	107
Pro II-DIO-32/1-TiCo Rev. E;	111
Pro II-DIO-32-TiCo2 Rev. E;	115

Pro II-DIO-8-D12 Rev. E;	119
Pro II-OPT-16 Rev. E;	123
Pro II-OPT-32-24V Rev. E;	126
Pro II-REL-16 Rev. E;	128
Pro II-TRA-16 Rev. E;	130
Pro II-TRA-16-G Rev. E;	130
Pro II-PWM-16 Rev. E;	132
Pro II-PWM-16-I Rev. E;	132
Pro II-COMP-16 Rev. E;	135
Pro II-CNT-T Rev. E;	139
Pro II-CNT-D Rev. E;	139
Pro II-CNT-I Rev. E;	139
Pro II-RTD-8 Rev. E;	149
Pro II-TC-8 ISO Rev. E;	153
Pro II-SG-4/18 Rev. E;	155
Pro II-CAN-2 Rev. E;	158
Pro II-CAN-FD-2 Rev. E;	164
Pro II-RSxxx Rev. E;	171
Pro II-RS422-4 Rev. E;	175
Pro II-LIN-2 Rev. E;	178
Pro II-Profi-SL Rev. E;	180
Pro II-Profinet Rev. E;	183
Pro II-MIL-1553 Rev. E;	187
Pro II-ARINC-429 Rev. E;	190
Pro II-FlexRay-2 Rev. E;	194
Pro II-EtherCAT-SL Rev. E;	196
Pro II-SENT-4 Rev. E;	199
Pro II-SENT-6 Rev. E;	204
Pro II-SENT-4-Out Rev. E;	209
Pro II-SPI-2 Rev. E;	213
Pro II-LS-2 Rev. E;	221

