ADwin

Driver for LabVIEW version 6 and higher

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

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

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

Table of contents

Typographical Conventions

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

You find a "note" next to

– information, which 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 instructions and user inputs are characterized by the font `Courier New`.

`Var_1` ADbasic source code elements such as instructions, variables, comments and other text are characterized by the font `Courier New` and are printed in color (see also the editor of the *ADbasic* development environment).

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

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

1 Information about this Manual

This manual contains comprehensive information about the ***ADwin***-LabVIEW® driver for LabVIEW® version 6 and higher.

The driver is not applicable for LabVIEW versions 5 or before. If necessary, please contact our hotline.

Additional information are available in

– the manual "***ADwin*** Driver Installation", which describes all interface installations for the ***ADwin*** systems.

  Begin your installation with his manual.

– if you work with Linux: the manual "ADwin Linux" which describes the software installation and the ***ADbasic*** compiler usage from Linux.

– the manual "***ADbasic***", which contains all instructions for the compiler ***ADbasic***. With this comfortable real-time development tool you are programming your ***ADwin*** system.

  The online help contains the same information as the manual.

– the hardware manuals for the ***ADwin*** systems you are using.

It is assumed that you are familiar with your LabVIEW® environment.

Please note:

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

Qualified personnel

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

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

Availability of the documents

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

Legal information

This documentation, including all pictures is protected by copyright. Reproduction, translation as well as electronical and photographical 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.

Subject to change.

Hotline address: see inner side of cover page.

2 *ADwin*-LabVIEW driver

More features for LabVIEW with ADwin

The ***ADwin*** system consists of an independent on-board CPU, which executes measurement and control tasks very fast and reliably, as well as of an interface under Windows or Linux in order to control the ***ADwin*** system with LabVIEW.

Consequently you transfer all time-critical processes to the ***ADwin*** system, but with LabVIEW you still have control of the processes and data processing.

How to program the *ADwin* system

ADwin systems are fast, reliable and flexible. You apply the easy-to-learn programming language ADbasic in order to use all these advantages.

Before you can apply the here described LabVIEW instructions, we recommend to familiarize yourself with ADbasic. Please use the ADbasic manual and the programming instructions as help. The descriptions will help you to understand the ***ADwin*** system more easily.

Controlling *ADwin* systems with LabVIEW

Now it's time to start working with this manual.

The sections 2.1 and 2.2 explain how LabVIEW and ***ADwin*** communicate with each other and deepens your knowledge for the ***ADwin*** concept.

In chapter 3 the installation and the integration of the new commands are described.

The general use of the LabVIEW drivers is explained in chapter 4, its functions in chapter 5 which can also be used as a kind of reference documentation.

2.1 Interface for the development environment

The ***ADwin***-LabVIEW driver is an interface for the development environment LabVIEW from version 6 used for the communication with ***ADwin*** systems.

The combination of LabVIEW® and an ***ADwin*** system offers you totally new possibilities. The most fast reaction and computing power of an ***ADwin*** system on the one hand and the LabVIEW functions for managing, analysis and documentation of measuring values on the other hand join into a powerful concept.

Typical applications are:

– Control of test stands
– Generating signals
– Measure with intelligence, collect data under complex trigger conditions
– Open-loop and closed-loop control
– Online processing, data reduction
– Hardware-in-the-Loop, simulation of sensor signals

2.2 Communication with the *ADwin* system

With the development environment you can control processes in the ***ADwin*** system, as well as getting data from there or sending data. You are programming processes with the real-time development tool ***ADbasic***, create a binary file and transfer it to the ***ADwin*** system (see ***ADbasic*** manual or online help).

Data and instructions between LabVIEW® and the ***ADwin*** system are processed according to the following illustration.

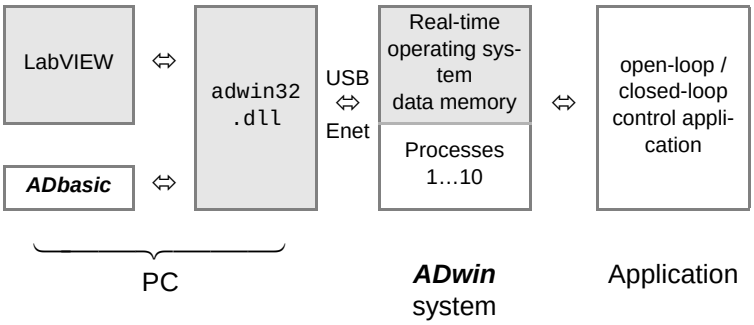


Fig. 1 – ***ADwin***-LabVIEW interface

adwin32.dll

The `adwin32.dll` is the only interface for Windows applications to the ***ADwin*** system and is therefore used by ***ADwin*** LabVIEW driver, too. With this interface several Windows programs can communicate with the ***ADwin*** system at the same time: Thus, the development environments, ***ADbasic***, and ***ADtools*** can work with the ***ADwin*** system simultaneously.

Real-time operating system

The interface `adwin32.dll` communicates with the real-time operating system of the ***ADwin*** device. Therefore you must load the operating system (e.g. the file `<adwin9.btl>`)after each power-up. After a successful loading the system will be able to receive and execute processes, receive instructions from the PC and exchange data with it. The processes programmed in ***ADbasic***, include the program code for measurement, open-loop or closed-loop control of your application.

The real-time operating system performs the following tasks:

10 processes

– Management of up to 10 real-time processes with low or high priority (selectable). Processes with low priority can be interrupted by processes with high priority, the latter cannot be interrupted by other processes.

Data memory

– Providing global variables:
  - 80 integer variables (`PAR_1` … `PAR_80`), predefined
  - 80 float variables (`FPAR_1` … `FPAR_80`), predefined
  - 200 data arrays (`DATA_1` … `DATA_200`) whose length can be set individually

  You can read and change the values of these variables or data arrays from the development environment at any time.

Communication

– Communication between ***ADwin*** system and PC (`adwin32.dll`).

  The communication process is running with medium priority on the ***ADwin*** system and can interrupt low-priority processes for a short time. It interprets and processes all instructions, which are sent from LabVIEW to the ***ADwin*** system: Control instructions and instructions for data exchange.

  The communication process never sends data to the PC without being asked to do so. This assures that data are transferred to the PC only if you have requested these data.

3 Installing the LabVIEW Driver

Please follow the installation steps described below in order to get easy access to your ***ADwin*** system from LabVIEW®.

3.1 Do the "*ADwin* driver installation"

For the installation you need an up-to-date ***ADwin*** CDROM.

3.1.1 Installation under Linux or Mac

Please follow the installation guide in the manual "ADwin Linux / Mac". Please pay attention to installing the archive `adwin-labview-x.y.tar.gz` at last.

After successful installation you will find the files in folders below `</opt/adwin/share>` (standard installation):

VI collection and LabVIEW examples	`./ADwin_v2.2.lib`
Examples for *ADbasic*	`./samples_ADwin`

Continue with chapter 3.2 "Including the ADwin-VIs".

3.1.2 Installation under Windows

If *ADwin* is installed yet

If you have already installed an ***ADwin*** system and software skip this section and continue with chapter 3.2.

Else: New installation

Else, if an ***ADwin*** system is to be newly installed, please start the installation with the manual "***ADwin*** driver installation" which is delivered with the ***ADwin*** hardware. It describes how to

- install the software from the ***ADwin*** CDROM.
- Install the communication driver under Windows.
- Install the hardware in the PC (if necessary) and set up the hardware connections between PC and ***ADwin*** system.

After successful installation you will find the files in folders below `<C:\ADwin>` (standard installation):

VI collection and LabVIEW examples	`.\Developer\LabVIEW\ADwin_v2.2.lib`
Examples for *ADbasic*	`.\ADbasic\samples_ADwin`
Test program for *ADwin-Gold*, *ADwin-light-16* and plug-in boards	`.\Tools\Test\ADtest.exe`
Test program for *ADwin-Pro*	`.\Tools\Test\ADPro.exe`

Continue with the next chapter "Including the ADwin-VIs".

3.2 Including the *ADwin*-VIs

In LabVIEW programs ready-made VIs will send instrudctions and data to an ***ADwin*** system and read data from the system. The VIs will use functions which are implemented in the interface (see chapter 2.2).

Include the VIs into LabVIEW with the following steps:

– Copy the folder <ADwin_v2.2.lib> from the installation folder into the LabVIEW folder, under Windows e.g. <C:\Programs\National Instruments\LabVIEW 6>:

– Start LabVIEW and choose the button "Blank VI" (before LabVIEW 8: "New VI").

– Activate the white diagram window, either click on the hidden white window or use the keys [Strg]+[E].

Please not, that the next steps are different for LabVIEW version 8 on the one hand and for versions 6+7 on the other hand.

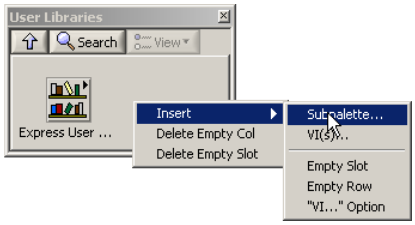
LabVIEW 8

– In the menu of the block diagram window select the option "Tools ▸ Advanced ▸ Edit Palette Set…".

The windows "Edit Controls and Functions Palettes" and "Functions" open.

– In the window "Functions", do a click on the symbol "User Libraries" and then do a right click (Mac: ⌘ + mouse click) in the window "User Libraries".

– Do a right click (Mac: ⌘ + mouse click) on the grey area in the window "User Libraries" and select the menu item "Insert ▸ Subpalette".



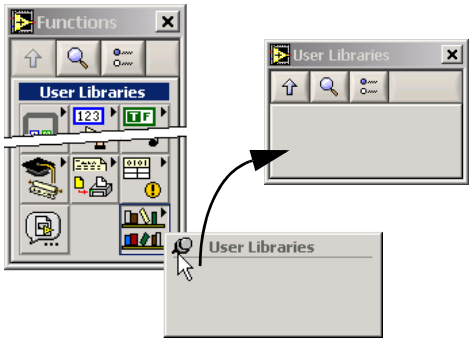
The window "Insert Subpalette" opens.

Continue with "All LabVIEW versions".

LabVIEW 6+7

– Do a right click (Mac: ⌘ + mouse click) on the symbol "User Libraries" in the window "Functions".

Anchor the opened window "User Libraries" on the desktop clicking on the drawing pin (see right).

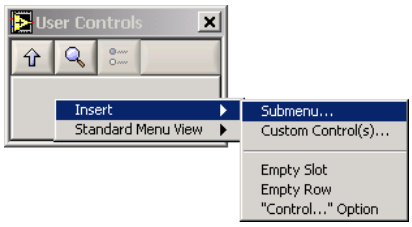


– Open the window "Edit Controls and Functions Palettes" like this:

- LabVIEW 6: In the window "User Controls" click on the symbol "Options" with the left mouse button.In the following window press the button "Edit Palettes …".
- LabVIEW 7: In the menu of the block diagram window select the option "Tools ▸ Advanced ▸ Edit Palette Views".

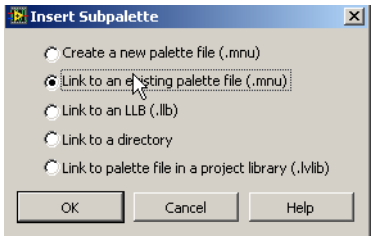
All LabVIEW versions

– Do a right click (Mac: ⌘ + mouse click) on the grey field of the window "User Libraries" and select the menu item "Insert ▸ Submenu".



The window "Insert Submenu" opens.

– Select the option "Link to an existing menu file (.mnu)" and confirm by pressing the OK button.

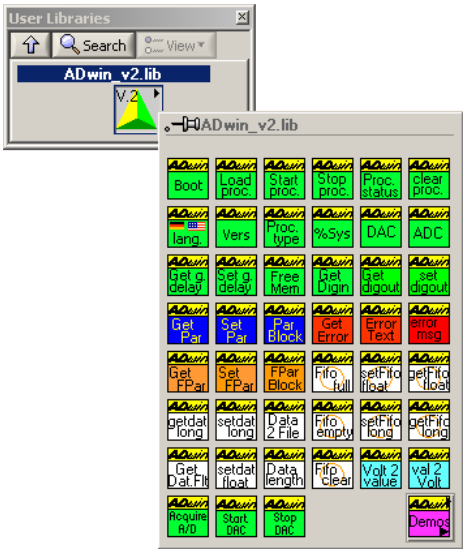


In the file selection box-in the LabVIEW directory-select the folder <ADwin_v2.2.lib> and then the file <ADwin_v2.2.mnu>.

The ***ADwin*** VIs are now included into the palette. Press the button "Save Changes" in the window "Edit Control and Functions Palettes".

– Activate the white diagram window once more.

If you move the mouse over the icon "ADwin_v2.1.lib" in the group "User Libraries", all VIs appear in a new windows (see right).



You can directly drag the VIs from the group into your LabVIEW diagram. The VIs are described in chapter 5.

3.3 Converting *ADwin* VIs

The ***ADwin*** VIs are stored from LabVIEW 6. If you use a later LabVIEW version, National Instruments® recommends to convert the VIs in order to have LabVIEW use less memory resources and avoid large run-time degradation of performance for the VIs.

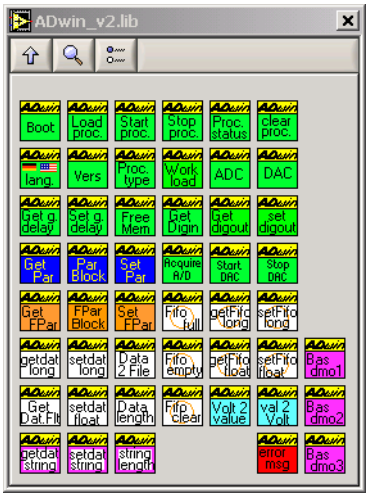
Find more advice in the LabVIEW manual from National Instruments® under „Converting VIs“; to convert a directory of VIs look under the keyword "mass compile".

4 General about *ADwin* VIs

4.1 Basic features

The colors of the VI icons show their function. The colors are partially related to the LabVIEW standard:

Green:	System information and process control
Blue:	Transfer of global long variables
Orange:	Transfer of global float variables
Red:	Error function
White:	Transfer of DATA arrays
White + ring:	Transfer of FIFO arrays (ring buffer)
White + frame:	Transfer of string data
`Light blue:`	Tools
Pink:	Demos



Please note the following information for the use of ***ADwin*** VIs:

– Connect all ***ADwin***-VIs with "`error in`" and "`error out`" using the error wiring (see chapter 4.3).

– Give a "Device No." (see chapter 4.4) as input value to the VIs. Only few VIs do not need a "Device No.".

– For every VI there is a context help window which is started with `[Strg]+[H]`.

  When you move the mouse to a VI symbol with the help window opened a help text and a graphic of the VI are displayed. The help text can also be found in this manual, but not the graphic.

4.2 Driver for LabVIEW 4, 5

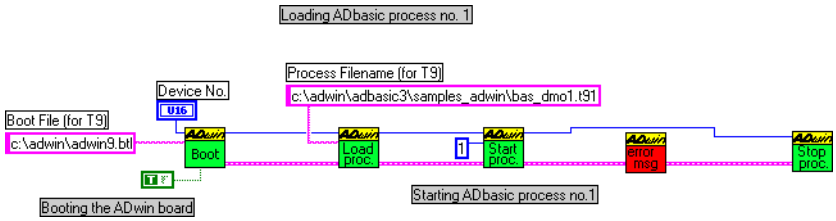
If you use an older driver for LabVIEW 4 or 5, you may use its VIs further on, but we do not recommend it.

The current driver uses revised VI names, is adapted for better use with LabVIEW 6 / 7 and enables easy error handling.

4.3 Error handling

This driver version provides "error wiring" as the possibility to handle errors. Furthermore, the error wiring defines the order LabVIEW will process the VIs.

The diagram below contains a dotted line (pink) connecting the bottom of the VIs: This is the error wiring. The error input of the first VI is left unconnected.



If an error occurs in an ***ADwin***-VI, the VI sends an error signal via the error wiring to the following VIs, which then stop accessing the ***ADwin*** system. Using the VI `error_msg`, an error message can be generated.

The error wiring is to be used consequently:

– Connect all VIs in a line via the connectors "`error in` " and "`error out`".

– Query occuring error signals with the VI `error_msg` to establish a simple error handling, e.g. once in a loop.

4.4 The "DeviceNo."

All ***ADwin*** systems are accessed via a unique "DeviceNo". The "DeviceNo" to the ***ADwin*** system is allocated by the program ADconfig.

In LabVIEW nearly all of the VIs have to know the "Device No.". Connect the Device No. to the first VI. Connect the output "Device No. out" to the next Vi and this way create a complete wiring (see example in "Error handling", upper blue line).

4.5 Example programs

The example programs show the cooperation between LabVIEW and the ***ADwin*** system. Every example consists of a LabVIEW and an ***ADbasic*** program:

– The LabVIEW program gives you control to the ***ADwin*** processes and displays the transferred data.

– The ***ADbasic*** program determines the processing on the ***ADwin*** system.

The LabVIEW programs are stored in the installation directory (see chapter 3.1), the ***ADbasic*** programs can be found in the folder

– Windows: <`C:\ADwin\ADbasic\samples_ADwin\`>.

– Linux: <`/opt/adwin/share/samples_ADwin/`>.

For better understanding, please have a look at the source codes as well as to the corresponding LabVIEW diagrams.

5 Functions of the *ADwin*-LabVIEW® Driver

The description of the VIs is divided into the following sections:

Please note the general notes on the use of ***ADwin*** VIs in chapter 4.

5.1 System control and information

Initialization of the ***ADwin*** system and information about the operating status.

Boot.VI

`Boot.VI` initializes the ***ADwin*** system and loads the file of the operating system.

Inputs

Device No.	Device No. of the system..
Filename	Path and filename of the operating system file, depending on processor type: T2: `ADwin2.btl` T4: `ADwin4.btl` T5: `ADwin5.btl` T8: `ADwin8.btl` T9: `ADwin9.btl` T10: `ADwin10.btl` T11: `ADwin11.btl` Default: `C:\ADwin\ADwin9.btl`.
Memsize	Processor type T2…T8 only: Built-in memory size (64 KiB…32 MiB)
error in	Connector for error signal.
showError	true: show error message with active error signal. false: no error message.

Outputs

Device No. out	The number connected to the input „Device No.“.
Memory	Status information: <1000: Error while booting 8000: Booting o.k.; T9, T10, T11 only >8000: Booting o.k. and correct MemSize value; T2…T8 only. Possible values MemSize: 10000: 64 KiB 100000: 1 MiB 200000: 2 MiB 400000: 4 MiB 800000: 8 MiB 1000000: 16 MiB 2000000: 32 MiB
error out	Error signal

Notes

The boot process clears the whole memory of the ***ADwin*** system: all program codes are lost, global variables are set to 0 (zero) and the parameter `Processdelay` is set to 1000.

The PC will only be able to communicate with the ***ADwin*** system, after the operating system has been loaded. Load the operating system every time you power-up the ***ADwin*** system.

Loading the operating system successfully with "`Boot`" lasts only 1 second.

To set the memory size you do a right click on the memsize input and select "`Create ▸ Constant`". In the next window you select the correct memory size.

Test_Version.VI

`Test_Version.VI` checks, if the correct operating system for the processor has been loaded and if the processor can be accessed.

Inputs

Device No.	Device No. of the system.
error in	Connector for error signal.
showError	true: show error message with active error signal. false: no error message.

Outputs

Device No. out	The number connected to the input „Device No.“.
Version	0: OK ≠0: Operating system has wrong version or does not respond.
error out	Error signal

Processor_Type.VI

`Processor_Type.VI` returns the processor type of the system.

Inputs

Device No.	Device No. of the system.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Processor type	Processor type: 0: error 2: T2 4: T4 5: T5 8: T8 9: T9 1010: T10 1011: T11
error out	Error signal

Workload.VI

`Workload.VI` returns the processor workload.

Inputs

Device No.	Device No. of the system.
priority	0: Total processor workload ≠0: no function
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Workload	≠255: Processor workload in % 255: Error
error out	Error signal

Free_Mem.VI

`Free_Mem.VI` determines the free memory.

Inputs

Device No.	Device No. of the system.
Processor	Type of processor or memory: 0: Processor type T2…T8 1: internal program memory (PM_LOCAL); as from T9 2: internal data memory (DM_LOCAL); as from T9 3: external data memory (DRAM_EXTERN); as from T9 4: internal extra memory (EM_LOCAL); T11 only
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Memory	≠255: Currently free memory (in bytes) 255: Error
error out	Error signal

5.2 Process control

Instructions for the control of processes on the ***ADwin*** system.

`Load_Process.VI` loads the binary file of a process into the ***ADwin*** system.

Load_Process.VI

Inputs

Device No.	Device No. of the system.
Filename	Path and filename of the binary file to be loaded
error in	Connector for error signal.
showError	true: show error message with active error signal. false: no error message.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	1: OK ≠1:Error
error out	Error signal

Notes

You generate binary files in ***ADbasic*** with "`Make Bin file`".

The last character of the binary file's filename shows the numberof the process in the ***ADwin*** system. A "0" stands for process 10.

`Start_Process.VI` starts a process on the system.

Start_Process.VI

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10) of the process.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255:OK 255:Error
error out	Error signal

Notes

The process starts with the priority in which it has been compiled in ***AD-basic***.

Stop_Process.VI

`Stop_Process.VI` stops a process on the system.

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10) of the process.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255:OK 255:Error
error out	Error signal

Clear_Process.VI

`Clear_Process.VI` deletes a process from memory.

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10, 15) of the process.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠1:OK 1: Error
error out	Error signal

Notes

This function is available only for systems with T9 or T10, T11 processor and not with Linux.

Loaded processes need memory in the system. You can delete processes from memory, in order to get more memory space; follow the order of the following steps:

1. stop the running process with `Stop_Process.VI`.
2. check if the process has stopped with `Process_Status.VI`.
3. delete the process from memory with `Clear_Process.VI`

Process 15 is responsible for the flashing of the LED in the Gold and Pro systems, after deleting it, the LED does not flash any more.

Processes should be cleared in reverse to the order they were loaded, i.e. het last loaded process is cleared first. This will avoid fragmentation of the program memory.

Please note, that clearing a process does NOT clear the arrays which were declared in the process.

Process_Status.VI

`Process_Status.VI` returns the status of a process.

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10, 15) of the process.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	0: Process is stopped ≠0: Process is running
error out	Error signal

Set_Processdelay.VI

`Set_Processdelay.VI` sets the parameter `Processdelay` for a process.

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10) of the process.
Processdelay	Value of `Processdelay` to be set.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

The parameter *`Processdelay`* controls the time interval between two events of a time-controlled process (see ***ADbasic*** manual).

The time interval is indicated in units of time, which are dependent on the processor type and the priority of a process.

Processor type	Process priority	
	high	low
T2, T4, T5, T8	1000ns	64µs
T9	25ns	100µs
T10	25ns	50µs
T11	3,3̄ns	0,003̄µs = 3,3̄ns

Get_Processdelay.VI

`Get_Processdelay.VI` returns the parameter `Processdelay` for a process.

Inputs

Device No.	Device No. of the system.
ProcessNo	Number (1…10) of the process.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: Value of the parameter `Processdelay` 255: Error
error out	Error signal

Notes

See more information under `Set_Processdelay.VI` or in the ***ADbasic*** manual.

5.3 Transfer of global variables

Instructions for data transfer between PC and ***ADwin*** system with the standard global variables PAR_1 ... PAR_80 (long) and FPAR_1 ... FPAR_80 (float), which are also called parameters.

5.3.1 Global long variables (PAR_1...PAR_80)

Set_Par.VI

Set_Par.VI sets a global long variable to the specified value.

Inputs

Device No.	Device No. of the system.
Index	Number (1...80) of a global Long variable PAR_1 ... PAR_80.
Value	New value for the variable.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Get_Par.VI

Get_Par.VI returns the value of a global long variable.

Inputs

Device No.	Device No. of the system.
Index	Number (1...80) of a global Long variable PAR_1 ... PAR_80.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Parameter value	≠255: Value of the selected variable 255: Error
error out	Error signal

Get_Par_Block.VI

`Get_Par_Block.VI` returns a number of global long variables, which is to be indicated.

Inputs

Device No.	Device No. of the system.
Startindex	Number (1…80) of the first global Long variable `PAR_1` … `PAR_80`, which is read.
Count	Number of variables which are transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	0: OK ≠0: Error
Data	Array with the read values.
error out	Error signal

Notes

The read values are saved in the array Data starting from array element 0. If e.g. Startindex = 5, the value of `PAR_5` is saved in the element 0 of Data .

5.3.2 Global float variables (FPAR_1…FPAR_80)

Set_FPar.VI

Set_FPar.VI sets a global float variable to a specified value.

Inputs

Device No.	Device No. of the system.
Index	Number (1…80) of a global Float variable FPAR_1 … FPAR_80.
Value	New value for the variable.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Get_FPar.VI

Get_FPar.VI returns the value of a global float variable.

Inputs

Device No.	Device No. of the system.
Index	Number (1…80) of a global Long variable FPAR_1 … FPAR_80.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Parameter value	≠255: Value of the selected variable 255: Error
error out	Error signal

Get_FPar_Block.VI

`Get_FPar_Block.VI` returns a number of global float variables, which is to be indicated, into an array.

Inputs

Device No.	Device No. of the system.
Startindex	Number (1…80) of the first global Float variable `FPAR_1` … `FPAR_80`, which is read.
Count	Number of variables which are transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	0: OK ≠0: Error
Data	Array with the read values.
error out	Error signal

Notes

The read values are saved in the array Data starting from array element 0. If e.g. Startindex = 9, the value of `FPAR_9` is saved in the element 0 of Data .

5.4 Transfer of data arrays

Instructions for data transfer between PC and ***ADwin*** system with global DATA arrays (DATA_1…DATA_200):

– DATA arrays

– FIFO Arrays

– DATA arrays with strings

You must define each array under ***ADbasic*** before usage (see ***ADbasic*** manual).

5.4.1 DATA arrays

You must define each DATA array under ***ADbasic*** before usage (see manual "***ADbasic***"): `DIM DATA_x[n] AS LONG/FLOAT`.

Data_Length.VI

`Data_Length.VI` returns the length of an array, declared under ***ADbasic***, that means the number of elements.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the DATA array DATA_1…DATA_200.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠0: Declared length of DATA array 0: Error or DATA array is not declared
error out	Error signal

SetData_Long.VI

Set_Data_Long.VI transfers data from a LabVIEW array into a DATA array of the system.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the DATA array DATA_1…DATA_200.
Data	Array with the values to be transferred.
Startindex	Index of the first element in Data to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

Alle values of the LabVIEW array are transferred (from the start element). The DATA array must be greater than the number of transferred values.

If the LabVIEW array contains FLOAT values, the values are changed into a 32-Bit LONG format during transfer.

GetData_Long.VI

GetData_Long.VI transfers parts of a DATA array from an ***ADwin*** system into a LabVIEW vector.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the DATA array DATA_1…DATA_200.
Startindex	Index of the first element in the selected array to be transferred.
Count	Number of values to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
Data	Array with the read values.
error out	Error signal

Notes

If the array DATA_x contains FLOAT values, the values are changed into a 32-Bit LONG format during transfer.

SetData_Float.VI

`SetData_Float.VI` transfers data from a LabVIEW array into a `DATA` array of the system.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the DATA array `DATA_1`…`DATA_200`.
Data	Array with the values to be transferred.
Startindex	Index of the first element in Data to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

Alle values of the LabVIEW array are transferred (from the start element). The `DATA` array must be greater than the number of transferred values.

If the LabVIEW array contains integer values, the values are changed into a 32-Bit FLOAT format during transfer.

GetData_Float.VI

`Get_Data_Float.VI` transfers parts of a `DATA` array from an ***ADwin*** system into a LabVIEW vector.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the DATA array `DATA_1`…`DATA_200`.
Startindex	Index of the first element in the selected array to be transferred.
Count	Number of values to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
Data	Array with the read values.
error out	Error signal

Notes

If the LabVIEW array contains integer values, the values are changed into a 32-Bit FLOAT format during transfer.

Data2File.VI

`Data2File` saves long or float data from a `DATA` array of the ***ADwin*** system into a file (on the hard disk).

Inputs

Device No.	Device No. of the system.
Mode	Writing mode: 0 : file will be created or overwritten (if file exists) 1 : data are appended to an already existing file
Filename	Path and file name.
DataNo	Number (1…200) of the DATA array `DATA_1`…`DATA_200`.
Startindex	Index of the first element in the selected array to be transferred.
Count	Number of values to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	0: OK ≠0: Error
error out	Error signal

Notes

The used `DATA` array must not be defined AS `FIFO` to use `Data2File`.

The data are saved in binary code.

5.4.2 FIFO Arrays

Instructions for data transfer between the PC and the ***ADwin*** system with global DATA arrays (DATA_1...DATA_200), which are declared as FIFO ring buffer (First in first out).

Before using it you have to declare each FIFO array under ***ADbasic*** (see manual ***ADbasic***): `DIM DATA_x[n] AS TYPE AS FIFO`

When working with FIFO arrays you should as a rule of thumb not write more elements into the FIFO as you have declared. You should definitely not read out more elements than you have written into the FIFO. Avoid this as follows:

– Check with the function `Fifo_Full`, if a FIFO array contains at least as many elements as you wish to read out with `Get_Fifo`.

– Check with the function `Fifo_Empty`, if a FIFO array contains at least as many free elements as you wish to write into the FIFO with `Set_Fifo`.

`Fifo_Empty.VI` returns the number of free elements of a FIFO array.

Fifo_Empty.VI

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array DATA_1…DATA_200.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Count	≠255: Number of free elements in the FIFO array. 255: Error
error out	Error signal

`Fifo_Full.VI` returns the number of used elements of a FIFO array.

Fifo_Full.VI

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array DATA_1…DATA_200.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
Count	≠255: Number of used elements in the FIFO array. 255: Error
error out	Error signal

Fifo_Clear.VI

`Fifo_Clear.VI` initializes the FIFO write and read pointers.

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array `DATA_1`…`DATA_200`.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

A FIFO array must be initialized before use, either in the ***ADbasic*** program or with `Fifo_Clear.VI`.

Initializing the FIFO pointers during the program is useful, when all values saved in the FIFO array are to be cancelled (for instance because of an error).

SetFifo_Long.VI

`SetFifo_Long.VI` transfers data from LabVIEW into a FIFO array of the ***ADwin*** system.

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array `DATA_1`…`DATA_200`.
Data	Array with the values to be transferred.
Count	Number of array elements to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

Check with the function `Fifo_Empty.VI`, if a FIFO array contains at least as many free elements as you wish to write into the FIFO with `Set_Fifo.VI`.

If the LabVIEW array contains FLOAT values, the values are changed into a 32-Bit LONG format during transfer.

GetFifo_Long.VI

`GetFifo_Long.VI` transfers FIFO data from the ***ADwin*** system to the PC.

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array `DATA_1`…`DATA_200`.
Count	Number of array elements to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
Data	Array with the read values.
error out	Error signal

Notes

Check with the function `Fifo_Full.VI`, if a FIFO array contains at least as many elements as you wish to read out with `Get_Fifo.VI`.

If the LabVIEW array contains FLOAT values, the values are changed into a 32-Bit LONG format during transfer.

SetFifo_Float.VI

`SetFifo_Float.VI` transfers data from LabVIEW to a FIFO array of the ***ADwin*** system.

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array `DATA_1`…`DATA_200`.
Data	Array with the values to be transferred.
Count	Number of array elements to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
error out	Error signal

Notes

Check with the function `Fifo_Empty.VI`, if a FIFO array contains at least as many free elements as you wish to write into the FIFO with `Set_Fifo.VI`.

If the LabVIEW array contains integer values, the values are changed into a 32-Bit FLOAT format during transfer.

GetFifo_Float.VI

`GetFifo_Float.VI` transfers FIFO data from the ***ADwin*** system to the PC.

Inputs

Device No.	Device No. of the system.
FifoNo	Number (1…200) of the FIFO array `DATA_1`…`DATA_200`.
Count	Number of array elements to be transferred.
error in	Connector for error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠255: OK 255: Error
Data	Array with the read values.
error out	Error signal

Notes

Check with the function `Fifo_Full.VI`, if a FIFO array contains at least as many elements as you wish to read out with `Get_Fifo.VI`.

If the LabVIEW array contains integer values, the values are changed into a 32-Bit FLOAT format during transfer.

5.4.3 DATA arrays with strings

Instructions for string transfer between the PC and the ***ADwin*** system with global DATA arrays (DATA_1…DATA_200).

Before using it you have to declare each array under ***ADbasic*** (see manual ***ADbasic***): `DIM DATA_x[n] AS STRING`.

SetData_String.VI

`SetData_String.VI` transfers a string from LabVIEW into a DATA array in the ***ADwin*** system.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the array DATA_1…DATA_200.
Data	Array with the string to be transferred.
error in	Error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠-1: Number of transferred characters -1: Error
error out	Error signal

GetData_String.VI

`GetData_String.VI` transfers a string from a DATA field in the ***ADwin*** system to LabVIEW.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the array DATA_1…DATA_200.
MaxCount	Max. number of characters to be transferred.
error in	Error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠-1: Number of transferred characters -1: Error
Data	Array with the read values.
error out	Error signal

Notes

A string in a DATA array ends with a delimiter (ASCII value 0). The transfer ends exactly at this position. The return value is the number of transferred characters without the end delimiter.

If MaxCount is greater than the number of string chars defined in ***ADbasic***, you will receive the error "Data too small" via the Error_msg.VI.

If you set *MaxCount* to a high value, the function will have an appropriately long execution time, even if the transferred string is short.
For time-critical applications with large strings, it may be faster to proceed as follows:

- You determine the actual number of chars in the string using the String_Length.VI.
- You read the string with Getdata_String.VI and pass the actual number of chars as MaxCount.

String_Length.VI

String_Length.VI returns the length of a string in a DATA-Feld on the ***ADwin*** system.

Inputs

Device No.	Device No. of the system.
DataNo	Number (1…200) of the array DATA_1…DATA_200.
error in	Error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
return value	≠-1: Length of string in the DATA array. -1: Error
error out	Error signal

Notes

String_Length.VI counts the characters in a DATA array until an end delimiter is found (ASCII value 0). The end delimiter is not counted as a character.

5.5 Control and error handling

Error_msg.VI

`Error_msg.VI` shows the error message corresponding to the last error which occurred with an ***ADwin*** system.

Inputs

show error dialog?	true: show error message false: no function
error in	Error signal.

Outputs

error ?	true: error signal is present false: no error
error out	Error signal

Notes

The output "error ?" is applicable to stop the program in case of an error e.g. in a loop.

You find a list of all error messages in section A.2 in the annex.

Set_Language.VI

`Set_Language.VI` sets the language for the error messages.

Inputs

Device No.	Device No. of the system.
Language	Used language: 0 : language is set by Windows 1 : English 2 : German
error in	Error signal.

Outputs

Device No. out	The number connected to the input „Device No.“.
error out	Error signal

Notes

If Windows is set to a different language than English or German, the error messages are shown in English.

5.6 Tools

Volt2Value.VI

`Volt2Value.VI` converts a voltage value into the appropriate digit value for a DAC.

Inputs

Input voltage	Single voltage value in Volt.
Input voltage array	Array with voltage values in Volt.
Input voltage array 2dim.	2-dimensional array with voltage values in Volt.
Converter resolution	Resolution of the ADC.
Voltage range	Voltage range of the ADC.

Outputs

Output value	Digit value to the input "Input voltage".
Output value array	Digit value to the input "Input voltage array".
Output value array 2dim.	Digit value to the input "Input voltage array 2dim.".

Notes

The VI calculates a digit value starting from a voltage value. If a DAC is set to the calculated digit value, the DAC set the voltage on the analog ouput. Set the inputs "Converter resolution" and "Voltage range" according to the used DAC.

The inputs "Input voltage" may be used as single oder in parallel.

Value2Volt.VI

`Value2Volt.VI` converts a digit value of an ADC into the appropriate voltage value.

Inputs

Input value	Single digit value.
Input value array	Array with digit values.
Input value array 2dim.	2-dimensional array with digit values.
Converter resolution	Resolution of the ADC.
Voltage range	Voltage range of the ADC.

Outputs

Output voltage	Voltage value to the input "Input value".
Output voltage array	Voltage value to the input "Input value array".
Output voltage array 2dim.	Voltage value to the input "Input value array 2dim.".

Notes

If an ADC returns a digit value, the `Value2Volt.VI` calculates the voltage value which is set at the ADC input. Set the inputs "Converter resolution" and "Voltage range" according to the used ADC.

The inputs "Input value" may be used as single oder in parallel.

Annex

A.1 Index of Functions

No.	Error message
0	No Error.
1	Timeout error on writing to the ADwin-system.
2	Timeout error on reading from the ADwin-system.
10	The device No. is not allowed.
11	The device No. is not known.
15	Function for this device not allowed.
20	Incompatible versions of ADwin operating system , driver (ADwin32.DLL) and/or ADbasic binary-file.
100	The Data is too small.
101	The Fifo is too small or not enough values.
102	The Fifo has not enough values.
150	Not enough memory or memory access error.
200	File not found.
201	A temporary file could not be created.
202	The file is not an ADBasic binary-file.
203	The file is not valid.[1]
204	The file is not a BTL.
2000	Network error (TcpIp).
2001	Network timeout.
2002	Wrong password.
3000	USB-device is unknown.
3001	Device is unknown.

1. Possibly the file <ADwin5.btl> has no memory table, or another file was renamed to <ADwin5.btl> or the file is damaged.