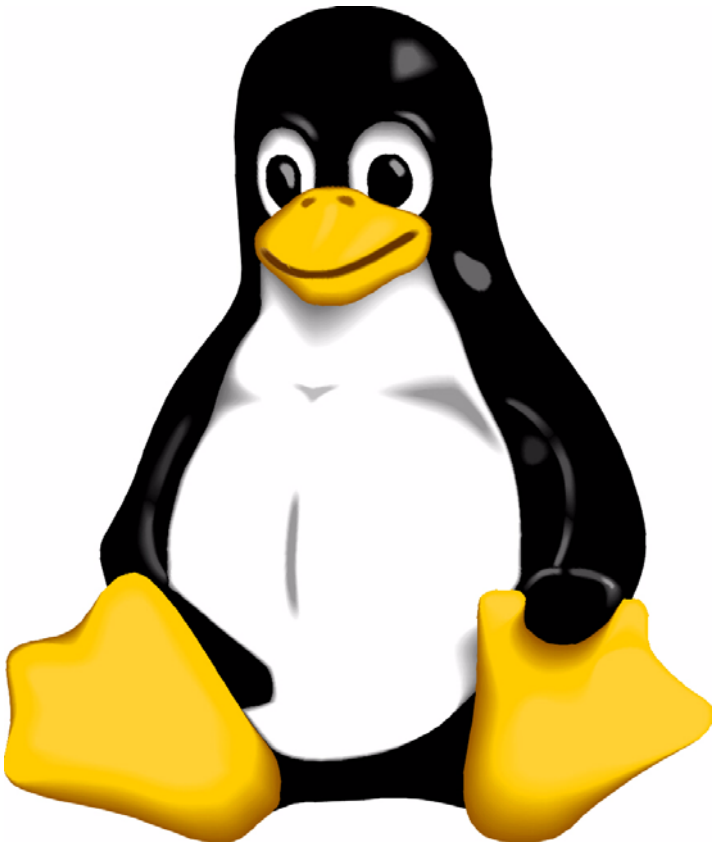
ADwin Linux

Manual



ADwin driver Linux, Manual version 2.2, October 2005

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

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

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

Table of contents

Typographical Conventions

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

You find a "note" next to

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

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

<C:\ADwin\ …>

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

Program text

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

Var_1

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

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

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

1 Information about this Manual

This manual contains comprehensive information about using the interface library ***ADwin*** Linux and the ***ADbasic*** compiler.

Additional information is available in

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

– the manual "***ADbasic***", which contains all instructions for the compiler ***ADbasic***. You will also find information about
  • the structure of ***ADbasic*** programs
  • the optimization of ***ADbasic*** programs
  • the run-time behaviour of processes

  A comfortable user interface as real-time development tool is available for your programming tasks – up to this time only for Windows. You may also use an editor of your choice.

– for ***ADwin-Pro*** systems only: Software manual with the instructions for programming the Pro modules.

– the hardware manual for the ***ADwin*** system you are using.

Knowledge in handling the development environment or programming language is assumed.

Please note:

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

Qualified personnel

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

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

Availability of the documents

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

Legal instructions

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

Subject to change.

Hotline address: see inner side of cover page.

2 *ADwin* Linux

ADwin Linux is a library. It provides interface functions, so that development environments, such as C, C++ or LabView can communicate with the ***ADwin*** system.

The library ***ADwin*** Linux defines methods, functions and system variables for process control and data transfer from and to the ***ADwin*** system. The definition includes the name of a function as well as number, data type and order of the parameters to be transferred.

Communication with the *ADwin* system

With the development environment you can control processes in the ***ADwin*** system, get data from there or send data to the system. The processes are programmed in ***ADbasic*** (see ***ADbasic*** manual) and compiled with the compiler which is provided as standard delivery item (see chapter 4).

Real-time operating system

The library functions communicate with the real-time operating system of the ***ADwin*** system. Therefore the operating system must be loaded after each power-up (e.g. the file `<adwin9.btl>`). 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`), already specified
  - 80 float variables (`FPAR_1` … `FPAR_80`), already specified
  - 200 data arrays of arbitrary length (`DATA_1`…`DATA_200`), user specified

  The values of these variables or data arrays can be changed or read from the development environment at any time. Local variables and arrays can be defined.

Communication

– Communication between ***ADwin*** system and PC.

  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 the PC to the ***ADwin*** system: Control instructions and instructions for data exchange. The following table shows examples for each group.

Control instructions, e.g.	
`LOAD_PROCESS`	transfers a process to the system
`START_PROCESS`	starts a process
Instructions for data exchange, e.g.	
`GET_PAR`	returns the current value of a parameter
`SET_PAR`	changes the value of a parameter
`GETDATA_LONG`	returns the values of a `DATA` array

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

3 Installation and Initialization of *ADwin* under Linux

The installation of the Linux library contains the following steps:

– Software Configuration and Installation

– Configuration of the ADwin System

– Including the Library in C

For parts of the installation "root" privileges are necessary. If root privileges are required, the # sign is indicated at the beginning of an edit line (otherwise the $ sign).

3.1 Software Configuration and Installation

The `./LINUX` directory with the following tar archives can be found on the ***ADwin*** CD-ROM:

CDROM

– `adwin-lib-x.y.tar.gz`: The archive (version `x.y`) contains:
  • the kernel module for ***ADwin*** Linux.
  • the ***ADwin*** Linux library (shared library)
  • the configuration program "`adconfig`"

– `adwin-compiler-x.y.tar.gz`: The ***ADbasic*** compiler

– `adwin-labview-x.y.tar.gz`: The driver for the LabView development environment (for complete installation, see manual "LabView Driver").

– `adwin-doc-x.y.tar.gz`: All documentations for software and hardware as pdf format.

Installing the archives

Install the archives in the same order as indicated above. Start with `adwin-lib-x.y.tar.gz` (see notes below).

To install an archive, proceed as follows:

1. Extract the archive `<filename>.tar.gz`:
```
$ cd ~
$ tar -xvzf <cdrom-path>/<filename>.tar.gz
$ cd <filename>
```
2. Go on with the usual sequence:
```
$ ./configure
$ make
# make install
```
   The files are installed in the directory `/opt/adwin/`. (We assume this configuration in the following text).
   If you want to use another installation directory execute the `configure` script using the following option:
```
$ ./configure --prefix=/your-dir-here
```
3. Install the other archives in the same way.

Take notice of the following information when you install the archive `adwin-lib-x.y.tar.gz`:

– The kernel module is generated and installed in the directory `/lib/modules/`, the library and the configuration program in the directory `/opt/adwin/`. The default settings are for the kernel version 2.4.

Kernel 2.6

For kernel version 2.6 use the following options:
```
$ ./configure --disable-link --disable-kernel-module
```

– If you haven't compiled a Linux kernel before, `make` may fail due to missing files. In this case, try the following lines:
```
$ cd /usr/src/linux
$ make oldconfig dep
```
Go on with the usual `configure`, `make`, `make install` sequence decribed above. If you are not successful call our hotline (inner side of cover page).

Kernel version

– The kernel module should be stable with all versions of the kernel 2.4. The kernel versions are for instance available under `ftp.kernel.org`.

If a problem nevertheless occurs, we will be able to help you when you send us the following information:

- A brief description of the problem
- The kernel version and patches you are using
- The file `.config` of the kernel

Environment variable

In earlier versions of the Linux library the installation directory was stored in the environment variable `ADWINDIR`. This is possible further on, but it is not required.
To set the variable, you may either

- indicate the directory (see `configure`) directly:
```
$ export ADWINDIR=/opt/adwin/
```
- or use a file (automatically generated from `make` / `make install`):
```
$ export ADWINDIR=´cat /etc/adwin/ADWINDIR/´
```

If you use `bash`, the following line may be useful in addition:
```
$ export PATH=$PATH:$ADWINDIR/bin:$ADWINDIR/sbin
$ export MANPATH=$MANPATH:$ADWINDIR/man
```

3.2 Configuration of the *ADwin* System

Each ***ADwin*** system is accessed under Linux via a "device no." To configure the device number use the program `adconfig` in the directory `/opt/adwin/sbin/`:

– Add the device no. of the ***ADwin*** system under Linux (there is no default device no. after a fresh installation). The PC uses the log-on data to access the ***ADwin*** system.

– If the ***ADwin*** system has an Ethernet interface, you must additionally configure the system. In the "ADwin Driver Installation" manual you will find a description about the basics of Ethernet operation.

  The configuration data are not stored on the PC, but are transferred to the ***ADwin*** system. The data must correspond to the log-on data.

The `adconfig` program requires root privileges (`su`).

3.2.1 Example: Standard configuration

The following lines explain how to configure the device no. 0x150 for an ***ADwin*** system with Ethernet interface:

1. Add the device no. 0x150 (decimal: 336) with a fixed IP address (here: 10.100.100.83) under Linux:
   ```
   # adconfig add 0x150 TYPE net IP 10.100.100.83
   ```
2. Display the results of adding the device no.
   ```
   # adconfig
   # DeviceNo=0x150 Type=net IP=10.100.100.83 Port=6543
    Timeout=1000 PackageCount=5
   ```
3. Configure the ***ADwin*** system with MAC address, IP address and subnet mask:
   ```
   # adconfig config 00:50:C2:0A:22:DD IP 10.100.100.83
    MASK 255.255.255.00
   ```

The device no. 0x150 is now configured and you can start working with the ***ADwin*** system.

3.2.2 Configuration in detail

With the `adconfig` program you can execute the following functions:

– Display the list of all device numbers under Linux:
  ```
  $ adconfig
  ```

– Add a new device no. under Linux:
  ```
  # adconfig add <deviceno> [OPTIONS] TYPE link|net
  ```

– Configure an ***ADwin*** system with Ethernet interface
  ```
  # adconfig config <deviceno> [OPTIONS]
  ```

  Use the same data you have used when adding the device number!

– Delete a device no.:
  ```
  # adconfig del <deviceno>
  ```

– Display the help for the `adconfig` program:
  ```
  $ adconfig --help
  ```

– Display the version number of the program:
  ```
  $ adconfig --version
  ```

The parameter `<deviceno>` can be indicated in decimal or hexadecimal notation, e.g. 336 or 0x150.

You will find more details about the functions of the program in the following text or you can display them with `man 8 adconfig`.

Displaying the device no.

Displaying the list of all device numbers

```
$ adconfig
```

A list of all configured device numbers with their parameters is displayed. The device numbers are displayed in hexadecimal notation i.e. `0x150` for decimal 336.

Adding a new link device no. under Linux

Adding the device no.: Link

This function is only available with kernel 2.4.

```
# adconfig add <deviceno> [UID <uid|username>]
  [GID <gid|groupname>][MODE <mode>][DESC <"">]
  <TYPE link BASE <#>>
```

`<deviceno>`	The device no. to add
`UID <uid|username>`	Name of the user (default: root)
`GID <gid|groupname>`	Name of the user group (default: root)
`MODE <mode>`	Access rights of the user (default: 0666)
`DESC <"">`	Description (up to 32 chars, default: "")
`TYPE link BASE <#>`	***ADwin*** system with link data connection <#>: link base address (e.g. 0x150)

You must set the device no. on the ***ADwin*** board as link address. You will find the description in the installation manual.

Enter only parameters, which have no default settings or whose default settings you want to change.

Adding the device no.: Ethernet

Add a new Ethernet device no. under Linux

```
# adconfig add <deviceno> [UID <uid|username>]
  [GID <gid|groupname>][MODE <mode>][DESC <"">]
  <TYPE IP <addr> [PW <"">][PORT <#>][TIMEOUT <#>]>
```

`<deviceno>`	The device no. to add.
`UID <uid|username>`	Name of the user (default: root)
`GID <gid|groupname>`	Name of the group (default: root).
`MODE <mode>`	User access rights (default: 0666)
`DESC <"">`	Description (up to 32 chars, default "")
`TYPE net IP <addr>`	***ADwin*** system with Ethernet interface. `<addr>`: IP address in the Ethernet network
`PW <"">`	Password (default: "")
`PORT <#>`	Network port number (default: 6543)
`TIMEOUT <#>`	Timeout of the network protocol (default: 1000 ms)

Enter only parameters, which have no default settings or whose default settings you want to change.

The access rights set by `UID`, `GID` and `MODE` do merely protect against accidentally faulty access, because they can be bypassed.

The installation of an ***ADwin*** system in an Ethernet network is described in the installation manual. Here you will also find a more detailed description of the network-specific parameters.

Pay attention that you do not have the same IP address in an Ethernet network twice. Otherwise massive communication problems may occur.

Example

```
# adconfig add 0x3A TYPE net IP 10.100.100.83
```

Display the results of the configuration:

```
# adconfig
# DeviceNo=0x3A Type=net IP=10.100.100.83 Port=6543
 Timeout=1000 PackageCount=5
```

Adding the Device no. - *ADwin* system and host

In an IP network (e.g. Ethernet) a client PC can communicate with an ***ADwin*** system conected to another PC (host PC) even if the system has a link or USB interface, or if it is running in a different network.

It is required that

- the host PC has access to the ***ADwin*** system,
- the host PC can be accessed from the client PC via network
- the program `ADwinTcpipServer` is running on the host PC (currently available for Windows only).

If the requirements are met, the following log-on parameters must be set:

```
# adconfig add <deviceno> …
 TYPE net IP <addr> [PW <"">][PORT <#>]
 [TIMEOUT <#>] HOSTLINK <#>
```

`net IP <addr>`	IP address of the host PC

`PW <"">`
`PORT <#>`
`TIMEOUT <#>`
	The parameters must be identical to the settings you made in the host PC program `ADwinTcpip-Server` (see online help of the program).

`HOSTLINK <#>`	Additionally: Device no. of the ***ADwin*** system, set with `adconfig` at the host PC.

Adding the Device no: *ADwin* system and host

Configuring the Ethernet device without DHCP

Configuring an *ADwin* system with Ethernet interface (without DHCP)

```
# adconfig config <hwaddr> IP <addr> MASK <addr>
 [GATEWAY <addr>] [PORT <#>] [PW <"">]
```

`<hwaddr>`	MAC address of the ***ADwin*** system in the format `00:50:C2:0A:nn:nn`.
`IP <addr>`	IP address of the ***ADwin*** system in the Ethernet network in the format `nnn.nnn.nnn.nnn`.
`MASK <addr>`	Subnet mask in the format `nnn.nnn.nnn.nnn`.
`GATEWAY <addr>`	IP address for the default gateway (default: 0.0.0.0, i.e. no gateway)
`PORT <#>`	Network port number (default: 6543)
`PW <"">`	New password (default: "", i.e. the existing password will be deleted!)

Enter only parameters, which have no default settings or whose default settings you want to change. Do use the same values you have used with `add`!

Please note that the password is always transferred during configuration. If you want to keep an already existing password, you must therefore indicate the password, otherwise it will be deleted.

Example

```
# adconfig config 00:50:C2:0A:22:DD IP 10.100.100.83
 MASK 255.255.255.00
```

Configuring the Ethernet device with DHCP

Configuring an *ADwin* system with Ethernet interface for DHCP

It is possible but not recommended that an ***ADwin*** system gets its IP address from the DHCP server (see option `DHCP`).

```
# adconfig config <hwaddr> [GATEWAY <addr>] [PORT <#>]
 DHCP yes [PW <"">]
```

`<hwaddr>`	The MAC address of the ***ADwin*** system in the format `00:50:C2:0A:nn:nn`.
`GATEWAY <addr>`	IP address for the default gateway (default: 0.0.0.0, i.e. no gateway)
`PORT <#>`	Network port number (default: 6543)
`PW <"">`	New password (default: "", i.e.the existing password will be deleted!)
`DHCP <no | yes>`	Here you determine if a DHCP server is used (default: `no`); see below.

We recommend that the option `DHCP` should not be activated!
If this option is active, communication problems may occur frequently.

Enter only parameters, which have no default settings or whose default settings you want to change. Do use the same values you have used with `add`!

Deleting a device no.

Deleting a device no.

```
# adconfig del <deviceno>
```

The device no. is deleted under Linux and cannot be accessed any more.

3.3 Including the Library in C

To use the functions of the library ***ADwin*** Linux in C or C++, go on as follows:

– Include the following lines into your program:
  - Functions for accessing the ***ADwin*** system:
    `#include <libadwin.h>`
  - Error numbers, you may get when accessing the ***ADwin*** system:
    `#include <libadwin/errno.h>`

– Include the following flags into your Makefile for the C compiler, to compile your program with the library ***ADwin*** Linux:
```
$ CFLAGS += -I$(ADWINDIR)/include -L$(ADWINDIR)/lib
–ladwin
$ CFLAGS += -Wl,--rpath -Wl,$(ADWINDIR)/lib
```
  Note that here the environment variable `ADWINDIR` is used. (For setting the variables see page 4).

  If you use threads you have to add:
```
$ CFLAGS += -D_REENTRANT
```

Porting from Windows

The functions of the ***ADwin*** Linux library are compatible to the functions of the interface under Windows (`adwin32.dll`). Therefore program source codes written in a programming language (here: C/C++) under Windows can normally be used under Linux without making any modifications (of course only with regards to accessing the ***ADwin*** systems).

Modifications in the program are only necessary where path or file names are used. Here the typical differences between Windows and Linux become apparent: Syntax and case sensitivity of the path names.

For ***ADbasic*** source codes porting from (and to) Windows is likewise possible.

4 Compiling *ADbasic* Source Code

In an ***ADbasic*** source code you program the functions of a process that is running on the ***ADwin*** system. You generate an ***ADbasic*** source code with an editor of your choice. A comfortable Windows user interface as real-time development tool is available.

The ***ADbasic*** compiler is called under Linux in the command line (shell). It compiles a source code file and generates – depending on the options and parameters – either a binary file or a library file. The function of the binary and library files is described in the ***ADbasic*** manual.

The license key must be entered with the option `-k` before the compiler can be used. Only then will you be able to generate binary or library files.
You will find the license key on the cover sheet of the ***ADbasic*** manual.

One or more command line calls may be combined in a shell script or Makefile, to compile several source codes of a project with only one call.

4.1 Compiler Syntax

There are 5 main options when calling the ***ADbasic*** compiler:

1. `adbasic -v`	Display the compiler version number
2. `adbasic -h`	Call the help
3. `adbasic -k <license>`	Enter license key (see cover sheet of the ***ADbasic*** manual). A write authorization is necessary for `/opt/adwin/var/license/`.
4. `adbasic [OPTIONS] <filename>`	Compile an ***ADbasic*** source code and generate a binary file.
5. `adbasic -l [OPTIONS] <filename>`	Compile an ***ADbasic*** source code and generate a library file.

The `[OPTIONS]` for compiling source code are described below.

Syntax

```
[path/]adbasic [-p <ptype>] [-s <system>]
       [-n <#>] [-e <event>] [-g <cycle>]
       [-r <prio>] [-l] [-L <path/>] [-I <path>]
       [-a [path/]<fout>] [path/]infile.bas
```

Switches

`[path/]` Directory where you find the program `adbasic`, at standard installation: `/opt/adwin/bin/`

`-p<ptype>` Processor type for which the file is to be compiled:

`-p2`	Processor T2
`-p4`	Processor T4
`-p5`	Processor T5
`-p8`	Processor T8
`-p9`	Processor T9; default
`-p10`	Processor T10
`-p11`	Processor T11

`-s<system>` ***ADwin*** system for which the file is to be compiled:

`-sc`	Boards (ISA bus)
`-sl`	Light-16
`-sg`	Gold; default
`-sp`	Pro

`-n<#>`	Sets the process number (1…10) of the file to be compiled; default = 1.
`-e<event>`	Select the event source for the process: `-et` Timer; default `-ee` External
`-g<cycle>`	Sets the 'Initial Processdelay'; default = 1000.
`-r<prio>`	Sets the priority of the process to: `-rh` High; default `-rl` Low `-rln` Low, Level n (1…255); default: n=1. Level 255 has the highest priority.
`-l`	Compile source code and generate a library file with the extension "`.liX`". If the option is not set a binary file with the extension "`.tXY`" is generated. X = 2, 4, 5, 8, 9, a (corresponds to processor type `-p`) . Y = 1…9, 0 (corresponds to process number `-n`: 1…10) .
`-L<path>`	Sets the search path for library files; default `/opt/adwin/share/lib/`
`-I<path>`	Sets the search path for include files; default `/opt/adwin/share/inc/`
`-a<fout>`	Optional: Name of the binary or library file to be generated, without file extension.
`infile.bas`	File name of the source code to be compiled. Path is optional.

Notes

Optional switches are set in square brackets. The order of the options is arbitrary. In command lines the writing is case sensitive.

If the option `-a` is not used the generated binary or library file is saved in the directory where the source code is located.

The following options mutually exclusive:

Option	excludes
`-sg`,`-sl`	`-p2`, `-p4`, `-p5`, `-p8`
`-sp`	`-p2`
-l	`-n`, `-e`, `-g`, `-r`, `-d`, `-o`

If an error occurs during compilation, the compiler stops and does not generate any binary or library files.

4.2 Examples

```
$ /opt/adwin/bin/adbasic -l /home/user/test.bas
```

The source code file `test.bas` is compiled and generates the library file `test.li9` in the directory `/home/user/` .
Since no other options than `-l` are indicated, the default settings are used:

- Processor: T9
- System: Gold
- Process number: 1
- Event: Timer
- Initial Processdelay: 1000
- Priority: High
- Include-path: `/opt/adwin/share/inc/`
- Library path: `/opt/adwin/share/lib/`

If you include the directory where the compiler `adbasic` is located, into your search path (see "Installation" page 3), you can write the line above in a shorter way:
`$ adbasic -l /home/user/test.bas`

In the following examples we assume that the `adbasic` path is part of your search path.

```
$ adbasic -l /string.bas -sp
```

The command line call compiles the source code file `string.bas` to a library file for a Pro system with a T9 processor.
The same call, but for a T10 processor, is as follows:
`adbasic -l /test.bas -p10 -sl`

```
$ adbasic /home/user/test.bas
/opt/adwin/share/example/bas_dmo6f -p9 -sg
```

Compiles the demo file `bas_dmo6f.bas` into a binary file for an ***ADwin-Gold*** system with T9 processor.

```
$ adbasic /opt/adwin/share/example/bas_dmo6 -p8 -sc
```

Compiles the demo file `bas_dmo6.bas` into a binary file for an ***ADwin*** board with a T8 processor.

```
$ adbasic /home/user/my_file.bas -p4 -sc -ayour_file
```

Compiles the file `my_file.bas` for an ***ADwin*** board with a T4 processor. The generated binary file has the name `your_file.t41` and can be found in the current directory.

```
$ adbasic /home/user/my_file.bas -a/somewhere/your_file
```

The name of the binary file is `your_file.t91` and is located in the directory `/somewhere`.

5 Methods and Functions of *ADwin* Linux

The syntax of ***ADwin*** Linux methods and functions depends on the development environment.
This is the reason why we have used a general syntax in the description below. Use the relevant syntax for your development environment.

The syntax of the instructions is illustrated below:

```
datatype instr (datatype Var1, datatype Var2, …)
```

Meaning

`datatype`	With an instruction: Data type of the return value. With a parameter: Data type of the parameter. Data types are: `void`, `long`, `float`, `double`, `char`. Note that here the data type `long` is equal to `signed int 32bit`.
`instr`	In some development environments (C, C++) instruction names are case sensitive.
`Var1`, *`Var2`*	Variables and return values are described under "Parameters", behind the instruction syntax.

Instructions for accessing analog and digital inputs and outputs (and other hardware functionalities) are not included in the ***ADwin*** Linux library. Such applications are programmed in ***ADbasic***.

5.1 System Control and System Information

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

Set_DeviceNo

The function `Set_DeviceNo` sets the device number of the ***ADwin*** system for all following instructions.

```
void Set_DeviceNo(long DeviceNo)
```

Parameters

`DeviceNo`	Device number of the system. Typical device numbers are 336 (= 0x150 hexadecimal), 400 (= 0x190 hexadecimal).

Notes

The PC distinguishes and accesses the ***ADwin*** systems using this device number. Systems with link adapter are already delivered with the default setting: 0x150h.

The function `Set_DeviceNo` does not communicate with the ***ADwin*** system and therefore does not affect error handling.

Example

```
Set_DeviceNo(3)
```

The device number 0x3 is set. All further instructions refer to this device (until a new device number is set with `Set_DeviceNo`).

Booting

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

```
void Boot (char pFilename())
```

Parameters

Filename	pointer to the file name of the operating system file

Notes

The initialization deletes all processes on the system and sets all global variables to the value 0.

The operating system which is to be loaded depends on the processor. The following table shows the file names for the different processors.

Processor	Short name	Operating system file
T225	T2	`ADwin2.btl`
T400	T4	`ADwin4.btl`
T450	T5	`ADwin5.btl`
T805	T8	`ADwin8.btl`
ADSP 21062	T9	`ADwin9.btl`
ADSP 21162	T10	`ADwin10.btl`
ADSP TS101S	T11	`ADwin11.btl`

The PC will only be able to communicate with the ***ADwin*** system after the operating system has been loaded. Reload the operating system after each power-up of the ***ADwin*** system.

Loading the operating system successfully with `Boot` takes about 1 second.

Test_Version

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

```
long Test_Version()
```

Parameters

return value	0 : OK 1 : wrong driver version, processor continues working 2 : wrong driver version, processor stops 3 : no response from the ***ADwin*** system

Processor_Type

`Processor_Type` returns the processor type of the system.

```
long Processor_Type()
```

Parameters

return value	processor code number of the system

Notes

The function returns the used processor type for testing purposes, according to the table below.

return value	processor type
0	Error
2	T200
4	T400
5	T450

return value	processor type
8	T800
9	T9
1010	T10
1011	T11

Workload

`Workload` returns the processor workload.

```
long Workload (long Priority)
```

Parameters

Priority	0 (zero): current total workload of the processor $\neq 0$: not supported at the moment
return value	processor workload (in percent)

Free_Mem

`Free_Mem` determines and returns the free memory for different memory types.

```
long Free_Mem (long Mem_Spec)
```

Parameters

Mem_Spec	type of memory 0: all types of memory; processors T2, T4, T5, T8 only 1: local program memory (PM_LOCAL); up from T9 2: local extra memory (EM_LOCAL); up from T11 3: local data memory (DM_LOCAL); up from T9 4: external DRAM memory (DRAM_EXTERN); up from T9
return value	currently free memory (in bytes)

Note

The different memory areas are described in more detail in the ***ADbasic*** manual.

5.2 Process Control

Instructions for controlling processes on the ***ADwin*** system.

Load_Process

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

```
void Load_Process (char pFilename())
```

Parameters

Filename	pointer to the file name of the binary file

Notes

You generate a binary file under Linux by calling the compiler `adbasic` (without the option `-l`, see chapter 4.1).

Switching a system on and off of deletes loaded processes. You must therefore load the necessary processes again after power-up.

Start_Process

`Start_Process` starts a process.

```
void Start_Process (long ProcessNo)
```

Parameters

ProcessNo	number of the process (1…10)

Stop_Process

`Stop_Process` stops a process.

```
void Stop_Process (long ProcessNo)
```

Parameters

ProcessNo	number of the process (1…12, 15)

Note

It may happen that a process cannot be stopped immediately after the instruction has been executed, but somewhat later. The status of the process is queried using `Process_Status`.

Clear_Process

`Clear_Process` deletes a process from memory.

```
void Clear_Process (long ProcessNo)
```

Parameters

ProcessNo	number of the process (1…12, 15)

Notes

Loaded processes allocate memory in the system. You can delete processes from memory with `Clear_Process`, in order to get more memory space for other processes.

A process, which is to be deleted, must not be running. Stop a running process first with `Stop_Process`, (and determine with `Process_Status` that the process has stopped), before you delete it from memory with `Clear_Process`.

After power-up the internal processes 11, 12 and 15 are running on the ***ADwin*** system. If you want to delete them do so in the following order: 15, 12, 11.
Process 15 flashes the LED in the Gold and Pro systems. After deleting it, the LED does not blink any more.

Process_Status

`Process_Status` returns the status of a process.

```
long Process_Status (ProcessNo)
```

Parameters

ProcessNo	number of the process (1…12, 15)
return value	status of the process 1 : process is running. 0 : process is not running, that is, it is not loaded, it has not been started or stopped. -1: process is being stopped, that is, it has received a `Stop_Process`, but still waits for the last event.

Set_Globaldelay

`Set_Globaldelay` sets the parameter `Globaldelay` for a process

```
void Set_Globaldelay (long ProcessNo,
       long Globaldelay)
```

Parameters

ProcessNo	number of the process (1…10)
Globaldelay	value to be set for the parameter `Globaldelay`.

Notes

The parameter *Globaldelay* 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 depend on the processor type and the priority of a process.

Processor type	Process priority	
	high	low
T2, T4, T5, T8	1 µs	64 µs
ADSP 21062 (T9)	0.025 µs (=25 ns)	100 µs
ADSP 21162 (T10)	0.025 µs (=25 ns)	50 µs

Example (C)

```
Set_Globaldelay(1,2000);
// the Globaldelay of process is set to 2000.
```

This time-controlled process is called every 50 µs (=2000 · 25 ns) in a high priority process and when using an T9 processor.

Get_Globaldelay

`Get_Globaldelay` returns the parameter `Globaldelay` of a process.

```
long Get_Globaldelay (long ProcessNo)
```

Parameters

ProcessNo	number of the process (1…10)
return value	currently set value for *Globaldelay*; in case of error: 255

Notes

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

Example (C)

```
long erg;
erg = Get_Globaldelay(1);
// query Globaldelay of ADbasic process 1
```

5.3 Transfer of Global Variables

Instructions for data transfer between PC and ***ADwin*** system with the standard global variables PAR_1 … PAR_80 and FPAR_1 … FPAR_80.

5.3.1 Global long variables (PAR_1 … PAR_80)

Set_Par

Set_Par sets a global long variable to the specified value.

```
void Set_Par (long Index, long Value)
```

Parameters

Index	number of the global long variable (1 … 80)
Value	value to be set for the long variable

Get_Par

Get_Par returns the value of a global long variable.

```
long Get_Par (long Index)
```

Parameters

Index	number of the global long variable (1 … 80).
return value	current value of a variable

Get_Par_Block

Get_Par_Block transfers the indicated amount of global long variables into an array.

```
void Get_Par_Block (long Array(), long StartIndex,
       long Count)
```

Parameters

Array	pointer to an array (Array length ≥ *Count*)
StartIndex	number of the first variable to be transferred (1…80)
Count	number of variables to be transferred (max. 80)

Get_Par_All

Get_Par_All transfers all global long variables into an array.

```
void Get_Par_All (long Array())
```

Parameters

Array	pointer to an array (array length ≥ 80)

5.3.2 Global float variables (FPAR_1…FPAR_80)

Set_FPar

Set_FPar sets a global float variable to a specified value.

```
void Set_FPar (long Index, long Value)
```

Parameters

Index	number of the global float variable (1…80)
Value	value to be set for a float variable

Get_FPar

Get_Par returns the value of a global float variable.

```
float Get_FPar (long Index)
```

Parameters

Index	number of the global float variable (1…80)
return value	current status of the variables

Get_FPar_Block

Get_FPar_Block transfers the indicated amount of global float variables into an array.

```
void Get_FPar_Block (float Array(), long StartIndex,
       long Count)
```

Parameters

Array	pointer to an array (array length ≥ *Count*)
StartIndex	number of the first variable to be transferred (1…80)
Count	number of variables to be transferred (max. 80)

Get_FPar_All

Get_FPar_All transfers all global float variables into an array.

```
void Get_FPar_All (float Array())
```

Parameters

Array	pointer to an array (array length ≥ 80)

Get_FPar_Block_Double

`Get_FPar_Block_Double` transfers the indicated amount of global float variables into a double array.

```
void Get_FPar_Block_Double (double Array(),
       long StartIndex, long Count)
```

Parameters

Array	pointer to an array (array length ≥ *Count*)
StartIndex	number of the first variable to be transferred (1…80)
Count	number of variables to be transferred (max. 80)

Notes

This method has been developed for applications, which process floating point data sets only with double precision.

Please consider that ***ADwin*** systems use single precision data. To avoid misunderstandings, you should display the array *Array()* only with single precision.

Get_FPar_All_Double

`Get_FPar_All_Double` transfers all global float variables into a double array.

```
void Get_FPar_All_Double (double Array())
```

Parameters

Array	pointer to an array (array length ≥ 80)

Notes

This method has been developed for applications, which process floating point data sets only with double precision.

Please consider that ***ADwin*** systems use single precision data. To avoid misunderstandings, you should display the array *Array()* only with single precision.

5.4 Transfer of Data Arrays

Instructions for data transfer between PC and ***ADwin*** system with global DATA arrays (DATA_1…DATA_200), which may be declared as FIFO arrays.

You must declare each array in ***ADbasic*** with the instruction DIM before usage (see "***ADbasic***" manual).

5.4.1 Simple data arrays

Data_Length

Data_Length returns the length of an array (i.e. number of elements), declared under ***ADbasic***.

```
long Data_Length (long DataNo)
```

Parameters

DataNo	number of the array (1…200)
return value	declared data length (=number of elements)

SetData_Long

SetData_Long transfers long data from the PC into a DATA array of the ***ADwin*** system.

```
void SetData_Long (long DataNo, long Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

GetData_Long

GetData_Long transfers long data from a DATA array of an ***ADwin*** system into an array.

```
void GetData_Long (long DataNo, long Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array (array length ≥ *Count*)
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

SetData_Float

`SetData_Float` transfers float data from the PC into a DATA array of the ***ADwin*** system.

```
void SetData_Float (long DataNo, float Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

GetData_Float

`GetData_Float` transfers float data from a DATA array of the ***ADwin*** system into an array.

```
void GetData_Float (long DataNo, float Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array (array length ≥ *Count*)
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

SetData_Double

`SetData_Double` transfers double data from the PC with single precision to a DATA array of the ***ADwin*** system.

```
void SetData_Double (long DataNo, double Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

Notes

This method has been developed for applications, which process floating point data sets only with double precision.

Please consider that ***ADwin*** systems use single precision data. To avoid misunderstandings, you should display the array *Data()* only with single precision.

GetData_Double

GetData_Double transfers float data from a DATA array of the ***ADwin*** system into a double array.

```
void GetData_Double (long DataNo, double Data(),
       long Startindex, long Count)
```

Parameters

DataNo	array number (1…200)
Data()	pointer to an array (array length ≥ *Count*)
StartIndex	number of the first variable to be transferred
Count	number of variables to be transferred

Notes

This method has been developed for applications, which process floating point data sets only with double precision.

Please consider that ***ADwin*** systems use single precision data. To avoid misunderstandings, you should display the array *Data()* only with single precision.

Data2File

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

```
void Data2File (char pFilename(), long DataNo,
       long Startindex, long Count, long Mode)
```

Parameters

Filename	pointer to the file name
DataNo	number of the DATA array (1…200)
Startindex	number of the first variable to be transferred
Count	number of elements to be transferred
Mode	0 : file will be overwritten (if file exists) 1 : data are appended to an already existing file

Notes

This DATA array must not be defined as FIFO. The data is saved in binary code.

Example (C)

```
Data2File("Test.dat", 1, 1, 1000, 0);
// Saves the elements 1...1000 from the ADbasic-
// array DATA_1 in the file Test.dat
```

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 FIFOs.

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

When working with FIFO arrays you should not write more elements into the FIFO than you have declared. You should definitely not read out more elements than you have written into the FIFO. To avoid malfunction check the FIFO status first:

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

– 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 `SetFifo_X`.

Fifo_Empty

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

```
long Fifo_Empty (long FifoNo)
```

Parameters

FifoNo	number of the FIFO array (1…200)
return value	number of free elements in the FIFO array

Fifo_Full

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

```
long Fifo_Full (long FifoNo)
```

Parameters

FifoNo	number of the FIFO array (1…200)
return value	number of used elements in the FIFO array

Fifo_Clear

`Fifo_Clear` initializes the write and read pointers of a FIFO array.

```
void Fifo_Clear (long FifoNo)
```

Parameters

FifoNo	number of the FIFO array (1…200)

Notes

While declaring a FIFO array (in ***ADbasic***) the FIFO pointers are not automatically initialized. Therefore call `Fifo_Clear` just at the beginning of your program, either in ***ADbasic*** or with this function.

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

SetFifo_Long

`SetFifo_Long` transfers long data from the PC to a FIFO array of the ***ADwin*** system.

```
void SetFifo_Long (long FifoNo, long Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFO array (1…200)
Data()	pointer to an array
Count	amount of elements to be transferred

GetFifo_Long

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

```
void GetFifo_Long (long FifoNo, long Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFO array (1…200)
Data()	pointer to an array (array length ≥ *Count*)
Count	amoung of elements to be transferred

SetFifo_Float

`SetFifo_Float` transfers float data from the PC into a FIFO array of the ***ADwin*** system.

```
void SetFifo_Float (long FifoNo, float Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFO array (1…200)
Data()	pointer to an array
Count	amount of the elements to be transferred

GetFifo_Float

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

```
void GetFifo_Float (long FifoNo, float Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFO array (1…200)
Data()	pointer to an array (array length ≥ *Count*)
Count	amount of elements to be transferred

SetFifo_Double

SetFifo_Double transfers double data from the PC with single precision to a FIFO array of the ***ADwin*** system.

```
void SetFifo_Double (long FifoNo, double Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFO array (1…200)
Data()	pointer to an array
Count	amount of the elements to be transferred

Notes

This method has been developed for applications, which process floating point data sets only with double precision.

Please consider that ***ADwin*** systems use single precision data. To avoid misunderstandings, you should display the array *Data()* only with single precision.

GetFifo_Double

GetFifo_Double transfers float FIFO data from the ***ADwin*** system into a double array.

```
void GetFifo_Double (long FifoNo, double Data(),
       long Count)
```

Parameters

FifoNo	number of the FIFo array (1…200)
Data()	pointer to the array (array length ≥ *Count*)
Count	amount of elements to be transferred

Notes

This method has been developed for applications, which process floating point data sets only with double precision (double precision arrays).

Please consider that the data have single precision. Therefore you should display data from the array *Data()* only with single precision, in order to avoid misunderstandings regarding the precision.

5.4.3 Data arrays with string data

Instructions for the data transfer between PC and ***ADwin*** system with global DATA arrays (DATA_1...DATA_200), which contain string data. The DATA array must be declared with DIM ... AS STRING in ***ADbasic***.

SetData_String

SetData_String transfers a string into a DATA array.

```
void SetData_String (long DataNo, char pData())
```

Parameters

DataNo	number of the DATA array (1...200)
Data	pointer to the string to be transferred

Notes

Each transferred string is automatically NULL-terminated(i.e. ASCII code 0 is appended).

Please consider the different usage of the development environments with this end identifier, when it is part of the string.

For instance, ***ADbasic*** and C transfer the string "Hello\0 world" only up to the end identifier, i.e. "Hello". Delphi (Kylix) however, transfers the complete string.

Example (C)

```
SetData_String (2, "this is a text" );
// the string "this is a text" is written into the array
// DATA_2 and the end identifier is added.
```

GetData_String

GetData_String transfers a string from a DATA array to a buffer.

```
long GetData_String (long DataNo, char pData(),
       long MaxCount)
```

Parameters

DataNo	number of the DATA array (1…200)
pData	pointer to the array, into which the string is to be written (array length ≥ *MaxCount*+1)
MaxCount	maximum amount of characters to be transferred
return value	amount of the read characters (without end identifier)

Notes

After writing the string into the array *pData* an additional end identifier is added as last string (ASCII number 0), so that the string contains *MaxCount*+1 elements.

If the string which is to be transferred has an end identifier, then the transfer stops exactly there. The number of the characters read up to this point without end identifier is the return value.

Example (C)

```
char ArrayString[100];
GetData_String (2, ArrayString, 100);
// get a string with 100 elements from DATA_2
```

If the DATA array in the ***ADwin*** system has an end identifier for instance at position 9, then 8 characters are read.

String_Length

String_Length returns the length of a data string in a DATA array.

```
long String_Length(long DataNo)
```

Parameters

DataNo	number of the DATA array (1…200)
return value	length of the string

Notes

String_Length counts the characters in a DATA array until the first end identifier occurs (ASCII number 0). The end identifier is not counted as a character.

Example (C)

```
erg = String_Length (2);
// determines the length of the string in DATA_2.
```

5.5 Error Handling

Get_Last_Error

`Get_Last_Error` returns the error number, which refers to the last instruction (of the program) sent to the ***ADwin*** system.

```
long Get_Last_Error()
```

Parameters

return value	≠0 : number of error, which occured last (see Annex A.1). 0 : no error occured

Notes

The returned error number always refers to

- the last instruction which was sent to the ***ADwin*** system (most but not all instructions are members of this group)
- the program, which sent that instruction

After the function is called, the error number is reset to 0 (zero).

Example (C)

```
long Error;
Error = Get_Last_Error();
```

Get_Last_Error_Text

`Get_Last_Error_Text` returns an error text related to an error number.

```
char Get_Last_Error_Text (long Errno)
```

Parameters

Errno	error number
return value	pointer to the error text

Notes

This function can be called in combination with the function `Get_Last_Error`.

Beispiel (C)

```
long Error = Get_Last_Error();
printf(„Get_Last_Error:%d\n“, Error);
printf(„Last_Error_Text:%s\n“,
 Get_Last_Error_Text(Error));
```

Annex

A.1 Error messages

No.	Error message
0	No Error.
1	Timeout error on writing to the ADwin-system.
2	Timeout error on reading from the ADwin-system.
3	Timeout error on fast writing to the ADwin-system.
4	Timeout error on fast 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 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.
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.
204	The file is not a BTL.
1000	Network error (RPC).
1001	Network error (Bind).
1002	Network error (String bind).
1003	Wrong password (RPC).
2000	Network error (TcpIp).
2001	Network timeout.
2002	Wrong password.
3001	Device is unknown.

A.2 Index of methods and properties