INTRODUCTION

This interface board has sixteen digital input/output channels which can be used as either input or output as desired. In addition, there are eight analogue outputs with 6 bit resolution, one analogue output with 8 bit resolution, and four analogue inputs of 8 bit resolution. If more digital outputs are required, then the analogue outputs can be used by setting to minimum or maximum output voltage. If there are too few inputs, then the analogue inputs may be used. In this way it is very simple to track the state of a rotating switch by connecting in a different voltage for every state. Please note that these extra input and output channels are not optically isolated.
The number of inputs/outputs can be further expanded by connecting more cards together, up to a maximum of four. Each card is given its own identification by means of a two pole DIP-switch SW1 (see table below for channel numbering).

To connect the card to the computer the printer port is used (note that there is no need to install an extra printer port as the printer can be connected to the interface card). Three lines from this port are used, namely, "Select" (pin 13), "Autofeed" (pin 14) and "Select in" (pin 17). Communication between the computer and interface card is by means of a serial link. One line (Select in) serves as the clock signal, the second (Autofeed) as the data output, and the third (Select) as the data input.

All communication routines are contained in a Dynamic Link Library (DLL) K8D.DLL. This document describes all the DLL functions and procedures that are available for your application programme. Using the DLL allows you to write custom Windows 9x, NT or XP based applications in Visual C++, Delphi, Visual Basic, or any other 32-bit Windows application development tool that supports calls to a DLL. Thus, there is no need to be concerned about the communication protocol.

A complete overview of all procedures, functions and variables that are used by the "K8D.DLL" follows. The example program can also be carefully studied in order to gain an insight as to how to construct your customised application programmes.
Note that all examples are written for Delphi.

At the end of this document there are full declarations for the DLL function and procedures for Delphi and Visual Basic.

TABLE 'SW1' SETTINGS

CARD NUMBER	CHIP NUMBER	CHANNEL NUMBERS
0 (OFF-OFF)	IO-chip no: 0	IO-channels: 1...8
	IO-chip no: 1	IO-channels: 9...16
	DAC-chip no: 0	DAC-channels: 1...8
	AD-chip no: 0	AD-channels: 1...4
		DA-channel: 1
1 (OFF-ON)	IO-chip no: 2	IO-channels: 17...24
	IO-chip no: 3	IO-channels: 25...32
	DAC-chip no: 1	DAC-channels: 9...16
	AD-chip no: 1	AD-channels: 5...8
		DA-channel: 2
2 (0N-OFF)	IO-chip no: 4	IO-channels: 33...40
	IO-chip no: 5	IO-channels: 41...48
	DAC-chip no: 2	DAC-channels: 17...24
	AD-chip no: 2	AD-channels: 9...12
		DA-channel: 3
3 (ON-ON)	IO-chip no: 6	IO-channels: 49...56
	IO-chip no: 7	IO-channels: 57...64
	DAC-chip no: 3	DAC-channels: 25...32
	AD-chip no: 3	AD-channels: 13...16
		DA-channel: 4

OVERVIEW OF THE 'K8D.DLL' VARIABLES

The application program has access to the the following variables via the k8d.dll:

VARIABLE	TYPE	INITIAL VALUE	DESCRIPTION
DA	Array[1..4] of Integer	0	Contains the data (value between 0 and 255) of the four 8-bit Digital to Analogue converters
DAC	Array[1..32] of Integer	0	Contains the data (value between 0 and 63) of the thirty two 6-bit Digital to Analogue Converter channels
IOconfig	Array[0..7] of Integer	$0FF	Each bit contains the configuration of the corresponding channels of the eight IO-ports. Bit high (1) = input; bit low (0) = output
IOdata	Array[0..7] of Integer		Each bit contains the status of the respective channel of the eight IO-ports. Bit high (1) = channel on; bit low (0) = channel off

The global constants used in the following descriptions:

CONSTANT	VALUE	DESCRIPTION
MaxIOcard	3	Highest possible address of the interface card
MaxIOchip	7	Highest possible Input/Output chip number
MaxIOchannel	64	Highest possible Input/Output channel number
MaxDACchannel	32	Highest possible 6-bit Digital to Analogue Converter channel number
MaxADchannel	16	Highest possible Analogue to Digital channel number
MaxDAchannel	4	Highest possible 8-bit Digital to Analogue channel number

OVERVIEW OF 'K8D.DLL' PROCEDURES AND FUNCTIONS

8-bit Analogue to Digital converter procedures

`ReadADchannel(Channelno)`	*Reads the status of the analogue input-channel*

8-bit Digital to Analogue conversion procedures

`OutputDAchannel(Channelno,Data)`	*Sets the analogue output channel according to the data*
`ClearDAchannel(Channelno)`	*Sets the analogue output channel to minimum*
`ClearAllDA`	*Sets all analogue output channels to minimum*
`SetDAchannel(Channelno)`	*Sets the analogue output channel to maximum*
`SetAllDA`	*Sets all analogue output channels to maximum*

6-bit Digital to Analogue Conversion procedures

`OutputDACchannel(Channelno,Data)`	*Sets the analogue output channel according to the data*
`ClearDACchannel(Channelno)`	*Sets the analogue output channel to minimum*
`ClearDACchip(Chipno)`	*Sets the 8 analogue output channels of the DAC-chip to minimum*
`ClearAllDAC`	*Sets all analogue output channels to minimum*
`SetDACchannel(Channelno)`	*Sets the analogue output channel to maximum*
`SetDACchip(Chipno)`	*Sets the 8 analogue output channels of the DAC-chip to maximum*
`SetAllDAC`	*Sets all analogue output channels to maximum*

IO configuration procedures

`ConfigAllIOasInput`	*Configures all IO-channels as inputs*
`ConfigIOchipAsInput(Chipno)`	*Configures all IO-channels of the IO-chip as inputs*
`ConfigIOchannelAsInput(Channelno)`	*Configures the IO-channel as input*
`ConfigAllIOasOutput`	*Configures all IO-channels as outputs*
`ConfigIOchipAsOutput(Chipno)`	*Configures all IO-channels of the IO-chip as outputs*
`ConfigIOchannelAsOutput(Channel)`	*Configures the IO-channel as output*

Setting of IOdata & IO variables (the physical status of the IO-channels does not change)

`UpdateIOdataArray(Chipno,Data)`	*Sets the output status according to the data (inputs do not change)*
`ClearIOchArray(Channelno)`	*Clears the output status of the selected channel (set low)*
`ClearIOdataArray(Chipno)`	*Clears the output status of the channels of the IO-chip (set low)*
`SetIOchArray(Channelno)`	*Sets the output status of the selected channel (set high)*
`SetIOdataArray(Chipno)`	*Sets the output status of the channels of the IO-chip (set high)*

Output procedures

`IOoutput(Chipno,Data)`	*Sets the outputs of the IO-chip according to the data (inputs do not change)*
`UpdateIOchip(Chipno)`	*Sets the outputs of the IO-chip according to the status of the 'IOdata' variable*
`UpdateAllIO`	*Sets all outputs according to the status of the 'IOdata' variables*
`ClearIOchannel(Channelno)`	*Clears the output channel*
`ClearIOchip(Chipno)`	*Clears the output channels of the IO-chip*
`ClearAllIO`	*Clears all output channels*
`SetIOchannel(Channelno)`	*Sets the output channel*
`SetIOchip(Chipno)`	*Sets the output channels of the IO-chip*
`SetAllIO`	*Sets all output channels*

Input procedures and functions

`ReadIOchannel(Channelno)`	*Reads the status of the input channel*
`ReadIOchip(Chipno)`	*Reads the status of the input channels of the IO-chip*
`ReadIOconficArray(Buffer)`	*Reads the IO configuration data from the DLL to the application program*
`ReadIOdataArray(Buffer)`	*Reads the IO status data from the DLL to the application program*
`ReadDACarray(Buffer)`	*Reads the DAC data from the DLL to the application program*
`ReadDAarray(Buffer)`	*Reads the DA data from the DLL to the application program*

General procedures

`SelectI2CprinterPort(Printer_no)`	*Chooses the communication port*
`Start_K8000`	*Opens a link to the K8000 device*
`Stop_K8000`	*Closes the link to the K8000 device*

'K8D.DLL' PROCEDURES AND FUNCTIONS

Start_K8000

Syntax
```
PROCEDURE Start_K8000;
```

Description
Initializes the communication routines for the K8000 card. Loads the drivers needed to communicate via the LPT port. This procedure must be performed in the beginning of the application program.

Example
```
  BEGIN
     Start_K8000;
  END;
```

Sop_K8000

Syntax
```
PROCEDURE Stop_K8000;
```

Description
Unloads the communication routines for K8000 card and unloads the drivers needed to communicate via the PLT port. This is the last action of the application program before termination.

Example
```
  BEGIN
     Stop_K8000;
  END;
```

SelectI2CprinterPort

Syntax
```
PROCEDURE SelectI2CprinterPort(Printer_no: Longint);
```

Parameter
`Printer_no:` Value between 0 and 2 given by the printer port to which the interface card is connected.
0: printer port address is 0BC (hex)
1: printer port address is 378 (hex), usually the address of LPT1
2: printer port address is 278 (hex), usually the address of LPT2

Result
Communication between the PC and the K8000 card will occur via the selected LPT port address.

Description
The printer port to be used must be specified at the start of your programme in order to run the interface card so that the communication routines are carried out correctly. The default designation is LPT1, but this can be changed using this procedure.

Example
```
  BEGIN
    SelectI2CprinterPort(1);
    // LPT1 address on mainboard is set to 378
```

```
END;
```

ReadADchannel AD

Syntax
```
FUNCTION ReadADchannel(Channel_no: Longint):Longint;
```

Parameter
`Channel_no:` Value between 1 and 16 which corresponds to the AD channel whose status is to be read.

Result
`AD:`
The corresponding 'AD' data is read according to the status of the AD input.

Description
The input voltage of the selected 8-bit Analogue to Digital converter channel is converted to a value which lies between 0 and 255 and registered in the respective 'AD' data variable.

Example

```
var data: longint;
BEGIN
    data := ReadADchannel(1);
    // AD channel 1 is read to variable 'data'
END;
```

OutputDAchannel DA

Syntax
```
PROCEDURE OutputDAchannel(Channel_no: Longint; Data: Longint);
```

Parameter
`Channel_no:` Value between 1 and 4 which corresponds to the 8-bit DA channel number whose data is to be changed.
`Data:` Value between 0 and 255 which is to be sent to the 8-bit Digital Analogue Converter .

Result
`DA:`
The 'DA' data variable of the selected channel is set according to the data which is to be sent. The selected DA-channel is changed.

Description
The indicated 8-bit Digital to Analogue Converter channel is altered according to the new data. This means that the data corresponds to a specific voltage. The value 0 corresponds to a minimum output voltage (0 Volt) and the value 255 corresponds to a maximum output voltage (Vmax) which is set according to the preset on the interface board. A value of 'Data' lying in between these extremes can be translated by the following formula : Data x Vmax/255.

Example

```
BEGIN
  OutputDAchannel(1,127);
  // DA channel 1 is at 1/2 Vmax
END;
```

ClearDAchannel DA

Syntax
`PROCEDURE ClearDAchannel(Channel_no: Longint);`

Parameter
`Channel_no:` Value between 1 and 4 which corresponds to the 8-bit DA channel number in which the data is to be erased.

Result
`DA:`
The 'DA' data variable of the selected DA-channel is set to minimum (0) . The selected DA-channel is set to minimum output voltage (0 Volt).

Description
The selected 8-bit Digital to Analogue Converter channel is set to minimum output voltage (0 Volt).

Example

```
BEGIN
  ClearDAchannel(1);
  // DA channel 1 is at Vmin
END;
```

ClearAllDA DA

Syntax
`PROCEDURE ClearAllDA;`

Result
All DA-channels are set to minimum output voltage (0 Volt) .

Description
All DA-channels of the 8-bit Digital to Analogue Converters are set to minimum output voltage (0 Volt) .

Example

```
BEGIN
  ClearAllDA;
  // All DA channels 1...4 are at Vmin
END;
```

SetDAchannel DA

Syntax
```
PROCEDURE SetDAchannel(Channel_no: TDAchannel);
```

Parameter
`Channel_no:` Value between 1 and 4 which corresponds to the 8-bit DA channel number in which the data is to be set to maximum.

Result
The selected DA-channel is set to maximum output voltage.

Description
The selected 8-bit Digital to Analogue Converter channel is set to maximum output voltage.

Example 15
```
PROGRAM Set_DA_channel;
USES I2C, WinCrt;
BEGIN
  SetDAchannel(1);
  Writeln('Set DA channel 1 at Vmax');
END;
```

SetAllDA DA

Syntax
```
PROCEDURE SetAllDA;
```

Result
All DA-channels are set to maximum output voltage.

Description
All DA-channels of the 8-bit Digital to Analogue Converters are set to maximum output voltage.

Example
```
BEGIN
  SetAllDA;
  // All DA channels 1...4 are at Vmax
END;
```

OutputDACchannel DAC

Syntax
```
PROCEDURE OutputDACchannel(Channel_no: Longint; Data: Longint);
```

Parameters
`Channel_no:` Value between 1 and 32 which corresponds to the 6-bit DAC channel number in which the data is to be changed.
`Data:` Value between 0 and 63 which is to be sent to the 6-bit Digital Analogue Converter .

Result

The selected DAC-channel is updated.

Description
The indicated 6-bit Digital to Analogue Converter channel is modified with the new data. This means that the data corresponds to a specific voltage. The value 0 corresponds to a minimum output voltage (Vmin) and the value 63 corresponds to a maximum output voltage (Vmax) which can be set using the preset on the interface board. A value of 'Data' lying in between these two limits corresponds to a voltage according to the formula:
Vmin + Data x (Vmax - Vmin)/63.

Example
```
BEGIN
  OutputDACchannel(1,21);
  // DAC channel 1 is at Vmin + 1/3(Vmax-Vmin)
END;
```

ClearDACchannel DAC

Syntax
```
PROCEDURE ClearDACchannel(Channel_no: Longint);
```

Parameter
`Channel_no:` Value between 1 and 32 which corresponds to the 6-bit DAC channel number in which the data is to be erased.

Result
The selected DAC-channel is set to minimum output voltage (Vmin).

Description
The selected 6-bit Digital to Analogue Converter channel is set to minimum output voltage (Vmin).

Example
```
BEGIN
  ClearDACchannel(2);
  // DAC channel 2 is at Vmin
END;
```

ClearDACchip DAC

Syntax
```
PROCEDURE ClearDACchip(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 3 which corresponds to the address of the 6-bit DAC chip in which the 8 channels are to be set to minimum output voltage.

Result
The 8 DAC-channels of the indicated DAC-chip are set to minimum output voltage.

Description

The eight DAC-channels of the selected 6-bit Digital to Analogue Converter chip are set to minimum (Vmin).

Example
```
BEGIN
  ClearDACchip(0);
  // DAC channels 1...8 are at Vmin
END;
```

ClearAllDAC DAC

Syntax
```
PROCEDURE ClearAllDAC;
```

Result
All DAC-channels are set to minimum output voltage.

Description
All DAC-channels of the 6-bit Digital to Analogue Converters are set to minimum output voltage (Vmin) .

Example
```
BEGIN
  ClearAllDAC;
  // All DAC channels 1...32 are at Vmin
END;
```

SetDACchannel DAC

Syntax
```
PROCEDURE SetDACchannel(Channel_no: Longint);
```

Parameter
`Channel_no:` Value between 1 and 32 which corresponds to the 6-bit DAC channel number in which the data is to be set to maximum.

Result
The selected DAC-channel is set to maximum output voltage.

Description
The selected 6-bit Digital to Analogue Converter channel is set to maximum output voltage.

Example
```
BEGIN
  SetDACchannel(3);
  // Set DAC channel 3 at Vmax
END;
```

SetDACchip DAC

Syntax
```
PROCEDURE SetDACchip(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 3 which corresponds to the address of the 6-bit DAC chip in which the 8 channels are to be set to maximum output voltage.

Result
The 8 DAC-channels of the indicated DAC-chip are set to maximum output voltage.

Description
The eight DAC-channels of the selected 6-bit Digital to Analogue Converter chip are set to the maximum output voltage (Vmax). The 'DAC' data variables of the respective DAC-channels are likewise adjusted.

Example
```
  BEGIN
    SetDACchip(0);
    // DAC channels 1...8 are at Vmax
  END;
```

SetAllDAC DAC

Syntax
```
PROCEDURE SetAllDAC;
```

Result
All DAC-channels are set to maximum output voltage.

Description
All DAC-channels of the 6-bit Digital to Analogue Converters are set to maximum output voltage (Vmax).

Example
```
  BEGIN
    SetAllDAC;
    // All DAC channels 1...32 are at Vmax
  END;
```

ConfigAllIOasInput IO

Syntax
```
PROCEDURE ConfigAllIOasInput;
```

Result
`IOconfig:` The 'IOconfig' variables for all Input/Output ports are given the value 255.

Description

All digital Input/Output channels (1...64) are configured as inputs. Each IO chip (0...7) contains a variable in which the configuration of each IO pin is returned in the form of a data bit which is not necessarily high. If this bit is high (1) it means that this is an input. Writing to this IO-channel will have no effect. The status of the IO-channels can only be determined by an external signal.

Example

```
BEGIN
  ConfigAllIOasInput;
  // All IO channels are now configured as inputs
END;
```

ConfigIOchipAsInput IO

Syntax

```
PROCEDURE ConfigIOchipAsInput(Chip_no: Longint);
```

Parameter

`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which all channels are configured as inputs.

Result

`IOconfig:` The 'IOconfig variable of the respective Input/Output chip is given the value 255.

Description

The 8 digital Input/Output channels of the selected IO chip are configured as inputs. Each IO chip (0...7) contains a variable in which the configuration of each IO pin is returned in the form of a data bit which is not necessarily high. If this bit is high (1) it means that it is an input. The status of the IO-channels can only be determined by an external signal.

Example

```
BEGIN
  ConfigIOchipAsInput(0);
  // The 8 channels from  IO Chip 0 are now configured as inputs
END;
```

ConfigIOchannelAsInput IO

Syntax

```
PROCEDURE ConfigIOchannelAsInput(Channel_no: Longint);
```

Parameter

`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be configured as an input.

Result

`IOconfig:` Via the channel number, the IO-chip number and bit number are determined in order to set this bit high (1) in the 'IOconfig' variable.

Description

The selected Input/Output channel is configured as an input while the configuration of the other channels remain unchanged. This occurs by making the correct bit high (1) in the configuration

variable of the respective IO chip. The status of the IO-channel can only be determined by an external signal that is fed to it.

Example

```
BEGIN
  ConfigIOchannelAsInput(1);
  // IO channel 1 is now configured as input');
END;
```

ConfigAllIOasOutput IO

Syntax

```
PROCEDURE ConfigAllIOasOutput;
```

Result

`IOconfig:` The 'IOconfig' variables for all Input/Output ports are set low (0).

Description

All digital Input/Output channels (1...64) are configured as outputs. Each IO chip (0...7) contains a variable in which the configuration of each IO pin is returned in the form of a data bit which is not necessarily high. If this bit is low (0) it means an output. The status of the IO-channels are determined by the value written to them.

Example

```
BEGIN
  ConfigAllIOasOutput;
  // All IO channels are now configured as outputs
END;
```

ConfigIOchipAsOutput IO

Syntax

```
PROCEDURE ConfigIOchipAsOutput(Chip_no: Longint);
```

Parameter

`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which all channels are to be configured as outputs.

Result

`IOconfig:` The 'IOconfig variable of the respective Input/Output chip is set low (0).

Description

The 8 digital Input/Output channels of the selected IO chip are configured as outputs. Each IO chip (0...7) contains a variable in which the configuration of each IO pin is returned in the form of a data bit which is not necessarily high. If this bit is low (0) then it means an output. The status of these IO-channels are determined by the value written to them.

Example

```
BEGIN
  ConfigIOchipAsOutput(1);
  // The 8 channels from IO Chip 1 are now configured as outputs
END;
```

ConfigIOchannelAsOutput IO

Syntax
```
PROCEDURE ConfigIOchannelAsOutput(Channel_no: Longint);
```

Parameter
`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be configured as an output .

Result
`IOconfig:` Via the channel number, the IO-chip number and bit number are determined in order to set the 'IOconfig' variable low (0).

Description
The selected Input/Output channel is configured as an output while the configuration of the other channels remain unchanged. This occurs by making the correct bit low (0) in the configuration variable of the respective IO chip. The status of this IO-channel is determined by the value written to it.

Example
```
BEGIN
  ConfigIOchannelAsOutput(2);
  // IO channel 2 is now configured as output
END;
```

UpdateIOdataArray IO

Syntax
```
PROCEDURE UpdateIOdataArray(Chip_no: Longint; Data: Longint);
```

Parameters
`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which the data is to be changed.
`Data:` Value between 0 and 255 which is to be sent to the IO port (8 channels).

Result
`IOdata:` The 'IOdata' variable of the selected chip is updated with the new data such that the status of the input channels remains unchanged.

Description
The status of the Input/Output ports are held in the 'IO data' variable. Each bit of this variable corresponds to the status of an IO-channel. If the status of the 8 channels of an IO-chip is to be changed then it is not advisable to immediately set the data in the 'IOdata' variable, because the status of the input channels would no longer correspond to the physical status. In order to prevent such conflict this procedure must be used.
Note that the 'IOdata' variable is just a buffer memory. When its value is changed the outputs do not immediately change status. Status only changes when its value is sent to the IO-chip.

Example
```
BEGIN
```

```
    ConfigIOchipAsOutput(0);
    UpdateIOdataArray(0,204);
    // 204 is in binary format 11001100
    // this sets channels 3,4,7,8  and clears channels 1,2,5,6
  END;
```

ClearIOchArray IO

Syntax
```
PROCEDURE ClearIOchArray(Channel_no: Longint);
```

Parameter
`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be cleared.

Result
`IOdata:` If selected channel is an output then the correct bit in the 'IOdata' variable of the corresponding chip is set low (0).

Description
The status of the Input/Output ports is held in the 'IO data' variable. Each bit of this variable corresponds to the status of an IO-channel. If a particular output of the IO-chip is to be cleared then it is not advisable to immediately set the corresponding bit in the 'IOdata' variable to low. If the selected channel has been configured as an input then its status would no longer correspond to the physical status. In order to avoid such conflict this procedure must be used.
Note that the 'IOdata' variable is just a buffer memory. When its value is changed the outputs do not immediately change status. This only occurs when this value is sent to the IO-chip.

Example
```
  BEGIN
    ConfigIOchipAsOutput(0);
    ConfigIOchannelAsInput(1);     {channel 1 configured as input}
    ConfigIOchannelAsInput(2);     {channel 2 configured as input}
    ConfigIOchannelAsInput(3);     {channel 3 configured as input}
    ConfigIOchannelAsInput(4);     {channel 4 configured as input}

    ClearIOchArray(2); // Clear channel 2 from IOdata array
    // The status of the inputs stays unchanged

    ClearIOchArray(8); // Clear channel 8 from IOdata array
  END;
```

ClearIOdataArray IO

Syntax
```
PROCEDURE ClearIOdataArray(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which the outputs are to be cleared.

Result
`IOdata:` The bits of the 'IOdata' variable of the selected chip are set low (0) but the bits that correspond to inputs remain unchanged.

Description
The status of the Input/Output ports are held in the 'IOdata' variable. Each bit of this variable corresponds to the status of an IO-channel. If the outputs of the IO-chip are to be cleared then it is not recommended to immediately set the 'IOdata' variable to 0, because the status of the input channels would no longer correspond to the physical status. In order to avoid such conflict this procedure must be used.
Note that the 'IOdata' variable is just a buffer memory. When this changes its value the outputs do not immediately change status. This occurs only when this value is sent to the IO-chip.

Example
```
BEGIN
  ConfigIOchipAsOutput(0);
  ConfigIOchannelAsInput(1);     {channel 1 configured as input}
  ConfigIOchannelAsInput(2);     {channel 2 configured as input}
  ConfigIOchannelAsInput(3);     {channel 3 configured as input}
  ConfigIOchannelAsInput(4);     {channel 4 configured as input}
  ClearIOdataArray(0); // Clear channels 1 to 8 from IOdata array
  // The status of the inputs stays unchanged
END;
```

SetIOchArray IO

Syntax
```
PROCEDURE SetIOchArray(Channel_no: Longint);
```

Parameter
`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be set.

Result
`IOdata:` If the selected channel is an output then the correct bit in the 'IOdata' variable of the corresponding chip is set high (1).

Description
The status of the Input/Output ports is held in the 'IOdata' variable. Each bit of this variable corresponds to the status of an IO-channel. If a particular output of the IO-chip is to be set then it is not advisable to immediately set the corresponding bit in the 'IOdata' variable to high. If the selected channel has been configured as an input then the status would no longer correspond to the physical status. In order to prevent such conflict this procedure must be used.
Note that the 'IOdata' variable is just a buffer memory. When it changes its value the outputs do not immediately change status. This only occurs when this value is sent to the IO-chip.

Example
```
BEGIN
  ConfigIOchipAsOutput(0);
  ConfigIOchannelAsInput(1);     {channel 1 configured as input}
  ConfigIOchannelAsInput(2);     {channel 2 configured as input}
  ConfigIOchannelAsInput(3);     {channel 3 configured as input}
  ConfigIOchannelAsInput(4);     {channel 4 configured as input}
  SetIOchArray(1); // Set channels 1 from IOdata array
  SetIOchArray(5); // Set channels 5 from IOdata array
  // The status of the input channel 1 stays unchanged
```

```
  END;
```

SetIOdataArray IO

Syntax
```
PROCEDURE SetIOdataArray(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 7 that corresponds to the address setting of the Input/Output chip in which the outputs are to be set.

Result
`IOdata:` The bits of the 'IOdata' variable of the selected chip are set high (1) but the bits that correspond to inputs remain unchanged.

Description
The status of the Input/Output ports are held in the 'IOdata' variable. Each bit of this variable corresponds to the status of an IO-channel. If the outputs of the IO-chip are to be set then it is not advisable to immediately to load 255 in the 'IOdata' variable, because the status of the input channels would no longer correspond to the physical status. In order to avoid such conflict this procedure must be used.
Note that the 'IOdata' variable is just a buffer memory. When it changes its value the outputs do not immediately change status. This only happens when this value is sent to the IO-chip.
The 'IO' variables of the respective IO-chip are likewise adjusted.

Example
```
  BEGIN
    ConfigIOchipAsOutput(0);
    ConfigIOchannelAsInput(1);     {channel 1 configured as input}
    ConfigIOchannelAsInput(2);     {channel 2 configured as input}
    ConfigIOchannelAsInput(3);     {channel 3 configured as input}
    ConfigIOchannelAsInput(4);     {channel 4 configured as input}
    SetIOdataArray(0); // Set channels 1 to 8 from IOdata array
    // The status of the inputs stays unchanged
  END;
```

IOoutput IO

Syntax
```
PROCEDURE IOoutput(Chip_no: Longint; Data: Longint);
```

Parameters
`Chip_no:` Value between 0 and 7 that corresponds to the address setting of the Input/Output chip in which the data is to be changed.
`Data:` Value between 0 and 255 that is sent to the IO port (8 channels).

Result
`IOdata:` The 'IOdata' variable of the selected chip is updated with the new data such that the status of the input channels remains unchanged.
The outputs of the selected IO-chip are updated.

Description
The channels of the selected IO-chip which have been configured as outputs are updated with the status of the corresponding bits in the data parameter. A high (1) level means that the output is set, and a low (0) level means that the output is cleared. The status of the inputs remains unchanged.
The 'IOdata' and 'IO' variables of the respective IO-chip are likewise updated.

Example
```
  BEGIN
    ConfigIOchipAsOutput(1);
    IOoutput(1,128);
    // Output channels 9..15 are off, Output channel 16 is on
  END;
```

UpdateIOchip IO

Syntax
```
PROCEDURE UpdateIOchip(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 7 that corresponds to the address setting of the Input/Output chip in which the outputs are to be changed.

Result
All outputs of the selected Input/Output chip are changed according to the status of the corresponding bits in the 'IOdata' variables.

Description
All channels of the selected IO-chip that have been configured as outputs are changed according to the status of the corresponding bits in the 'IOdata' array. A high (1) level means that the output is set, a low (0) level means that the output is cleared. The status of the inputs remain unchanged.

Example
```
  BEGIN
    ConfigIOchipAsOutput(1);
    UpdateIOdataArray(1,64); // channels 9...14 & 16 off, 15 on
    UpdateIOchip(1);
    // Output channels 9..14 & 16 are off, Output channel 15 is on
  END;
```

UpdateAllIO IO

Syntax
```
PROCEDURE UpdateAllIO;
```

Result
All outputs are changed according to the status of the corresponding bits in the 'IOdata' variables.

Description

All channels of the IO-chips which have been configured as outputs are changed according to the status of the corresponding bits in the 'IOdata' array. A high (1) level means that the output is set, a low (0) level means that the output is cleared. The status of the inputs remain unchanged.

Example

```
BEGIN
  ConfigAllIOasOutput;
  UpdateIOdataArray(0,1);      // channel  1 on
  UpdateIOdataArray(1,128);    // channel 16 on
  UpdateAllIO;
  // Output channel 1 & 16 are on, Output channels 2..15 are off
END;
```

ClearIOchannel IO

Syntax

```
PROCEDURE ClearIOchannel(Channel_no: Longint);
```

Parameter

`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be cleared.

Result

`IOdata:` If the selected channel is an output then the correct bit in the 'IOdata' variable of the corresponding chip is set low (0).

Description

If the selected channel has been configured as an output then it is cleared. The status of the inputs remain unchanged. The 'IOdata' and 'IO' variables are adjusted according to the new status.

Example

```
BEGIN
  ConfigIOchannelAsOutput(9);
  ClearIOchannel(9); // Output channel 9 is off
END;
```

ClearIOchip IO

Syntax

```
PROCEDURE ClearIOchip(Chip_no: Longint);
```

Parameter

`Chip_no:` Value between 0 and 7 that corresponds to the address setting of the Input/Output chip in which the outputs are to be cleared.

Result

`IOdata:` The bits in the 'IOdata' variable of the selected IO-chip that correspond to outputs are set low (0) , those of the inputs remain unchanged.

Description

All channels of the selected Input/Output chip which have been configured as outputs are cleared. The status of the inputs remain unchanged. The 'IOdata' and 'IO' variables are changed according to the new status.

Example

```
BEGIN
  ConfigIOchipAsOutput(1);
  ClearIOChip(1); // Output channels 9...16 are off
END;
```

ClearAllIO IO

Syntax

```
PROCEDURE ClearAllIO;
```

Result

`IOdata:` The bits in the 'IOdata' variables that correspond to outputs are set low (0), those of the inputs remain unchanged.
All outputs are cleared.

Description

All channels of the IO-chips that have been configured as outputs are cleared. The status of the inputs remain unchanged. The 'IOdata' is changed according to the new status.

Example

```
BEGIN
  ConfigAllIOasOutput;
  ConfigIOchannelAsInput(2);   // channel 2 configured as input
  ClearAllIO;
  // All Output channels are off
  // Input channel 2 unchanged
END;
```

SetIOchannel IO

Syntax

```
PROCEDURE SetIOchannel(Channel_no: Longint);
```

Parameter

`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel that is to be set.

Result

`IOdata:` If the selected channel is an output then the the correct bit in the 'IOdata' variable of the corresponding chip is set high (1).
The selected output channel is set.

Description

If the selected channel has been configured as an output then it is set. The status of the inputs remain unchanged. The 'IOdata' is changed according to the new status.

Example

```
BEGIN
  ConfigIOchannelAsOutput(9);
  SetIOchannel(9); // Output channel 9 is on
END;
```

SetIOchip IO

Syntax
```
PROCEDURE SetIOchip(Chip_no: Longint);
```

Parameter
`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which the outputs are to be set.

Result
`IOdata:` The bits in the 'IOdata' variable of the selected IO-chip that correspond to outputs are set high (1), those of the inputs remain unchanged.

Description
All channels of the selected Input/Output chip that have been configured as outputs are set. The status of the inputs remain unchanged.

Example
```
BEGIN
  ConfigIOchipAsOutput(1);
  SetIOchip(1); // Output channels 9...16 are on
END;
```

SetAllIO IO

Syntax
```
PROCEDURE SetAllIO;
```

Result
`IOdata:` The bits in the 'IOdata' variables that correspond to outputs are set high (1), those of the inputs remain unchanged.
`IO:`
The 'IO' variables that correspond to outputs are set high (True). Those of the input channels remain unchanged.
All outputs are set.

Description
All channels of the IO-chips that have been configured as outputs are set. The status of the inputs remain unchanged. The 'IOdata' variables are changed according to the new status.

Example
```
BEGIN
  ConfigAllIOasOutput;
  ConfigIOchannelAsInput(2);  // channel 2 configured as input
  SetAllIO;
  // All Output channels are on
  // Input channel 2 unchanged
```

```
END;
```

ReadIOchannel IO

Syntax
```
FUNCTION ReadIOchannel(Channel_no: Longint): Boolean;
```

Parameter
`Channel_no:` Value between 1 and 64 which corresponds to the Input/Output channel whose status is to be read.

Result
`IOdata:` The bit in the 'IOdata' variable of the selected channel is changed according to the status of that channel. A high (1) means that the channel has been set, a low (0) means that it has been cleared.

Description
The status of the selected Input/Output channel is read and registered in the 'IOdata' variable. The function returns the status of the channel.

Example
```
var status: boolean;
BEGIN
  ConfigIOChannelAsInput(2);
  status := ReadIOchannel(2); // Read Input channel 2
END;
```

ReadIOchip IO

Syntax
```
FUNCTION ReadIOchip(Chip_no: Longint): Longint;
```

Parameter
`Chip_no:` Value between 0 and 7 which corresponds to the address setting of the Input/Output chip in which the status of the inputs are to be read.

Result
`IOdata:` The bits in the 'IOdata' variable of the selected IO-chip are changed according to the status of the IO-channels. A high (1) means that the channel has been cleared, a low (0) means that it has been cleared.

Description
The status of all channels of the selected Input/Output chip are read and registered in the 'IOdata' variables. The function returns the status of the chip inputs.

Example
```
var status: longint;
BEGIN
  ConfigIOchipAsInput(0);
   status := ReadIOchip(0); // Read Input channels from chip 0
END;
```

ReadIOconfigArray IO

Syntax
```
PROCEDURE ReadIOconfigArray(Buffer:Pointer);
```

Parameter
`Buffer:` A pointer to the array of long integers where the IO configuration data will be read.

Description
The IO configuration data array is read from the K8D.DLL to the application program.

Example
```
var  // global variables
  IOconfig: ARRAY[0..MaxIOchip] OF Integer;

procedure TForm1.Button1Click(Sender: TObject);
var p:pointer;
i:integer;
s:string;
begin
  p:=@IOconfig;    // Address of the data buffer for config array
  ReadIOconficArray(p); // Read the data from K8D.DLL
  memo1.clear;
  s:='';
  for i:=0 to MaxIOchip do s:=s +inttostr(IOconfig[i])+chr(9);
  memo1.lines.add(s); // Display the IO cofig data
end;
```

ReadIOconfigArray IO

Syntax
```
PROCEDURE ReadIOdataArray(Buffer:Pointer);
```

Parameter
`Buffer:` A pointer to the array of long integers where the IO data will be read.

Description
The IO status data array is read from the K8D.DLL to the application program.

Example
```
var  // global variables
  IOdata: ARRAY[0..MaxIOchip] OF Integer;

procedure TForm1.Button1Click(Sender: TObject);
var p:pointer;
i:integer;
s:string;
begin
  p:=@IOdata;      // Address of the data buffer for the data array
  ReadIOdataArray(p); // Read the data from K8D.DLL
  memo1.clear;
  s:='';
  for i:=0 to MaxIOchip do s:=s +inttostr(IOdata[i] and $ff)+chr(9);
  memo1.lines.add(s); // Display the IO data
end;
```

ReadDACarray DAC

Syntax
```
PROCEDURE ReadDACarray(Buffer:Pointer);
```

Parameter
`Buffer:` A pointer to the data array of long integers where the DAC data will be read.

Description
The DAC data array is read from the K8D.DLL to the application program.

Example
```
var  // global variables
  DAC: ARRAY[1..MaxDACchannel] OF Integer;
procedure TForm1.Button1Click(Sender: TObject);
var p:pointer;
i:integer;
s:string;
begin
  p:=@DAC;         // Address of the data buffer for the DAC array
  ReadDACarray(p); // Read the data from K8D.DLL
  memo1.clear;
  s:='';
  for i:=1 to MaxDACchannel do s:=s +inttostr(DAC[i] and $ff)+chr(9);
  memo1.lines.add(s); // Display the DAC dataend;
```

ReadIOconfigArray DA

Syntax
```
PROCEDURE ReadDAarray(Buffer:Pointer);
```

Parameter
`Buffer:` A pointer to the data array of long integers where the data will be read.

Description
The DA data array is read from the K8D.DLL to the application program.

Example
```
var  // global variables
  DA: ARRAY[1..MaxDAchannel] OF Integer;

procedure TForm1.Button1Click(Sender: TObject);
var p:pointer;
i:integer;
s:string;
begin
  p:=@DA;          // Address of the data buffer for the DA array
  ReadDAarray(p);  // Read the data from K8D.DLL
  memo1.clear;
  s:='';
  for i:=1 to MaxDAchannel do s:=s +inttostr(DA[i] and $ff)+chr(9);
  memo1.lines.add(s); // Display the DA data
end;
```

Using the DLL in Delphi

Here is the declaration of some array variables, K8D.DLL procedures and functions and the two most important procedures of the application program (`FormCreate` and `FormClose`).

```
var
  IOconfig: ARRAY[0..MaxIOchip] OF Integer;
  IOdata: ARRAY[0..MaxIOchip] OF Integer;
  DAC: ARRAY[1..MaxDACchannel] OF Integer;
  DA: ARRAY[1..MaxDAchannel] OF Integer;


implementation

{$R *.DFM}

{IO CONFIGURATION PROCEDURES}
PROCEDURE ConfigAllIOasInput; stdcall; external 'K8D.dll';
PROCEDURE ConfigAllIOasOutput; stdcall; external 'K8D.dll';
PROCEDURE ConfigIOchipAsInput(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE ConfigIOchipAsOutput(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE ConfigIOchannelAsInput(Channel_no: TIOchannel); stdcall; external 'K8D.dll';
PROCEDURE ConfigIOchannelAsOutput(Channel_no: TIOchannel); stdcall; external 'K8D.dll';

{UPDATE IOdata & IO ARRAY PROCEDURES}
PROCEDURE UpdateIOdataArray(Chip_no: TIOchip; Data:Longint); stdcall; external 'K8D.dll';
PROCEDURE ClearIOdataArray(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE SetIOdataArray(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE SetIOchArray(Channel_no: TIOchannel); stdcall; external 'K8D.dll';
PROCEDURE ClearIOchArray(Channel_no: TIOchannel); stdcall; external 'K8D.dll';

{OUTPUT PROCEDURES}
PROCEDURE IOoutput(Chip_no: TIOchip ; Data: Longint); stdcall; external 'K8D.dll';
PROCEDURE UpdateAllIO; stdcall; external 'K8D.dll';
PROCEDURE ClearAllIO; stdcall; external 'K8D.dll';
PROCEDURE SetAllIO; stdcall; external 'K8D.dll';
PROCEDURE UpdateIOchip(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE ClearIOchip(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE SetIOchip(Chip_no: TIOchip); stdcall; external 'K8D.dll';
PROCEDURE SetIOchannel(Channel_no: TIOchannel); stdcall; external 'K8D.dll';
PROCEDURE ClearIOchannel(Channel_no: TIOchannel); stdcall; external 'K8D.dll';

{6 BIT DAC CONVERTER PROCEDURES}
PROCEDURE OutputDACchannel(Channel_no: TDACchannel ; Data: Longint); stdcall; external
'K8D.dll';
PROCEDURE ClearDACchannel(Channel_no: TDACchannel); stdcall; external 'K8D.dll';
PROCEDURE SetDACchannel(Channel_no: TDACchannel); stdcall; external 'K8D.dll';
PROCEDURE ClearDACchip(Chip_no: TIOcard); stdcall; external 'K8D.dll';
PROCEDURE SetDACchip(Chip_no: TIOcard); stdcall; external 'K8D.dll';
PROCEDURE ClearAllDAC; stdcall; external 'K8D.dll';
PROCEDURE SetAllDAC; stdcall; external 'K8D.dll';

{8 BIT DA CONVERTER PROCEDURES}
PROCEDURE OutputDAchannel(Channel_no: TDAchannel ; Data: Longint); stdcall; external
'K8D.dll';
PROCEDURE ClearDAchannel(Channel_no: TDAchannel); stdcall; external 'K8D.dll';
PROCEDURE SetDAchannel(Channel_no: TDAchannel); stdcall; external 'K8D.dll';
PROCEDURE ClearAllDA; stdcall; external 'K8D.dll';
PROCEDURE SetAllDA; stdcall; external 'K8D.dll';

{GENERAL PROCEDURES}
PROCEDURE SelectI2CprinterPort(Printer_no: Longint); stdcall; external 'K8D.dll';

PROCEDURE Start_K8000; stdcall; external 'K8D.dll';
PROCEDURE Stop_K8000; stdcall; external 'K8D.dll';

{INPUT FUNCTIONS}
function ReadIOchip(Chip_no: TIOchip):longint; stdcall; external 'K8D.dll';
function ReadIOchannel(Channel_no: TIOchannel):boolean; stdcall; external 'K8D.dll';
function ReadADchannel(Channel_no:TADchannel):longint; stdcall; external 'K8D.dll'
PROCEDURE ReadIOconficArray(Buffer:Pointer); stdcall; external 'K8D.dll';
PROCEDURE ReadIOdataArray(Buffer:Pointer); stdcall; external 'K8D.dll';
PROCEDURE ReadDACarray(Buffer:Pointer); stdcall; external 'K8D.dll';
PROCEDURE ReadDAarray(Buffer:Pointer); stdcall; external 'K8D.dll';

procedure TForm1.FormCreate(Sender: TObject);
begin
  Start_K8000;
```

```
end;

procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Stop_K8000;
end;
```

Using the DLL in Visual basic

Here is the declaration of the K8D.DLL procedures and functions, some array variables, and the two most important procedures of the application program (Form_Load and Form_Terminate).

```
Option Explicit

'Declare use of the DLL
'K8D.DLL interfaces

'IO CONFIGURATION PROCEDURES
Private Declare Sub ConfigAllIOasInput Lib "k8d.dll" ()
Private Declare Sub ConfigAllIOasOutput Lib "k8d.dll" ()
Private Declare Sub ConfigIOchipAsInput Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub ConfigIOchipAsOutput Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub ConfigIOchannelAsInput Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub ConfigIOchannelAsOutput Lib "k8d.dll" (ByVal Channel_no As Long)

'OUTPUT PROCEDURES
Private Declare Sub IOoutput Lib "k8d.dll" (ByVal Chip_no As Long, ByVal Data As Long)
Private Declare Sub UpdateAllIO Lib "k8d.dll" ()
Private Declare Sub ClearAllIO Lib "k8d.dll" ()
Private Declare Sub SetAllIO Lib "k8d.dll" ()
Private Declare Sub UpdateIOchip Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub ClearIOchip Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub SetIOchip Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub SetIOchannel Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub ClearIOchannel Lib "k8d.dll" (ByVal Channel_no As Long)

'INPUT FUNCTIONS AND PROCEDURES
Private Declare Function ReadIOchip Lib "k8d.dll" (ByVal Chip_no As Long) As Long
Private Declare Function ReadIOchannel Lib "k8d.dll" (ByVal Channel_no As Long) As
Boolean
Private Declare Sub ReadIOconficArray Lib "k8d.dll" (Array_Pointer As Long)
Private Declare Sub ReadIOdataArray Lib "k8d.dll" (Array_Pointer As Long)
Private Declare Sub ReadDACarray Lib "k8d.dll" (Array_Pointer As Long)
Private Declare Sub ReadDAarray Lib "k8d.dll" (Array_Pointer As Long)
'How to use these calls:
' ReadIOconficArray IOconfig(0)
' ReadIOdataArray IOdata(0)
' ReadDACarray DAC(1)
' ReadDAarray DA(1)

'6 BIT DAC CONVERTER PROCEDURES
Private Declare Sub OutputDACchannel Lib "k8d.dll" (ByVal Channel_no As Long, ByVal Data
As Long)
Private Declare Sub ClearDACchannel Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub SetDACchannel Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub ClearDACchip Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub SetDACchip Lib "k8d.dll" (ByVal Chip_no As Long)
Private Declare Sub ClearAllDAC Lib "k8d.dll" ()
Private Declare Sub SetAllDAC Lib "k8d.dll" ()

'8 BIT DA CONVERTER PROCEDURES
Private Declare Sub OutputDAchannel Lib "k8d.dll" (ByVal Channel_no As Long, ByVal Data
As Long)
Private Declare Sub ClearDAchannel Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub SetDAchannel Lib "k8d.dll" (ByVal Channel_no As Long)
Private Declare Sub ClearAllDA Lib "k8d.dll" ()
Private Declare Sub SetAllDA Lib "k8d.dll" ()

'8 BIT AD CONVERTER FUNCTION
Private Declare Function ReadADchannel Lib "k8d.dll" (ByVal Channel_no As Long) As
Boolean

'GENERAL PROCEDURES
Private Declare Sub SelectI2CprinterPort Lib "k8d.dll" (ByVal port As Long)
Private Declare Sub Start_K8000 Lib "k8d.dll" ()
Private Declare Sub Stop_K8000 Lib "k8d.dll" ()

'COMMON USED GLOBALS
Const MaxIOcard As Long = 3
Const MaxIOchip As Long = 7
Const MaxDACchannel As Long = 32
Const MaxDAchannel As Long = 4

'Declare variables
Dim IOconfig(0 To MaxIOchip) As Long
```

```
Dim IOdata(0 To MaxIOchip) As Long
Dim DAC(1 To MaxDACchannel) As Long
Dim DA(1 To MaxDAchannel) As Long

Private Sub Form_Load()
    Start_K8000
End Sub

Private Sub Form_Terminate()
    Stop_K8000
End Sub
```