USER'S REFERENCE MANUAL

SCILIB32

SCIDYNE 32-Bit Universal Dynamic Link Library

Model No.	100-7645			
Doc. No.	M7645	Rev:	0.1	08/02/00

Beta Release

This documentation and associated software is considered preliminary and is provided "as is" without any expressed or implied warranties, claims of usefulness, suitability, performance, accuracy, or reliability. SCIDYNE reserves the right to make changes without notice or consideration to existing users. The user assumes all responsibility and risks when using this software.

SCIDYNE®
649 School Street / Pembroke, MA 02359 USA / Tel: (781) 293-3059
www.scidyne.com

Previous revision: None

DISCLAIMER: This document contains proprietary information regarding SCIDYNE and its products. The information is subject to change without notice. SCIDYNE makes no warranty of any kind with regard to this material, including but not limited to, the implied warranties of merchantability and fitness for a particular purpose. SCIDYNE shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance, or use of this material. No part of this document may be duplicated in any form without prior written consent of SCIDYNE.

LIFE SUPPORT POLICY

Certain applications may involve the risks of death, personal injury or severe property or environmental damage ("Critical Applications").

SCIDYNE products are not designed, intended, authorized or warranted to be suitable for use in life-support applications, devices or systems or other critical applications without the express written approval of the president of SCIDYNE.

Table of Contents

Conventions and Terminology Used Throughout This Publication

Safety and Usage Conventions

> ✎ NOTE: *Contains important information and useful tips that will assist in the understanding and operation of the product.*

> ✋ CAUTION: *Calls attention to a procedure, practice or condition that could possibly cause personal injury or damage to equipment.*

> ⚠ WARNING: *Calls attention to a procedure, practice or condition that could possibly cause severe bodily injury, death or extensive equipment damage.*

Terminology

Host This is the computer or similar device using SCILIB32.

Numbering Systems Computerized equipment often requires its numeric data to be represented in different forms depending on the audience and information being conveyed. Decimal numbers are typically used for end-user data entry and display while internally these values are converted and manipulated in native binary. Hexadecimal numbers are often used by programmers as an intermediate level between binary and decimal notations.

Base	Name	Format (MS <---> LS)
2	Binary	1011 1001
10	Decimal	185
16	Hexadecimal	0xB9 or $B9_{16}$ or &HB9

Multi-Byte Word Formats
Unless otherwise specified numbers or registers spanning multiple bytes are stored in "little endian" format. The first address (ADDR+0) will contain the Least Significant Byte (LSB) while the Most Significant Byte (MSB) will reside at the highest address.

ADDR+0	ADDR	ADDR+n
LSB	LS <----> MS	MSB

Supported Data Types

The following table shows the supported data types, including storage sizes and ranges.

Data type	Size	Range storage
Byte	1 byte	0 to 255
Boolean	2 bytes	True or False
Integer	2 bytes	-32,768 to 32,767
Long (long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
Single (single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E38 for positive values
Double (double-precision floating-point)	8 bytes	-1.79769313486232E308 to -4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486232E308 for positive values
String (variable-length)	10 bytes + string length	0 to approximately 2 billion
String (fixed-length)	Length of string	1 to approximately 65,400
Variant (with numbers)	16 bytes	Any numeric value up to the range of a Double
Variant (with characters)	22 bytes + string length	Same range as for variable-length String
User-defined (using Type)	Number required by elements	The range of each element is the same as the range of its data type.

Acknowledgments

VisualBasic and Windows are registered trademarks of the Microsoft corporation.

Introduction

SCILIB32 is a Dynamic Link Library (DLL) which adds low level I/O operations and driver support for SCIDYNE peripheral boards to languages such as Microsoft's Visual basic. It is written in C++ and will run on 32-Bit operating systems such as Windows95, Windows98 and Windows2000.

The purpose of SCILIB32 is to provide a layer of abstraction which relives the programmer from having to understand the lower levels of operation for each SCIDYNE peripheral board. It also provides a common API (Application Program Interface) which enables easy software transitions when migrating between varies SCIDYNE products.

Intended Audience

This documentation assumes the reader is familiar with software programming in general and is proficient using the Visual Basic programming language. The intent is to illustrate the use of SCILIB32 and not to educate the reader on how to program software.

Installation

SCILIB32.DLL is generally placed in the "windows" directory of the host computer. However, it can be placed in any directory the user chooses provided that the system PATH points to its location or the user specifies its location in all DECLARE statements.

VB Declarations

In order for Visual Basic to make use of SCILIB32, the functions and subroutines contained within it must first be declared. This is done at the general declaration level of a Visual Basic project as shown below. Only those SCILIB32 functions which the user's program will actually use need to be declared.

```
option explicit

.... other declarations

Private  Declare  Function outbyte Lib "scilib.dll" (ByVal  portaddress As Integer, ByVal  portdata As Integer) As Integer

.... other declarations
```

NOTE: *The text length of some declarations shown in this document exceed the width of the printed page. For this reason the declaration text is shown simply continuing on the next line. However, when entering the declaration in VisualBasic, it MUST be entered complete all on one line or an error will occur.*

Function: return_single = get_version() **Revision:** 1.00

Description: Retrieves the build version number of SCILIB32

Declaration: Declare Function get_version Lib "scilib32.dll" ( ) As Single

Conditions:

Before: None

After: return_single = Single-precision floating point value representing build version of SCILIB32

Example: Determine the version of the SCILIB32 you are using. Result shown in a text box.

```
Text1.Text = get_version()      ‘ Retrieve the SCILIB32 build version
```

Function: return_integer = inbyte(portaddress) **Revision:** 1.00

Description: Reads a single 8-bit byte of data from an I/O hardware port.

Declaration: Declare Function inbyte Lib "scilib32.dll" (ByVal portaddress As Integer) As Integer

Conditions:

Before: portaddress = is a 16-bit integer value representing the I/O mapped 8-bit hardware location to be read. The valid range is 0 to 1023 (0x000 to 0x3ff) for most PC/AT computers.

After: return_integer = Any valid variable which will be loaded with byte data read from portaddress. Although an integer is returned only the lower 8-bits (0-7) will contain valid byte data. The upper bits (8-15) will sign extend bit-7 of the lower byte.

Example: Read the Status Register of the computers LPT1 port. The returned decimal result is displayed in a text box.

```
Text1.Text = inbyte(&H379)          ‘ Read the LPT1 Status Port
```

Function: return_integer = inword(portaddress) **Revision:** 1.00

Description: Reads a single 16-bit word of data from an I/O hardware port.

Declaration: Declare Function inword Lib "scilib32.dll" (ByVal portaddress As Integer) As Integer

Conditions:

Before: portaddress = is a 16-bit integer value representing the I/O mapped 16-bit hardware location to be read. The valid range is 0 to 1023 (0x000 to 0x3fe) for most PC/AT computers and should generally be an even number.

After: return_integer = 16-bit integer value of the data at portaddress:portaddress+1.
The least significant byte is generally located at portaddress.

Example: Read the value of a register of located at address 768. Result is displayed in a text box.

```
Text1.Text = inword(&H300)          ‘ Read a 16-bit Port
```

Function: return_integer = outbyte(portaddress, portdata) **Revision:** 1.00

Description: Writes a single 8-bit byte of data to an I/O hardware location.

Declaration: Declare Function outbyte Lib "scilib32.dll" (ByVal portaddress As Integer, ByVal portdata As Integer) As Integer

Conditions:

Entry: portaddress = is a 16-bit integer value representing the 8-bit I/O mapped hardware location to be written. The valid range is 0 to 1023 for most PC/AT computers.

portdata = A 16-bit integer containing the 8-bit byte value to be written. The lower 8-bits are written and the upper 8-bits are ignored.

After: return_integer = Echos the value of portdata.

Example: Write a byte of data to a port at address 768.

```
Dim temp As Integer


temp = outbyte( &H300, 32 )                    ‘Write 0010 0000 to port 768
```

CAUTION: *This function make no determination as to the validity of the portaddress or portdata parameters. Computer operation and functionality can be compromised if incorrect values are specified.*

Function: return_integer = outword(portaddress, portdata) **Revision:** 1.00

Description: Writes a single 16-bit value to an I/O hardware location.

Declaration: Declare Function outword Lib "scilib32.dll" (ByVal portaddress As Integer, ByVal portdata As integer) As Integer

Conditions:

Entry: portaddress = is a 16-bit integer value representing the I/O mapped 16-bit hardware location to be written. The valid range is 0 to 1023 for most PC/AT computers and should be an even number.

portdata = a 16-bit value to be written.

After: return_integer = Echos the value of portdata.

Example: Write the decimal value 1234 to 16-bit I/O hardware location 768

```
Dim temp As Integer

temp = outword( 768, 1234 )      ‘Perform word output
```

CAUTION: *This function make no determination as to the validity of the portaddress or portdata parameters. Computer operation and functionality can be compromised if incorrect values are specified.*