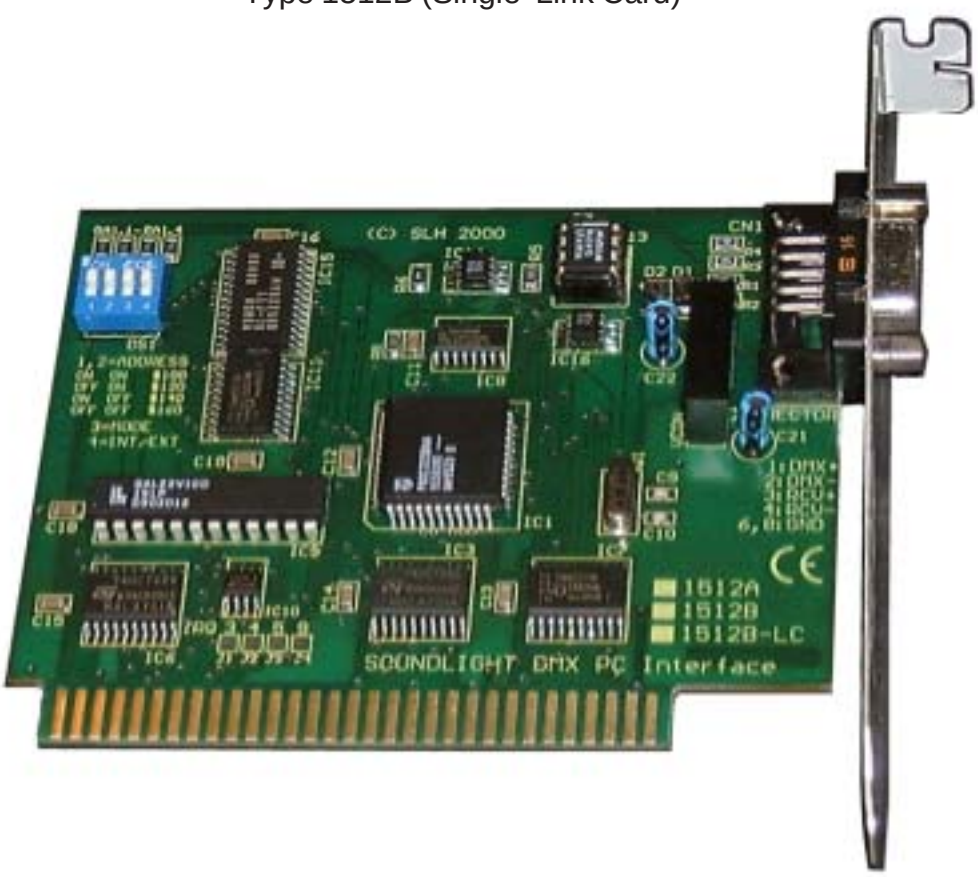
Ver 06/99

User-Manual

DMX 512 PC Interface Card Mk2

Type 1512B (Single Link Card)



SOUNDLIGHT
The DMX Company

Contract of Use

Before putting this product to work, please read the following license agreement thoroughly. If you do not accept these terms of license, please return all materials unopened to your supplier. Only this can void the license agreement. By opening the product package or installation of the software you accept the license agreement.

Terms of Use

The sole right to duplicate programs (and parts thereof) remains with SOUNDLIGHT. The copyright of the written materials remains with SOUNDLIGHT.

The conditions of use resemble those of a book:
You may use and operate the product at different locations, but not from multiple computers simultaneously. You are not allowed to change, modify or re-engineer the software supplied.

Limited Warranty

SOUNDLIGHT warrants the functionality of the materials delivered. Warranty is limited to 6 month and repair/ replacement of defective components at our decision. Repair or replacement does not prolong the initial warranty.

Liability

The customer has to check carefully the suitability of the materials for its intended use. A suitability for a special purpose is not given, software is delivered "as is". No liability for damages of any kind, direct or consequential, is accepted.

If you have any questions regarding these terms, please contact:

SOUNDLIGHT
The DMX Company
Glashuettenstrasse 11
D-30165 Hannover GERMANY
Tel: 0511-3730-267
Fax: 0511-3730-423
E-Mail: info@soundlight.de

CE

The DMX Interface Card 1512B has been CE certified in conjunction with a Gateway P5-90 (Pentium) Computer.

FCC Notice:

This equipment has been tested and found to comply with the limits for class B digital devices, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed in accordance with the instruction, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by switching the equipment off and on, the user in encouraged to to correct the interference by one or more of the following measures:
- Reorient or relocate the receiving antenna.
- Increase the separation between the equipment and the receiver
- Connect the equipment into an outlet on a circuit different to that to which the receiver is connected.
- Consult the dealer or an experienced radio/TV technician for help.

Introduction

DMX512 has gained wide acceptance within the lighting industry. The protocol, originally published by USITT (United States Institute for Theatre Technology, Inc., New York, http://www.usitt.org) uses the common RS-485 interface for data transmission and describes a method to transfer control information for up to 512 channels (slots) of information. A copy of this proposal has been published as german standard DIN 56930-2, which is available through Beuth-Verlag, Berlin (http://www.beuth.de).

At present, the international protocol working group, headed by ESTA (Entertainment and Services Technology Association, http://www.esta.org) is working on an implementation of DMX-512/2000. This protocol is expected to be fully compatible to DMX512/1990, but requires additional protection. As all DMX ports of the 1512B interface card are optically isolated, this card already fulfills these requirements.

The advantage of digital control is the precision of control, the speed achieved and the simple wiring scheme of the attached units due to the use of a bus structure.

These are the advantages presented by our DMX512 interface cards 1512A, 1512B and 1512B. These cards. can control up to 512 dimming channels 44 times per second, which is equivalent to a data transfer rate of 250,000 bit/second.

All interface cards have been designed for use within a IBM compatible PC, equipped with ISA-Bus. Though is does not matter which PC architecture you are using (any CPU type from 8088, 8086, 80286, 80386, 80486 through Pentium or Athlon will do), we recommend to use a fast machine when designing complex graphic user interfaces.

The interface by itself is a so-called "intelligent card", that is, the card has its own microprocessor who takes care for the generation of the DMX512 signal and the data output. Thus the PC CPU is freed from these tasks and can run your application program at full speed. Though there are standard RS-485 interface cards available, these cards lack on-board intelligence and have to be driven continuously by the PC host CPU. As the DMX signal has to be generated continuously, this consumes immense computing power and slows down the application software considerably. Our cards just need synchronization and leave the full computing power at your application.

To put the DMX interface to work, please read the following chapters demonstrating the installation and the use of the software.

All applications and informations regarding operation of the interface card 1512B can be found on the accompanying CD, folder \DMX\1512B. Late news which could not be printed can be found in \DMX\1512B\LATEST. If available, eech folder has a READ.ME or README.TXT file. Please read these files before using or copying data, or before installing programs. Remember, that most information given is for developers and required special attention.
Also please check our website regularly. This is a special support website for DMX512 pc interfaces and is available at

http://www.pcdmx.com

Have fun, and good luck!

If you have any comments and suggestions, want to report errors or submit your application for publication, we always welcome your input. Simply mail to:

info@pcdmx.com

Table of contents

SOUNDLIGHT
The DMX Company

1.1 Installation of the DMX512 Interface card 1512B

These are the few steps required to install your card:

1. Unplug the computer from mains supply !!
   This applies especially to computers with ATX power supply, as the internal PSU continues to deliver power even if the computer has been switched off by softweare command, and thus the PC may start up erroneously.

2. Open the case

3. Remove the fastening screw from a free ISA slot (usually black)

4. Gently insert the card and press into slot

5. Re-fit fastening screw

6. Close the computer housing and plug the DMX512 adapter into the 9-pin outlet of the DMX interface card

IMPORTANT

Always make sure that the computer is completely separated from any mains supply when installing or replacing interface cards, connecting or disconnecting plugs and cables or setting jumpers.

If you are not absolutely familiar with how to install, change or replace interface cards, please refer to a qualified technician.

1.2 Making a DMX Adaptor

The DMX interface card 1512B consists of two separate and optically isolated DMX ports, and thus features two female 9-pin Sub D output connector jacks. Each port has a DMX input and a DMX output, thus you may need two identical cable adaptors.

The USITT standard DMX512/1990 requires a 5-pin XLR plug as standard DMX512 connector. This is the standard pinout of the XLR connector:

Pin 1 GND, screen, shield
Pin 2 DMX -
Pin 3 DMX +
Pin 4 Reserve (DMX - 2. Link)
Pin 5 Reserve (DMX + 2. Link)

DMX outputs shall use female, DMX inputs shall use male connectors.

To prepare a suitable adaptor cable, please connect:

COMPUTER **DMX-Equipment**
Sub-D 9pin XLR 5-pin

2 <----------> 2 XLR PLUG FEMALE: TRANSMIT (OUTPUT) DMX -
1 <----------> 3 XLR PLUG FEMALE: TRANSMIT (OUTPUT) DMX +
6 <----------> 1 XLR PLUG FEMALE: GND / screen

do not use Pin 5

1.3 Addresses of the DMX interface card

To be able to communicate with the interface card, the computer needs one or more interface addreses where to access the card. The interface address is set by DIP coding switches. It is only necessary to set the address if more than one DMX512 interface card is used within the samecomputer, or (which is very unlikely) the standard address is already in use by another interface card.

These are the adresses available:

Hex	**Decimal**
$0100	256
$0120	288
$0140	320
$0160	355

Only four individual and consecutive addresses, beginning with the given base address, are used. (for example $100...$103).

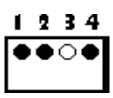
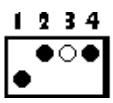
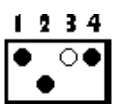
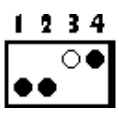
The base adress of the DMX interface card is set to $100..$103 (256..260 dec) when shipped.

In most cases this will ensure trouble-free operation, since there are only very few cards known to use this same address range. If you have problems running the card at the initial adress setting, try another address setting.

The software will detect the presence of the card automatically, regardless of the address setting, using the routine DMXCARD() as specified later.

For cards equipped with DIP switches the card base adress can be set using DIP switches 1 & 2.

These settings apply:

Hex	Decimal	Jumper setting	
			
$0100	256	SW1 = ON SW2 = ON	
$0120	288	SW1 = OFF SW2 = ON	
$0140	320	SW1 = ON SW2 = OFF	
$0160	352	SW1 = OFF SW2 = OFF	

Switches SW3 and SW4 are factory set (both switches ALWAYS set to ON) and should NOT be changed.

The following chapters cover programming of the card and show some examples for different programming languages. If you want to skip programming the card, please proceed to chapter 2.6 "Application Programs".

1.4 How the card works

The DMX interface card should be low-priced, but at the same time as flexible as can be. This allows to adapt the card to different needs, and upgrade features simply by downloading new software. Your 1512B card allows you to do so.

Looking at the card will unveil two things: first, there is a 80C3x microprocessor, and second, there is no ROM or EPROM to contain the uP operating system software. The program memory is fully simulated by RAM area, and the card OS downloaded before the card is initalized. Changing or upgrading the card operating system can simply be done by replacing the download file. Thus the card can be very easily upgraded to future versions of DMX512, should they ever come.

Even changig the operating system while the card is running is possible. The solution we have selected, has lots of advantages, but requires to load the OS every time before you start your application. A special function, supplied in a Dynamic Link Library (DLL), does the job for you.

To ensure maximun compatibility to all PCs we have chosen the 8 bit wide bus. As all DMX data simply represent 8-bit data, this is no restriction. If 16 it data are needed, DMX by itself requires to split these data into a HighByte and a LowByte, the order of which is solely defined by the unit (e.g. scanner) attached to the DMX line.

All data and the card operating system are held within an on-board RAM, which can be accessed through the card base address, and the subsequent four adresses. The onboard RAM areaallocation is as follows:

$0000 - $03FF	Block 0-3: Operating system of the 8032/80C320 CPU
$0400 - $05FF	Block 4-5: DMX data
$0600 - $1FFF	free for user data

Within the card, there are two data paths:

- from the PC into the DMX RAM and vice versa;
- from the DMX RAM to the RS-485 output.

The PC has nothing more to do than write the output data into the DMX memory, beginning from $0400. 512 DMX channels occupy 512 bytes of data, thus Channel 1 -> $0400, channel 2 -> $0401 ... channel 512 -> $05FF.

This completes the data transfer from the PC, and the host CPU as free for other tasks. The card CPU collects the data, generates the DMX data stream and sends all to the RS-485 output.

Both microprocessors have to access the same RAM, and that requires that to avoid collision, the card CPU be halted while the PC accesses the card. Onboard circuitry is supervising the PC access and controls the card processor accordingly.

PUTTING THE CARD TO WORK

Before writing DMX data to the card, you have to transfer the card OS into the card RAM and then start the card CPU. If special working parameters are required, these have to be set aftertransferring the card OS. How to detect the presence of a card, transfer the program file and parameters is described later.

After starting the card CPU the transmission of serial DMX data begins and continues until the card is disabled or a complete packet has been sent. When initalizing the card for the very first time, all RAM contents will be erased from memory. Every consecutive RESET command leaves tha data i n t a c t , that is, only changes have to be written to the memory. DMX512 by itself will repeat all data, changed or not, and the card CPU takes care of this task.

Transmission of data is limited to one complete packet, and repeated upon request from the host PC. Beforeissuing a repeat command, the user shall update changed channel values. This ensures highes data integrity for 16bit data, which are usually being transferred as twio 8-bit words: because a transmission can never start while one of these bytes is being updated, the receiver will always receive data pairs belonging together. Remember, that DMX512 by itself only defines 8-bit data, but all practical uses, mainly for moving lights, require 16-bit resolution - at least for the movement. An interface card using a nonsynchronized scheme will not be able delivering consistent 16-bit data over DMX512.

2 The Software

The software for this interface card consists of several parts:

- ready-to-use application programs for MS Windows
- the PC card operating system
- example programs
- a Dynamic Link Library (DLL) for Windows

Information for Windows Users:

Several projects are availble in source code format. Please have a look at our support website http://www.pcdmx512.com and check the SOFTWARE / SKELETONS department. Wherever possible, all source code files are saved as ASCII files. This allows to open and browse the files using any ASCII text editor.

2.1 The Card Operating System

The card operating system is needed to enable the card fulfulling the basic functions: sending and receiving DMX512. Full compatibility to both DMX standards, USITT DMX-512/1990 and DIN 56930-2, is ensured. Furthermore the card will be able to send/receive future DMX-512/2000.

The operating system file is a file specific to a card type and is supplied as binary file. This file must be transferred into the card memory area.

Card type	Filename
1512A	SLHDMX12.BIN
1512B	SLHDMX16.BIN
1512B/LC	SLHDMX12.BIN
1512C	SLHDMX17.BIN

If you need a DOS loader to transfer the card OS, a suitable program can be found as file INITLO.EXE in folder \DMX\1512B\SOURCES. Please refer to the appropriate programming language section to find the method for OS transfer.
Windows applications can use the vbWOS(CardAdr) call to transfer the OS file, or copy the file contents byte by byte. Examples for various programming languages can be found asprogramming skeletons on our support website, http://www.pcdmx.com.

Before starting the operating system transfer, it is necessary to reset the card once. This is accomplished by reading the card base address or, using Windows applications, calling the vbDMXStop(CardAdr) function. Aftert ransferring the OS file, remove the reset by reading the card base Address+3 or calling the DLL function vbDMXStart(Cardadr).

2.2 The Windows DLL

The Windows folder contains a Dynamic Link Library SLHDMX2.DLL (16 Bit version, for Windows 3.1/ 3.11) and a DLL SLHDMX33.DLL (32 Bit Version, for Windows 95/98/Millennium). These libraries contain all functions necessary to communicate with the card:

Name:	**vbCardAdr**
Purpose:	Detect the adress of the interface card within the PC
for cards:	1512A, 1512B, 1512B/LC, 1512B
Type:	Function
Calling parameters:	0 (Word) or CardAdr (Word)
Return parameters:	CardAdr% (Word)

Example:
Declare Function vbCardadr% Lib "SLHDMX2.DLL" (ByVal w%)

Calling parameter: 0
The function vbCardAdr checks, if a DMX interface card is present and returns the card base address. If no card can be found, Adress 0 will be returned.

Calling parameter: CardAdr
Checks if a card is present at the specified base address. If found, card adress is returned, if none present, zero returned.

Name:	**vbWByte**
Purpose:	Write a Byte into the DMX card RAM
for cards:	1512A, 1512B, 1512B/LC, 1512B
Type:	Sub
Calling parameters:	CardAdr% (Word) ByteAdr% (Word) Ausgabe% (Byte)
Return parameters:	-

Example:
Declare Sub vbWByte Lib "SLHDMX2.DLL" (ByVal CardAdr%, ByVal ByteAdr%, ByVal Ausgabe%)

The Sub vbWByte write a byte of data into the card RAM area. To do so, the RAM address (Word, written as two Bytes: HiByte, LowByte) and the data (Byte) have to be transferred.

Example:	write value $80 (80h, 50%) as DMX channel 1 vbWByte &H100, &H400, &H80

Name:	**vbRByte**
Purpose:	Reads a Byte from the DMX card RAM
for cards:	1512A, 1512B, 1512B/LC, 1512B
Type:	Function
Calling parameters:	CardAdr% (Word) ByteAdr% (Word)
Return parameters:	Byte% (Byte)

Example:
Declare Function vbRByte Lib "SLHDMX2.DLL" (ByVal CardAdr%, ByVal ByteAdr%)

The Function vbRByte reads a byte of data from the card RAM area. To do so, the RAM address (Word, written as two Bytes: HiByte, LowByte) has to be transferred.

Example:	read card ID from RAM address $03F9 byte% = vbRByte (&H100, &H3F9)

Name: **vbDMXStart**
Purpose: starts the DMX transmission
for cards: 1512A, 1512B, 1512B/LC
Type: Function
Calling parameters: CardAdr% (Word)
Return parameters: Dummy% (Byte)

Example:
Declare Function vbdmxstart% Lib "SLHDMX2.DLL" (ByVal CardAdr%)

Name: **vbDMXStop**
Purpose: Stops the DMX transmission
for cards: 1512A, 1512B, 1512B/LC
Type: Function
Calling parameters: CardAdr% (Word)
Return parameters: Dummy% (Byte)

Example:
Declare Function vbDMXStop% Lib "SLHDMX2.DLL" (ByVal CardAdr%)

Name: **vbDMXStop**
Purpose: Stops the card CPU (Reset)
for cards: 1512B
Type: Function
Calling parameters: CardAdr% (Word)
Return parameters: Dummy% (Byte)

Example:
Declare Function vbDMXReset% Lib "SLHDMX2.DLL" (ByVal CardAdr%)

Name: **vbDMXStart**
Purpose: Starts the card CPU (Reset off)
for cards: 1512B
Type: Function
Calling parameters: CardAdr% (Word)
Return parameters: Dummy% (Byte)

Example:
Declare Function vbDMXResOff% Lib "SLHDMX2.DLL" (ByVal CardAdr%)

Name: **vbWOS**
Purpose: transfers the standard OS
for cards: 1512A, 1512B, 1512B/LC, 1512B
Type: Sub
Calling parameters: CardAdr% (Word), Cardtype (Byte)
1512A: Cardtype = 12
1512B: Cardtype = 16
1512B/LC: Cardtype = 12
1512C: Cardtype = 17

Return parameters: -

Example:
Declare Sub vbWOS Lib "SLHDMX2.DLL" (ByVal CardAdr%, ByVal Cardtype%)

DUMMY% denominates a value without meaning.

Please refer to the source code of application "DESK6", which shows how to use the DLL functions within your project.

Memory allocation of DMX Card 1512B:

$0000-$03EF	Operating system
$03F0-$03FE	Operating system parameters
$0400-$05FF	Send data for channels 1- 512
$0600-$0FFF	reserved
$1000-$1FFF	free (User RAM)

Operating System Parameters:

$03F0 Start Sync Duration — typical value = 60
Length of sync pulse (RESET). Minimum value required for DMX512 is 88us.
value = time (us) / 2 * 16 / 12

$03F1 Startbyte value — typical value = 0
Value of send startbyte

$03F3 channel count LowByte (0-255) — typical value = 0
$03F4 channel count HighByte (0-2) — typical value = 2

The bytes for the channel counter have to be set according to the following table:
LO: 001 HI: 000 1 sent channels
LO: 002 HI: 000 2 sent channels
LO: 003 HI: 000 3 sent channels
…
LO: 000 HI: 000 256 sent channels
LO: 001 HI: 001 257 sent channels
LO: 002 HI: 001 258 sent channels
…
LO: 255 HI: 001 511 sent channels
LO: 000 HI: 001 512 sent channels
Here are the formulas to calculate the channel count bytes:
LOWBYTE = channelcount AND 255
HIGHBYTE = INT((channelcount -1)/256)

$03F5 Send Repeat — typical value = 0
>0: send continously (repetitive)
0: one transmission, then stop

$03F6 Interdigit time — typical value = 0
Idle time between Bytes. Minimum 0s, max. 1s.
value = time (us) / 2 * 16 / 12

Return parameters:

$03F9 Card type
Returns the card type identifier.
Valid card IDs are:
128: Card 1512A, 1512B/LC
001: Card 1512B
004: Card 1512C
005: Card 1512C

NOTE FOR WINDOWS 2000 / XP USERS

The DLLs cannot be used with Windows NT, Widows 2000 or Windows XP. If you want to program an application for any of the OS versions, you may have to use the freeware tool DLPORTIO which supports card access with Windows 2000/XP. See the DRIVERS section of www.pcdmx512.com for more information and download.

2.3 Application programs

The CD contains several programs which xcan be run under Windows 3.1, 3.11, 95, 98 or Windows Millennium (the programs do not support Windows NT or 2000). To install and start these programs, please proceed as follows:

- start Windows
- place the CD ROM into your drive bay
- change to folder \DMX\1512B\WINDOWS
- Enter the command:
  RUN: A:\SETUP

This will call the Setup routine, which copies the required files to your harddisk and installs a program group "DESK36". There you will find these apllcation programs:

- **DESK12**
  The surface of a 12-channel 2-preset light control desk.
  You can create scenes and running lights.

- **DESK18**
  same as above, but 18 channels

- **DESK36**
  same as above, but 36 channels

- **PANEL**
  A touchpad with mouse control.

Please find instructions for use in the help files, which come with the programs.
Press <F1> to call the help screen.

IMPORTANT NOTICE:
These programs have been developed for DMX interface cards 1512A and 1512B-LC. To enable operation with the 1512B card, you have to copy the file SLHDMX16.BIN into the \DMX512 folder, or the card cannot be found and initalized.

If the installation program reports a "file already present" or "file already in use by another application", please select IGNORE and continue. Windows wants to replace files, which already have been installed on your system by another program. As these files are already present, it is okay to skip that step.

2.4 Third Party Software

There are several programs available, from lighting control to complete showmanagement, which support the DMX PC interface cards. Please refer to our support website at http://www.pcdmx.com for more details.

3 Programming Examples

3.1 Pascal routines in high level language

Listing of Unit DMX512.PAS

This unit conains the asembled operating system for the DMX PC card 1512B.
Important: All zero bytes must be transferred!

UNIT dmxbts;

```
interface
const b_sys : array[0..530] of byte = (
$02,$00,$7D,$02,$01,$BD,$32,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$20,$98,$06,$20,$99,
$0B,$32,$00,$32,$C2,$98,$02,$01,
$D1,$00,$00,$32,$C2,$99,$32,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$32,$00,$00,$00,$00,
$00,$00,$00,$00,$00,$75,$87,$00,
$75,$D0,$00,$75,$81,$60,$75,$B8,
$10,$D2,$08,$75,$98,$88,$75,$A8,
$80,$75,$89,$11,$75,$88,$01,$75,
$90,$67,$75,$B0,$FF,$75,$3A,$01,
$30,$95,$03,$75,$3A,$80,$90,$03,
$E8,$74,$CF,$F0,$A3,$74,$03,$F0,
$A3,$74,$0B,$F0,$A3,$74,$60,$F0,
$90,$03,$F0,$E0,$F5,$31,$A3,$E0,
$F5,$32,$A3,$E0,$F5,$33,$A3,$E0,
$F5,$34,$A3,$E0,$F5,$35,$A3,$E0,
$F5,$38,$A3,$E0,$C3,$13,$F5,$3B,
$A3,$A3,$E0,$F5,$44,$A3,$E5,$3A,
$F0,$A3,$E0,$F5,$3E,$A3,$E0,$F5,
$41,$A3,$E0,$F5,$42,$A3,$E5,$43,
$F0,$A3,$E5,$8A,$F0,$A3,$E5,$8C,
$F0,$75,$8A,$00,$75,$8C,$00,$C2,
$8D,$78,$80,$79,$80,$E5,$44,$B4,
$FF,$06,$75,$A8,$91,$02,$01,$0D,
$D2,$90,$C2,$92,$75,$A8,$80,$D5,
$31,$FD,$D2,$92,$90,$04,$00,$AB,
$35,$AA,$34,$85,$32,$99,$C2,$99,
$0B,$E5,$3B,$60,$2D,$E0,$AC,$3B,
$30,$99,$FD,$DC,$FE,$F5,$99,$C2,
$99,$A3,$DA,$F1,$DB,$EF,$30,$99,
$FD,$D5,$33,$FD,$C2,$8C,$E5,$38,
$60,$03,$02,$00,$A6,$90,$03,$FE,
$E5,$8A,$F0,$A3,$E5,$8C,$F0,$02,
$01,$57,$E0,$30,$99,$FD,$F5,$99,
$C2,$99,$A3,$DA,$F5,$DB,$F3,$30,
$99,$FD,$D5,$33,$FD,$C2,$8C,$E5,
$38,$60,$03,$02,$00,$A6,$90,$03,
```

```
     $FE,$E5,$8A,$F0,$A3,$E5,$8C,$F0,
     $02,$01,$80,$02,$01,$83,$75,$C7,
     $AA,$75,$C7,$55,$43,$D8,$01,$22,
     $50,$43,$2D,$31,$35,$31,$32,$42,
     $2F,$31,$36,$20,$28,$43,$29,$20,
     $44,$2E,$20,$48,$6F,$66,$66,$6D,
     $61,$6E,$6E,$20,$26,$20,$53,$4C,
     $48,$20,$31,$39,$39,$34,$2F,$39,
     $35,$20,$20,$20,$20,$D2,$9C,$C0,
     $E0,$85,$41,$3F,$85,$42,$40,$D2,
     $0A,$C2,$08,$90,$06,$00,$D0,$E0,
     $32,$C0,$D0,$C0,$E0,$20,$08,$14,
     $30,$0A,$18,$E5,$99,$F5,$43,$B4,
     $00,$0A,$C2,$0A,$90,$06,$00,$D0,
     $E0,$D0,$D0,$32,$D2,$08,$D0,$E0,
     $D0,$D0,$32,$20,$08,$16,$E5,$99,
     $F0,$A3,$D0,$E0,$D0,$D0,$32,$E5,
     $3F,$70,$02,$15,$40,$15,$3F,$D0,
     $E0,$D0,$D0,$32,$D0,$E0,$D0,$D0,
     $32,$00,$00);
     IMPLEMENTATION
     end.


     var d_sys:array[0..511] of byte;
     var cardadr:word;

     function find_card:word;
     procedure dmxinit;
     procedure cardstart;
     procedure dmxtrans(kanaladress:word; data:byte);
     procedure dmxtrans512;
     function dmxread(adresse:word):byte;

     Implementation
     function find_card ;
     {************************************
     Searches for the card address and returns
     the address if found successfully, or
     returns 0 if no card is found
      ************************************}
     var
       i, portreceive:byte;
       portadr:word;
       dummy:byte;

begin
      portadr:=0;
                                                                    {no card}
      for i:= 0 to 3 do
      begin
      dummy := port[$100+$20*i];                                  {card reset}
       port[$100+$20*i] := 255;
       port[$101+$20*i] := 3;                                {set address $03FF}
       port[$102+$20*i] := $AA;                                 {set test value}
      end;
```

```
for i:= 0 to 3 do
 begin
  port[$100+$20*i] := 255;
  port[$101+$20*i] := 3;
 portreceive:=port[$102+$20*i];
 If Portreceive = $AA then                          {check for test value}
  portadr:= ($100+$20*i);
 end;
find_card:= portadr;
end;


procedure dmxinit;
{************************************
  transfers the CPU operating system
 ************************************}
var
i :integer;
dummy:byte;

begin
 dummy := port[cardadr];
                                                          {reset card}
 for i:= 0 to 530 do
 begin                                          {transfer operating system}
  port[cardadr]   := lo(i);                              {address lowbyte}
  port[cardadr+1] := hi(i);                             {address highbyte}
  port[cardadr+2] := b_sys[i];                                {data value}
 end;

 for i:= $0 to $3ff do
 begin                                                 {Set all channels =0}
  port[cardadr]   := lo(i+$400);
  port[cardadr+1] := hi(i+$400);
  port[cardadr+2] := 0;
 end;
end;


procedure cardstart;
{************************************
 Starts the card DMX stransfer, removes
 CPU Reset.
 ************************************}
var
dummy : word;
begin
  dummy := port[cardadr];                                   {reset card}
  dummy := port[cardadr+3];                               {release reset}
end;
```

```
procedure dmxtrans(kanaladress:word;
data:byte);
{************************************
 ransfers a DMX data value into the
 appropriate location within the DMX RAM
 Example: Channel 3: -> RAMadsress $402
 ************************************}
begin
 port[cardadr]    := lo($400+kanaladress-1);
 port[cardadr+1]  := hi($400+kanaladress-1);
 port[cardadr+2]  := data;
end;

procedure dmxtrans512;
{************************************
 transfers 512 DMX values from array
 d_sys into the DMX-RAM, range 400h-5FF
 ************************************}
var i : word;
begin
 for i := 0 to $1ff do
  begin
   port[cardadr]    := lo($400+i);
   port[cardadr+1]  := hi($400+i);
   port[cardadr+2]  := d_sys[i];
end;


function dmxread(adresse : word) : byte;
{************************************
   Reads a byte from the DMX RAM
 ************************************}
begin
 port[cardadr]   := lo(adresse);
 port[cardadr+1] := hi(adresse);
 dmxread         := port[cardadr+2];
end;
end.
```

```
Program DMXDEMO;
{************************************
 Demo programm shows how to initialize
 the card, reading from and writing into
 the DMX RAM
 ************************************}
uses crt, dos, dmx512;
var i, data : byte;
          j : word;
begin
 cardadr := find_card;                          {find card base address}
 dmxinit;                                {copy operating system into RAM}
 cardstart;                                      {start the DMX card}
 clrscr;

 for j:= 0 to $5ff do
                                                 {read DMX RAM from 0-5FFh
                                                    and display on screen}
 begin
  data:=dmxread(j);
  writeln(j:4,'  ', data:4);
  delay(10);
 end;

 for i:= 0 to 255 do
                                               {dim channels 1- 4 slowly}
 begin
  dmxtrans(1,i);
  dmxtrans(2,i);
  dmxtrans(3,i);
  dmxtrans(4,i);
  dmxstart;
  delay(50);
  gotoxy(10,10);
  write('Dimmerwert :', i:3);
 end;

 dmxtrans(1,0);
                                                {switch off channels 1-4}
 dmxtrans(2,0);
 dmxtrans(3,0);
 dmxtrans(4,0);
 readln;
end.
```

3.2 Using a recognizing the cards within your own software

As could be seen above, the operating system file must be transferred into the card and must then be started. Thus card-internal functions could easily changed or expanded by changing or replacing the card OS file. Our applications come with different binary files, which support the following card types:

SLHDMX12.BIN	for cards 1512A or for cards 1512B/LC, equipped with a 12 MHz crystal.
SLHDMX16.BIN	for cards 1512B (professional series), equipped with a 16 MHz crystal.
SLHDMX17.BIN	for cards 1512C

This is how to initialize the card:

- **Determine the card address.** Please use a routine similar to find_card from chapter 3.1. Let's assume the found card address is $0100 (100h, 256 dec).

  (1) Reset the card by reading from address $0100
  (2) Write $FF into address $0100
  (3) Write $03 into address $0101 (0100 +1)
  This selects est address $03FF in the internal card RAM.
  (4) Write $AA into address $0102 (0100 +2)
  (5) Read address $0102 (0100+2)
  (6) If the return value is $AA, the card has been found.
  (7) If the return value is different, repeat steps (2)-(5) for addresses $0120, $0140, $0160

- **Transfer the card OS into the card, to make it functional. Assume to find a 1512B card.**
  (1) Reset the card by reading from address $0100
  (2) Open file SLHDMX17.BIN.
  Note: Close this file, before opening it. This ensures to get no "file open" error. The pointer will then point to the beginning of the file. Open the file as RANDOM or as BINARY, depending on your preferred programming language.

  The file is then transferred into the card. This is done byte by byte, null bytes must also be transferred.
  (3) Set the RAM address pointer = $0000
  (4) Write the LowByte of this RAM address into Card address +0
  (5) Write the HighByte of this RAM address into Card address +1
  (6) Read a Byte from file and write this byte into Card address +2
  (7) Increment RAM address and repeat steps (4) to (5) until EOF

  We have nom transferred the card OS into the card memory and are ready to start the card CPU and detect the card type.

  (8) Read card address +3, to start the card and wait at minimum 100 us before reading the result.
  (9) Write $F9 into card address +0
  (10) Write $03 into card address +1
  You have now selected return address $03F9 within the card RAM.
  (11) Read card address +2
  Return value $04: card 1512C present

Return value $05: card 1512C present
Return value $80: card 1512A or 1512B/LC (12 MHz card) present
Return value $01: card 1512B (16 MHz card) present
If a 1512B card has been detected, everything is okay.
If a 12 MHz standard card (1512A, 1512B-LC) has been detected, the complete initialization must be repeated. Use the file
SLHDMX12.BIN for card 1512A and 1512B/LC
SLHDMX17.BIN for card 1512C
Then check the return value again. The identifier must match; if you get a different return value please check the card, or the card OS transfer was faulty. Return codes $06 through $1F are reserved for future SLH products.
NOTE:
If a 1512C card is expected to be found, always check for this card type first. Then check for the presence of a 1512B or 1512A card, respectively.

- **Transfer the Default Parameters into the card.**
Most operating system files allow setting of card timing, channel count and more. If parameters are specified beyond allowable limits, the card may correct worng settings by itself.

IMPORTANT NOTICE:

When loading the operating system file SLHDMA16.BIN, all parameter values are set to factory defaults automatically. Thus normally no parameter transfer is required.

Memory allocation of PC DMX card 1512B:

$0000-$03EF	Operating system, max. 1000 Bytes
$03F0-$03FE	Operating system parameters, dmxdef()
$0400-$05FF	Send data for Link 1, Channels 1- 512

Card RAM

Address	Parameter	Default
$03F0	Start Sync Duration Length of sync pulse (RESET). Minimum value required for DMX512 is 88us. value = time (us) / 2 * 16 / 12	typical value = 60
$03F1	Startbyte value Value of send startbyte	typical value = 0
$03F3	channel count LowByte (0-255)	typical value = 0
$03F4	channel count HighByte (0-2)	typical value = 2

The bytes for the channel counter have to be set according to the following table:

LO: 001 HI: 000 1 sent channels
LO: 002 HI: 000 2 sent channels
LO: 003 HI: 000 3 sent channels
…
LO: 000 HI: 000 256 sent channels
LO: 001 HI: 001 257 sent channels
LO: 002 HI: 001 258 sent channels
…
LO: 255 HI: 001 511 sent channels
LO: 000 HI: 001 512 sent channels

Here are the formulas to calculate the channel count bytes:
LOWBYTE = channelcount AND 255
HIGHBYTE = INT((channelcount-1)/256)

$03F5	Send Repeat >0: send continously (repetitive) 0: one transmission, then stop	typical value = 0
$03F6	Interdigit time Idle time between Bytes. Minimum 0s, max. 1s. value = time (us) / 2 * 16 / 12	typical value = 0

Return parameters:

$03F9	Card type Returns the card type identifier. Valid card IDs are: 128: Card 1512A, 1512B/LC 001: Card 1512B 004: Card 1512B 005: Card 1512B It is recommended to set $03F9 to 0 before startting the card, and then read the result of the detection process	dmxdef(9) = 0

To transfer the default values into the card RAM:

(1) Set n=0
(2) Write $F0 + n into card address +0
(3) Write $03 into card address +1
(4) Write dmxdef(n) into card address +2
(5) n= n+1
(6) Repeat (2) until (5) while n<10

This completes the initialization of the interface card. Additionally, you may want to erase the DMX data RAM before operation to start from a blank buffer. To do so, simply write 0 to all data RAM l ocations.

NOTICE
The card address +1 register is a latch; values written to this register remain intact until overwritten. Unless this value changes it need not be re-written. This may save several accesses to the card. The examples above always write all registers for clarity.

Using the card

After initialization and setting of the OS parameters (if required) the card is ready to go.
Start the card by issuing the command

(11) Read card address +3: start card CPU (releases RESET command)

To transfer data into the card or retrieving data from the card, we recommend to install a timer performing these tasks exery xx milliseconds (typ. 50ms):

(1) Write LowByte(DMX-channel) to card address +0
(2) Write HiByte(DMX-channel) OR $04 to card address +1
(3) Write DMX data to card address +2
(4) Repeat (1) through (3) for all DMX channels needing an update

The update rate possible and thus the timer setting depends on the total channel count. If fewer channels (slots) are sent, the time interval may be set to smaller values. We do not recommend to use timer intervals below 10ms, as this might pose problems to Windows, or to the attached DMX devices (not to the interface card!).

The total DMX frame duration can be calculated from:

Start byte duration	typ. 88 us	=	88
+ Mark-after-Break	typ. 8 us	=	8
+ number of channels * 44 us	typ. 512*44	=	22528
+ number of channels * Interdigit time	typ. 512*0	=	0
+ Mark before Break time	typ. 100 us	=	100
	total us		22724

4. Looking ahead

No software will ever be ready. The same applies to DMX PC interfaces, and we are busy developing new stuff. If you would like to comment on this, we always welcome your opinion. Please mailto:

info@pcdmx.com

Please keep in mind for your own implementations:
The USITT DMX512/1990 protocol describes 512 channels, but does not define the need to transfer all 512 slots. Transferring lesser channels will enable you to program much higher repeat rates than the 44 Hz originally defined for DMX512. The card will support this, simply by setting other parameters. Feel free to experiment while you go.

The enclosed SLH CD-ROM #2 is filled with various examples for our PC interface cards. Since all cards are to be programmed similarly, many examples can easily be transferrd to another card type.

New user programs, additional utilities and Visual Basic programming lessons (german language for explanations!) can be found on our internet domain,

http://www.soundlight.de

Additionally, we have set up a special support site exclusively dedicated to PC DMX interfaces. This site also lists third party software, which is available for the interfaces.
The URL is:

http://www.pcdmx512.com

and is available in english language also.