SOUNDLIGHT
Lichtsteueranlagen

User-Manual

DMX 512 PC Interface Card 1512B Mk1.2

Nutzungsvertrag

Bevor Sie dieses Produkt einsetzen, lesen Sie bitte die folgende Lizenzvereinbarung vollständig durch. Sollten Sie nicht mit allen darin genannten Bedingungen einverstanden sein, so senden Sie es mit allen Unterlagen an den Lieferanten zurück. Nur dann gilt der Kauf als nichtig. Mit der Installation der Hardware und/oder dem ersten Aufruf eines der mitgelieferten Programmfiles gilt der Vertrag als geschlossen.

Nutzungsrecht

Das Recht der Vervielfältigung der Programme (auch Teile der Programme), der Hardware und des Handbuches bleiben bei SOUNDLIGHT und Donald Hoffmann.

Das Nutzungsrecht entspricht dem eines Buches:
Das Produkt kann nacheinander an unterschiedlichen Orten betrieben werden, jedoch nie an mehreren Rechnern oder an verschiedenen Orten gleichzeitig. Eine Abänderung der Hardware und der Software ist nicht gestattet.

CE

Die DMX-Interfacekarte 1512B ist in Verbindung mit einem Computer Gateway 2000 P5-90 (Pentium) CE-geprüft.

SOUNDLIGHT Lichtsteueranlagen

Gewährleistung

SOUNDLIGHT gewährleistet den einwandfreien Zustand des gelieferten Materials. Defekte Lieferungen werden ersetzt, wenn innerhalb von 10 Tagen nach Kauf berechtigte Gewährleistungsansprüche beim Händler geltend gemacht werden.

Haftung

Der Käufer hat sich von der Eignung des Materials für den von Ihm vorgesehenen Einsatzzweck zu überzeugen. Eine Eignung des Vertragsgegenstandes für einen bestimmten Verwendungszweck wird nicht zugesichert. Die Auslieferung von Software erfolgt "as is". Eine Haftung für Schäden oder Folgeschäden, gleich welcher Art, ist ausgeschlossen.

Bei Fragen oder bei Verbesserungsvorschlägen wenden Sie sich bitte an:

SOUNDLIGHT
Lichtsteueranlagen
Vahrenwalder Straße 205-207
D-30165 Hannover
Tel: 0511-3730-267
Fax: 0511-3730-423

Einleitung

DMX-512 ist, was die Steuerung in der Bühnenlichttechnik anbelangt, zu einem festen Steuerungsstamndard avanciert. Das Protokoll benutzt die PC-übliche RS-485 Schnittstelle zur Datenübertragung und beschreibt das Verfahren zur Übermittlung von Steuerinformationen für bis zu 512 zu steuernde Kanäle. Der Übertragungsstandard DMX-512 wurde durch das USITT (United States Institute for Theatre Technology, New York) beschrieben und dokumentiert. Eine Kopie des Standards ist beim Verband für professionelle Licht- und Tontechnik e.V., Hannover, erhältlich.

Die Vorteile der digitalen Steuerung sind die damit erreichbare Präzision, die Geschwindigkeit und die einfache Verkabelung der angeschlossenen Geräte durch die Verwendung einer Busstruktur.

Genau diese Vorteile bieten Ihnen unsere DMX-512 Interfacekarten 1512A und 1512B. Sie sind in der Lage, 512 Lichtkanäle ca. 44mal pro Sekunde anzusteuern (das entspricht einer Datenübertragungsrate von 250.000 Baud).

Diese Interfacekarte ist so konzipiert, daß sie für den Einsatz in IBM-kompatiblen PC's mit ISA-Bus geeignet ist. Dabei ist es prinzipiell unerheblich, ob Sie einen PC mit einer 8088, 8086, 80286, 80386, 80486 oder Pentium-CPU besitzen. Selbstverständlich ist für komplexe Anwendersoftware ein schnellerer Rechner von Vorteil.

Bei der Interfacekarte handelt es sich um eine sogenannte intelligente Karte, d.h., sie besitzt einen eigenen Mikroprozessor und ein eigenes Betriebssystem. Sie kann daher die DMX-Datenausgabe vornehmen, ohne daß dafür die CPU des PC beansprucht wird.

Zum Einsatz der DMX-Schnittstelle lesen Sie bitte die folgenden Kapitel, die Ihnen sowohl die Installation der Hardware, als auch den Gebrauch der Software zeigen werden.

Inhaltsverzeichnis

1 Die Hardware

1.1 Installation der DMX512-Karte

1.2 Die Herstellung eines DMX-Adapterkabels

1.3 Die Adressen der DMX-Karte

1.4 Funktionsbeschreibung der Karte

2 Die Software

2.1 Allgemeines

2.2 Das Objektfile fur C

2.3 Das Objektfile fur PASCAL

2.4 Die DLL für Windows

2.5 Das Assembler-Betriebssystem

2.6 Anwenderprogramme

3 Programmbeispiele

3.1 Programmbeispiele in C

3.2 Programmbeispiele in Pascal

3.3 Pascal-Funktionen in Hochsprache

3.4 Starten und Erkennen der Karte mit eigenen Programmen

4 Bemerkungen und Ausblicke

SOUNDLIGHT Lichtsteueranlagen

1.1 Die Installation der DMX-512-Karte

Die folgenden Schritte dienen dem Einbau der DMX512-Karte

1. Netzstecker des Computers ziehen !!!!!
2. Gehäuse öffnen
3. Adapterschraube aus einem freien Steckplatz entfernen
4. DMX-Karte in diesen freien Slot vorsichtig einsetzen
5. Adapterschraube wieder anbringen
6. Gehäuse des Computers wieder schließen und den Stecker des DMX-Kabels in die Buchse der DMX-Karte einstecken

Ein wichtiger Hinweis

Bitte beachten Sie, daS beim Einstecken bzw. Herausnehmen von Interfacekarten der Computer ausgeschaltet ist. Gleiches gilt fur das Umstecken von Jumpern oder das Anschließen bzw. Abziehen von Steckern.

Sollten Ihnen die Installation der Hardwarekarte nicht 100%ig geläufig sein, so lassen Sie einen Fachmann die Karte installieren.

1.2 Die Herstellung eines DMX-Adapterkabels

Die DMX-Karte 1512A verfügt über zwei getrennte, jedoch parallel angesteuerte Ausgangstreiber, die Karte 1512B verfügt über einen opto-isolierten Ausgang und einen Ausgangstreiber, der zwei Ausgangsanschlüsse parallel bedient.
Das Signal kann an bei beiden Karten an jeweils zwei Punkten der 9-poligen Sub-D-Anschlußleiste abgegriffen werden.

Die USITT-Norm DMX-512 schreibt als Standardsteckverbindung einen 5-poligen XLR-Steckverbinder vor. Seine Belegung ist wie folgt:

Pin 1: Masse
Pin 2: DMX -
Pin 3: DMX +
Pin 4: Reserve (DMX - 2. Link)
Pin 5: Reserve (DMX + 2. Link)

Zur Herstellung eines Adapterkabels nehmen Sie bitte die folgenden Verbindungen vor:

```
COMPUTER              DMX-Equipment
Sub-D 9polig          XLR 5-polig

1 <-------------->  3     DMX +
2 <-------------->  2     DMX -
3 <-------------->  5     Reserve-Link
4 <-------------->  4     Reserve-Link
5 frei
6 <-------------->  1     Abschirmung,
                                    Masse
7-9 frei
```

Die Leitungen für die 2. Verbindung (Reserve-Link) werden in den meisten Geräten nicht benötigt und können daher auch weggelassen werden.

1.3 Die Adressen der DMX-Karte

Damit der Computer die Interface-Karte findet bzw. ansprechen kann, benötigt jede Interface-Karte eine oder mehrere Adressen. Die Adreßzuweisung erfolgt, je nach Kartentyp, entweder durch Jumper oder durch DIP-Schalter. Bei der Einstellung von Jumpern ist es wichtig, stets beide Jumper zu setzen!

Mögliche Adressen der DMX-Karte sind:

Hex	Dezimal	Jumperstellung
$0100	256	J1 = rechts J2 = rechts
$0120	288	J1 = rechts J2 = links
$0140	320	J1 = links J2 = rechts
$0160	355	J1 = links J2 = links

Es werden jeweils 4 Adressen, beginnend mit der angegebenen Basisadresse, benötigt (z.B. $100...$103).

Die Basisadresse der DMX-Karte ist bei Auslieferung herstellerseitig mittels der Jumper J1 und J2 auf $160..$163 (352..355) eingestellt.

In den meisten Fällen wird die DMX-Karte mit der Basisadresse $0160 fehlerfrei laufen. Es ist allerdings nicht auszuschließen, daß andere Interfacekarten dieselbe Basisadresse benutzen. Sollte es beim Betrieb der DMX-Karte zu Störungen kommen, so versuchen sie die Interfacekarte mit einer anderen Basisadresse in Betrieb zu nehmen.

Die Software läuft auf allen diesen Basisadressen, und erkennt die Basisadresse selbständig durch die Routine DMXCARD().

Bei Karten, die mit einem DIP-Schalter ausgestattet sind, erfolgt die Einstellung der Kartenadresse durch die DIP-Schalter 1 und 2. Hier gelten folgende Zuordnungen:

Hex	Dezimal	Jumperstellung	
$0100	256	SW1 = ON SW2 = ON	
$0120	288	SW1 = OFF SW2 = ON	
$0140	320	SW1 = ON SW2 = OFF	
$0160	352	SW1 = OFF SW2 = OFF	

Die Schalter SW3 und SW4 sind werkseingestellt und sollten nicht verstellt werden.

DIP-Schalter 3 wird entsprechend der vorhandenen Prozessortaktfrequenz der Steckkarte gesetzt.
Wenn Ihre Karte mit einem 12 MHz-Quartz ausgestattet ist, setzen Sie Schalter 3 of OFF, wenn die Karte jedoch mit einem 16 MHz Quartz ausgestattet ist, setzen Sie hingegen Schalter 3 auf ON.

Einstellung für 12 MHz CPU-Clock:

SW3 = OFF

Einstellung für 16 MHz CPU-Clock:

SW3 = ON

Die Einstellung wird in der Messagebox unter "Karte initialisieren" ausgegeben. In der Kopfzeile wird die CPU-Frequenz angezeigt.

Die folgenden Kapitel beschäftigen sich mit der Programmierung der Karte und erläutern Beispiele für verschiedene Programmiersprachen. Wenn Sie zunächst auf die Programmierung der Karte verzichten wollen, lesen Sie nunmehr bitte im Kapitel 2.6 ANWENDERPROGRAMME weiter.

1.4 Die Funktion der Karte

Die DMX-512-Interfacekarte sollte zwei Eigenschaften besitzen: Zum einen sollte sie preiswert sein, zum anderen sollte sie aber auch so flexibel wie möglich sein, um jederzeit an spezielle Anforderungen und Veränderungen des DMX-Standards angepaßt werden zu können.

Beim Betrachten der Karte werden zwei Merkmale auffallen: erstens, daß diese Interfacekarte einen Mikroprozessor 80(C)31 enthält, und zweitens, daß ein ROM bzw. ein EPROM fehlt, das das Programm für die CPU enthalten könnte. Der Grund für das Fehlen eines Festwertspeichers (EPROM) ist die o.g Flexibilität. Durch den Einsatz eines RAMs als Speicher sowohl für die Betriebssoftware als auch für die Daten ist es möglich, Softwareänderungen ohne den Ausbau der Interfacekarte vorzunehmen. Auch eine Änderung des Betriebssystems während des Programmlaufs ist ohne weiteres möglich. Die vorgestellte Lösung hat allerdings den Nachteil, daß das Betriebssystem zu Beginn jedesmal in das DMX-RAM übertragen werden muß. Dazu wird eine spezielle Initialisierungsroutine verwendet.

Die bestehende Software geht von der folgenden RAM-Aufteilung aus:

$0000 - $03FF	Betriebssystem des 8032
$0400 - $05FF	DMX-Daten
$0600 - $1FFF	Bereich für eigene DMX-Daten

Um eine maximale Kompatibilität zu allem, was sich als IBM-PC kompatibel bezeichnet, zu gewährleisten, wurde eine 8-Bit-Busbreite gewählt. Das bedeutet keine Einschränkung, da alle DMX-Daten ohnedies lediglich 8-Bit-Daten darstellen. Die Karte wurde so ausgelegt, daß sie mittels I/O-Adressen im Bereich $0100h - $0160 zu adressieren ist. Dabei kann die Karte auf vier Adressen gelegt werden, nämlich $0100, $0120, $0140 und $0160.

Grundsätzlich muß man bei der Karte zwischen zwei Datenübertragungsarten unterscheiden:

- vom PC in das DMX-RAM
- vom DMX-RAM zum RS485-Ausgang

Aus der Sicht des PCs werden die zu übertragenden Daten einfach in das DMX-RAM ab der Adresse $0400 übertragen. Dabei liegen die Daten im RAM in der Reihenfolge, wie sie dann auch an die Geräte weitergeleitet werden (Kanal 1 -> $0400, Kanal 2 -> $0401 ... Kanal 512 -> $05FF).

Danach ist die Übertragung aus Sicht des PC abgeschlossen, er kann sich um andere Aufgaben kümmern. Es beginnt die Arbeit des Karten-Prozessors. Dieser holt sich die Daten aus dem DMX-RAM und transportiert diese dann seriell zum RS-422/RS-485 Leitungstreiber (siehe Flußdiagramm).

Damit sich nicht beide Prozessoren beim Zugriff auf das gemeinsame RAM behindern, wird der 80C31 für die Zeit, in der der PC die Daten in das RAM schreibt, angehalten, und muß nach Beendigung der Übertragung wieder gestartet werden.

Wurde die DMX-Übertragung des 80C31 einmal gestartet, so werden die DMX-Daten solange immer wieder übertragen, bis entweder der PC neue Daten ins RAM schreibt, oder die Übertragung per Befehl gestoppt wird. Da es nach USITT nicht unbedingt erforderlich ist, alle 512 Kanäle komplett zu senden, ist ein Unterbrechen der Übertragung während des RAM-Updates durchaus zulässig.

Bei Programmierung unter komplexen Oberflächen oder bei Einsatz einer Timer-Funktion mag es sinnvoll sein, das RAM-Update und das Senden eines neuen DMX-String miteinander zu synchronieren. Das erfolgt einfach dadurch, daß die Karten-CPU nach Durchlaufen einer kompletten DMX-Sendung durch ein BREAK-Kommando oder eine Endlosschleife angehalten wird.

2 Die Software

Die zu der DMX-Karte dazugehörige Software besteht aus mehreren Teilen:

- lauffähige Anwenderprogramme unter MS-Windows
- ausdokumentierte Sources
- Beispiele und Objektfiles für verschiedene Programmiersprachen:
  - linkbares Objektfile fur die Programmiersprache C
  - linkbares Objektffle fur die Programmiersprache Pascal
  - das Betriebssystem fur die DMX-Karte
  - Beispielprogramme in BASIC, Pascal und C
  - eine Dynamic Link Library (DLL) für Windows

Die Objektfiles und die DLL sind in Maschinensprache geschrieben, um die Übertragungsgeschwindigkeit zwischen dem PC und der DMX-Karte zu maximieren.

Zu Pascal:
Alle Pascal-Beispieldateien auf der beiliegenden Diskette wurden in Turbo Pascal 6.0 programmiert. Bei Verwendung anderer Compiler ist es wichtig, daß die externen Prozeduren als **far** gekennzeichnet werden.

Zu C:
Alle C-Beispieldateien sind auf der Diskette nur als Quellcode vorhanden und müssen von Anwender gegebenenfalls selbst compiliert werden. Die Syntax der Programme bezieht sich auf Turbo C++ 3.0.

Zu Windows:
Das Programm DESK6 ist im Quellcode vorhanden und muß mit Visual Basic compiliert werden. Die Funktionen der DLL sind unter Windows 3.1, Windows 3.11 und Windows 95 verfügbar.

2.2 Das Objektfile für C

Auf der Diskette befindet sich im Directory C ein File mit dem Namen DMXLIB.OBJ . Hierbei handelt es sich um ein linkbares (einzubindendes) Maschinenprogramm. Dieses File enthalt die folgenden DMX-Befehle:

`DMXCARD( )` Dieser Befehl sucht im Computer nach der Adresse der DMX-Karte und speichert die gefundene Adresse in der Variablen Cardadr. Der Aufruf erfolgt ohne Parameter und schreibt die Kartenadresse in die globale Variable Cardadr. Dieser Befehl muß stets als erster ausgeführt werten.

`DMXINIT( )` dient zur Initialisierung der DMX-Karte. Das Betriebssystem wird in die Karte geladen. Der Aufruf erfolgt ohne Parameter.

`DMXSTART( )` startet die Ausgabe der DMX-Karte. Die Ausgabe bleibt solange aktiv, bis ein anderer DMX-Befehl ausgeführt wird. Der Aufruf erfolgt ohne Parameter.

`DMXSTOP( )` beendet die Ausgabe der DMX-Karte. Der Aufruf erfolgt ohne Parameter.

`DMXCHANNEL(Kanal, Wert)`
überträgt einen DMX-Wert für einen DMX-Kanal in den Speicher der DMX-Karte. Übergibt zwei Parameter, Wert (8-bit) und Kanal (16-bit).

`DMXCHANNEL_S(Kanal, Wert)`
überträgt einen DMX-Wert fur einen DMX-Kanal in den Speicher der DMX-Karte und startet die DMX-Ausgabe. Übergibt zwei Parameter, Wert (8-bit) und Kanal (16-bit).

`DMXTRANS(Feldname[], Anzahl)`
überträgt eine bestimmte Anzahl von DMX-Werten aus einem Array, beginnend bei Kanal 1. Übergibt zwei Parameter, Anzahl (16-bit) und Feldname[] (Adresspointer).

`DMXTRANS_S(Feldname[], Anzahl)`
überträgt eine bestimmte Anzahl von DMX-Werten aus einem Array beginnend bei Kanal 1 und startet anschließend die Ausgabe. Übergibt zwei Parameter, Anzahl (16-bit) und Feldname[] (Adresspointer).

2.3 Das Objektfile für PASCAL

Auf der Diskette befindet sich im Directory PASCAL ein File mit dem Namen DMXLIB.OBJ. Hierbei handelt es sich um ein linkbares (einzubindendes) Maschinenprogramm. Dieses File enthalt die folgenden DMX-Befehle:

`DMXCARD()` der Aufruf erfolgt ohne Parameter und schreibt die Kartenadresse in die globale Variable cardadr.

`DMXINIT()` der Aufruf erfolgt ohne Parameter und initialisiert die DMX-Karte.

`DMXSTART()` der Aufruf erfolgt ohne Parameter und startet die DMX-Ausgabe.

`DMXSTOP()` der Aufruf erfolgt ohne Parameter und beendet die DMX-Ausgabe.

`DMXCHANNEL(Kanal, Wert)`
übergibt zwei Parameter, Kanal (16-bit) und Wert (8-bit).

`DMXCHANNEL_S(Kanal, Wert)`
übergibt zwei Parameter, Kanal (16-bit) und Wert (8-bit) und startet die DMX-Ausgabe.

`DMXTRANS(Feldname[], Anzahl)`
übergibt zwei Parameter Feldname[] (Adresspointer) und Anzahl (16-bit).

`DMXTRANS_S(Feldname[], Anzahl)`
übergibt zwei Parameter Feldname[] (Adresspointer) und Anzahl (16-bit) und startet die DMX-Ausgabe.

2.4 Die DLL für Windows

Das Windows-Verzeichnis enthält eine Dynamic Link Library, die Ihnen alle benötigten Funktionen zur Kommunikation mit der Karte zur Verfügung stellt.

Name:	vbCardAdr
Zweck:	Adresse der DMX-Karte im Rechner ermitteln
Typ:	Funktion
Übergabeparameter:	dummy% (Byte)
Rückgabeparameter:	CardAdr% (Word)

Beispiel:
Declare Function vbCardadr% Lib "SLHDMX1.DLL" (ByVal w%)

Name:	vbWByte
Zweck:	Schreibt ein Byte in die DMX-Karte
Typ:	Sub
Übergabeparameter:	CardAdr% (Word)
	ByteAdr% (Word)
	Ausgabe% (Byte)
Rückgabeparameter:	-

Beispiel:
Declare Sub vbWByte Lib "SLHDMX1.DLL" (ByVal CardAdr%, ByVal ByteAdr%, ByVal Ausgabe%)

Name:	vbDMXStart
Zweck:	Startet die DMX-Übertragung der Karte
Typ:	Funktion
Übergabeparameter:	CardAdr% (Word)
Rückgabeparameter:	Dummy% (Byte)

Beispiel:
Declare Function vbdmxstart% Lib "SLHDMX1.DLL" (ByVal CardAdr%)

Name:	vbDMXStop
Zweck:	Stoppt die DMX-Übertragung der Karte
Typ:	Funktion
Übergabeparameter:	CardAdr% (Word)
Rückgabeparameter:	Dummy% (Byte)

Beispiel:
Declare Function vbDMXStop% Lib "SLHDMX1.DLL" (ByVal Byte%)

Die Bezeichnung DUMMY% bezeichnet einen Wert ohne Bedeutung.

Vergleichen Sie bitte hierzu den Quellcode der Applikation "DESK6", der die Einbindung der DLL zeigt. Zum Schreiben in die Karte ist die Kenntnis der RAM-Belegung innerhalb der Karte wichtig. Außerdem muß, damit die Karte gestartet werden kann, das Betriebssystem der Karte in den Kartenspeicher übertragen werden. Das Kartenbetriebssystem finden Sie als erweiterte Form des unter 2.5 erläuterten Betriebsprogrammes als assemblierte Datei SLHDMX.BIN ebenfalls im Windows-Verzeichnis vor.

Speicherbelegungsplan der DMX-Karte:

$0000-$03EF	Betriebssystem, max. 1000 Bytes
$03F0-$03FF	Betriebssystemparameter
$0400-$05FF	Sendedaten für 512 Kanäle

Betriebssystemparameter:

$03F0 Start-Sync-Dauer * 2 us
typischer Wert = 44

$03F1 StartByte-Wert
typischer Wert = 0

$03F2 Ende-Leerlaufzeit * 2 us + 40 us
typischer Wert = 30

$03F3 Kanalzähler LowByte (0-255)
typischer Wert = 100

$03F4 Kanalzähler HighByte (0-2)
typischer Wert = 0

Die folgenden Bytes für den Kanalzähler sind gemäß folgender Vorschrift zu setzen:

LO: 001 HI: 000 1 gesendeter Kanal
LO: 002 HI: 000 2 gesendete Kanäle
LO: 003 HI: 000 3 gesendete Kanäle
...
LO: 000 HI: 000 256 gesendete Kanäle
LO: 001 HI: 001 257 gesendete Kanäle
LO: 002 HI: 001 258 gesendete Kanäle
...
LO: 255 HI: 001 511 gesendete Kanäle
LO: 000 HI: 001 512 gesendete Kanäle

$03F5 Sendewiederholung
0: einmal senden >0: endlos senden
typischer Wert = 0

$03F6 Interdigit-Zeit, Pause zwischen Bytes 0...44 us
typischer Wert = 0

2.5 Das Assembler-Betriebssystem

Betrachtet man das DMX-Timing (siehe USITT-Standard "DMX-512"), so ist leicht erkennbar, daß sich alle typischen Timingwerte auf ein Vielfaches einer Mikrosekunde beziehen. Es lag daher nahe, einen Mikroprozessor zu verwenden, der eine Befehlzykluszeit von genau einer Mikrosekunde besitzt. Durch die einfache Programmierbarkeit fiel die Wahl schließlich auf den 80C31. Hierbei wird der serielle Datenstrom wird auf einen Port (Port 1.2) und von dort aus zum Linedriver AM26LS31 bzw. MAX485 geführt. Dieses Verfahren hat zwar den Nachteil, daß man in den Assemblerroutinen alle Befehlzykluszeiten mit berücksichtigen muß, hat aber den Vorteil, daß gerade in der Entwicklungsphase jede Art des Timings explizit getestet werden kann.

Assemblerlisting:

Das nachfolgende Listing gilt nur für Karten mit einem 12 MHz Quartz und ist in erweiterter Form in der Datei SLHDMX12.BIN enthalten.

```
$PROCESSOR(8032)
$NOPAGING
;
;****************************************
;  DMX-Betriebssystem für 80C31 Processor
;  (C) Donald Hoffmann
;  Programmversion V1.1 vom 31.12.92
;  überträgt 512 Kanäle
;****************************************
; zuerst wird 88us low gesendet
; dann 8us high
; dann startbit (low) 4 us
; dann Daten       8 * 4 us
; dann Stopbits    2 * 4 us

     ORG $0000

loop: MOV R2,#2    ;2x 256 Bytes senden
     MOV DPTR, #$400 ;Start Daten: $400

     MOV R0, #23    ;Startsync = 88us
     CLR P1.2       ;Ausgang low
loop1: NOP
     NOP
     DJNZ R0, loop1

     MOV A, #0      ;sende Startbyte "0"
     CLR P1.2       ;sende Startbit
     NOP
     RRC A
     MOV P1.2, C
     NOP
     RRC A
     MOV P1.2,C
     NOP
     RRC A
     MOV P1.2,C
     NOP
     RRC A
     MOV P1.2,C
     NOP
     RRC A
```

```
      MOV P1.2,C
      NOP
      RRC A
      MOV P1.2,C
      NOP
      RRC A
      MOV P1.2,C
      NOP
      RRC A
      MOV P1.2,C
      NOP
      NOP
      NOP
      SETB P1.2   ;2 Stopbit
      NOP
      NOP
      NOP
      MOP
      NOP
      NOP
      NOP
      NOP
send10:
      MOV R1, #0
send11:
      MOVX A, @DPTR ;hole Datenbyte
      CLR P1.2    ;Startbit
      NOP
      RRC A
bit0: MOV P1.2,C
      NOP
      RRC A
bit1: MOV P1.2,C
      NOP
      RRC A
bit2: MOV P1.2,C
      NOP
      RRC A
bit3: MOV P1.2,C
      NOP
      RRC A
bit4: MOV P1.2,C
      NOP
      RRC A
bit5: MOV P1.2,C
      NOP
      RRC A
bit6: MOV p1.2,C
      NOP
      RRC A
bit7: MOV P1.2,C
      NOP
      NOP
      NOP
stop: SETB P1.2   ;2x Stopbit
      NOP
      NOP
```

```
      NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      INC DPTR
      DJNZ  R1, send11
                  ;Kanäle 1-256 fertig?
      MOV DPTR, #$500
      DJNZ  R2, send10
                  ;Kanäle 257-512 fertig?

      LJMP loop   ;von vorn
end
```

Möchte man nur einen Sendedurchlauf oder -wie unter Kapitel 2.4 gezeigt- die Sendewiederholung schaltbar machen, dann ersetze man das letzte LJMP-Statement durch folgende Anweisungen:

```
      MOV DPTR, #$305
                  ;Repeat-Variable
      MOVX A, @DPTR
      JZ $        ;bei Null: endlos
      LJMP loop   ;sonst von vorn

end
```

2.6 Anwenderprogramme

Im Lieferumfang der Karte ist eine CD mit verschiedenen Anwenderprogrammen enthalten, die unter Windows 3.1, Windows 3.11 und Windows 95 lauffähig sind. Um diese Programme zu installieren, gehen Sie bitte wie folgt vor:

- Starten Sie Windows
- Legen Sie die CD in Ihr CD-ROM Laufwerk
- Wechseln Sie in das Verzeichnis/Ordner "DMX", dann in das Unterverzeichnis "DESK", "SCROLLER" etc.
- Starten Sie das jeweilige Installationsprogramm mit dem Befehl DATEI AUSFÜHREN: SETUP.EXE

Es wird dann das Installationsprogramm gestartet. Es kopiert die benötigten Dateien auf die Festplatte und richtet eine Programmgruppe "DMX-512" ein. In dieser Programmgruppe finden Sie drei Anwenderprogramme:

- DESK12 / DESK18
  Die Oberfläche eines 12- bzw. 18-kanaligen 2-Preset-Lichtregiepultes. Es können Stimmungen erstellt und gespeichert werden, sowie Lauflichter programmiert werden.

- DMXTG
  Die Oberfläche zeigt einen DMX-Testgenerator, mit dem die Länge und die Timing-Einstellungen des gesendeten DMX-Signals variiert werden können. Damit lassen sich Empfänger sehr einfach auf Funktion überprüfen.

- SCROLLER
  Die Oberfläche stellt eine Steuerung für DMX-kompatible Rollenfarbwechsler dar. Strings können in Farbe, Länge und Anordnung frei definiert werden, Szenen lassen sich speichern und abrufen.

Die Bedienungsanleitungen zu allen Programmen entnehmen Sie bitte den jweils zugehörigen Hilfstexten, die Sie jeweils über das Menü "HILFE" oder durch die Taste <F1> erreichen können.

Die Installation kann unter Windows 3.1, Windows 95 oder Windows 98 erfolgen. Sollte während der Installation die Meldung "Datei ist bereits vorhanden" oder "Datei ist in Benutzung" erscheinen, versucht das Setup-Programm, eine Datei auf Ihrem Rechner zu installieren, die dort schon vorhanden ist. Wählen Sie in diesem Fall "übergehen/ignorieren" bzw. "weitermachen" und setzen Sie die Installation fort.

3. Programmbeispiele

3.1 Programmbeispiele in C

1. Beispiel
Zeigt den Gebrauch des Befehls DMXCARD

```
#include <stdio.h>
int cardadr;
extern far dmxcard();

main()
{
  dmxcard();
  printf("die Kartenadresse ist: %x", cardadr);
  getch();
  return(0);
}
```

2. Beispiel
Zeigt das Übertragen von DMX-Werten auf die Kanäle 1-4
Die Kanäle 1-4 werden 10 mal langsam aufgedimmt.

```
#include <stdio h>
typedef unsigned char byte;
byte kanal,wert,k;
int cardadr,i;
extern far dmxcard();
extern far dmxinit();
extern far dmxchannel(byte kanal, byte wert);
extern far dmxstart();
main()
{
  k = 0;
  dmxcard();
  dmxinit();

  for(i=0; i<2550;i++)
  {
    dmxchannel(1,k);
    dmxchannel(2,k);
    dmxchannel(3,k);
    dmxchannel(4,k);
    dmxstart();
    k++;
    delay(100);                          /* Verzögere 100 msec */
  }
  return(0);
}
```

3. Beispiel

Mit dem Befehl dmxchannel(x,y) wird ein Kanalwert an die DMX-Karte übergeben. Hierbei erfolgt noch keine DMX-Ausgabe. Erst der Befehl dmxstart() sendet die Werte zu den entsprechenden DMX-Geräten.

```
#include <stdio.h>
typedef unsigned char byte;
byte kanal,wert;
int cardadr;
extern far dmxcard();
extern far dmxinit();
extern far dmxstart();
extern far dmxstop();

extern far dmxchannel(byte kanal,byte wert);
main()
{
  dmxcard();
                                        /* finde Adresse der DMX-Karte */
  printf("die Kartenadresse ist %x",cardadr);
  dmxinit();
                                        /* initialisiere DMX-Karte */
  dmxchannel(1,0);
                                        /* Übertrage Wert=0 in Kanal 1 */
  dmxchannel(2,160);
                                        /* Übertrage Wert=160 in Kanal 2 */
  dmxchannel(3,80);
                                        /* Übertrage Wert=80 in Kanal 3 */
  dmxchannel(4,0);
                                        /* Übertrage Wert=0 in Kanal 4 */
  dmxstart();
                                        /* Starte die DMX-Ausgabe */
  return(0);
}
```

4. Beispiel
mit dem Befehl dmxchannel_s(x,y) wird ein Kanalwert an die DMX-Karte übergeben und es erfolgt sofort die Ausgabe.

```
#include <stdio.h>
#include <dos.h>
typedef unsigned char byte;
byte kanal,wert;
int cardadr;
extern far dmxcard();
extern far dmxinit();
extern far dmxstart();
extern far dmxstop();
extern far dmxchannel_s(byte kanal,byte wert);
main()
{
  dmxcard();                          /* finde Adresse der DMX-Karte */

  dmxinit();                          /* initialisiere DMX-Karte */

  dmxchannel(1,0);                    /* Übertrage Wert=0 in Kanal 1 */

  dmxchannel(2,160);                  /* Übertrage Wert=160 in Kanal 2 */

  dmxchannel(3,80);                   /* Übertrage Wert=80 in Kanal 3 */

  dmxchannel_s(4,0);                  /* Übertrage Wert=0 in Kanal 4 */


                                      /* und starte die Ausgabe */

  delay(100);
  dmxstop();
  return(0);
}
```

5. Beispiel
Mit dem Befehl dmxtrans_s(x,y) werden 100 Kanäle langsam aufgedimmt

```
#include <stdio.h>
#include <dos.h>
typedef unsigned char byte;
byte kanal,wert;
int cardadr;
extern far dmxcard();
extern far dmxinit();
extern far dmxstart();
extern far dmxstop();
extern far dmxtrans_s(int dmx[],byte wert);
main()
{
  dmxcard();                                    /* finde Adresse der DMX-Karte */
  dmxinit();                                    /* initialisiere DMX-Karte */
  for(j=0;j<=255;j++)
  {
  for(i=1;i<=100;i++)
    {
    dmx[i] = i;                                 /* Übertrage und starte */
    dmxtrans_s(dmx,100);
    delay(100);
    }
  }
  dmxstop();                                    /* Beende DMX-Ausgabe */
  return(0);
}
```

3.2 Programmbeispiele in Pascal

1. Beispiel
Zeigt den Gebrauch des Befehls DMXCARD

```
program bsp1;
uses crt;
var int cardadr;
{$f+}
procedure dmxcard(var cardadr:word);
  external;
procedure dmxstart; external;
procedure dmxstop;  external;
procedure dmxinit;  external;
procedure dmxtrans(dmx:dmxarray;
  anzahl:integer); external;
procedure dmxtrans_s(dmx:dmxarray;
  anzahl:integer); external;
procedure dmxchannel(channel:integer;
  wert:byte); external;
procedure dmxchannel_s(channel:integer;
  wert: byte); external;

{$f-}
{$L dmxlib.obj}

begin
  dmxcard;
  writeln("die Kartenadresse ist ", cardadr);
  readln;
end.
```

2. Beispiel
Zeigt das Übertragen von DMX-Werten auf die Kanäle 1-4
Die Kanäle 1-4 werden 10 mal langsam aufgedimmt.

```
program bsp2;
uses crt;,
var int cardadr;
{$f+}
procedure dmxcard(var cardadr:word);
  external;
procedure dmxinit; external;
procedure dmxstart;external;
procedure dmxstop; external;
procedure dmxchannel(channel:integer;
  wert:byte); external;
{$f-}
{$L dmxlib.obj}

begin
  dmxcard;
  dmxinit;
  for k:= 1 to 10 do
  begin
    for i:= 0 to 255 do
    begin
     dmxchannel(1,i);
     dmxchannel(2,i);
     dmxchannel(3,i);
     dmxchannel(4,i);
     dmxstart;
     delay(100);
    end;
  end;
  dmxstop;
end.
```

3. Beispiel
Mit dem Befehl dmxchannel(x,y) wird ein Kanalwert an die DMX-Karte übergeben. Hierbei erfolgt noch keine DMX-Ausgabe. Erst der Befehl dmxstart() sendet die Werte zu den entsprechenden DMX-Geräten.

```
program bsp3;
uses crt;
var cardadr:int;
    i,k:byte;
{$f+}
procedure dmxcard(var cardadr:word);
  external;
procedure dmxinit; external;
procedure dmxstart; external;
procedure dmxstop; external;
procedure dmxchannel(channel:integer;
  wert:byte); external;
{$f-}
{$L dmxlib.obj}
begin
  dmxcard;
  dmxinit;
  for k:= 1 to 10 do
  begin
    for i:= 0 to 255 do
    begin
     dmxchannel(1,i);
     dmxchannel(2,i);
     dmxchannel(3,i);
     dmxchannel_s(4,i);
     delay(100);
    end;
  end;
  dmxstop;
end.
```

4. Beispiel
Mit dem Befehl dmxchannel_s(x,y) wird ein Kanalwert an die DMX-Karte ubergeben und es erfolgt sofort die Ausgabe.

```
program bsp4;
uses crt;
var int cardadr;
{$f+}
procedure dmxcard(var cardadr:word);
  external;
procedure dmxinit; external;
procedure dmxstart; external;
procedure dmxstop; external;
procedure dmxchannel(channel:integer;
  wert:byte); external;
procedure dmxchannel_s(channel:integer;
  wert:byte); external;
{$f-}
{$L dmxlib.obj}
```

```
begin
  dmxcard;
  dmxinit;
  dmxchannel(1,0);
  dmxchannel(2,160);
  dmxchannel(3,80);
  dmxchannel_s(4,0);
  delay(100);
  dmxstop;
end.
```

5. Beispiel
Mit dem Befehl dmxtrans_s(x,y) werden 100 Kanäle langsam aufgedimmt.

```
program bsp5;
uses crt;
var cardadr:int;
    i,k: byte;
{$f+}
procedure dmxcard(var cardadr:word);
  external;
procedure dmxinit external;
procedure dmxstart;external;
procedure dmxstop; external;
procedure dmxchannel(channel:integer;
  wert:byte); external;
{$f-}
{$L dmxlib.obj}
begin
  dmxcard;
  dmxinit;
  for k:= 0 to 255 do
  begin
    for i:= 1 to 100 do
    begin
     dmx[i] := k;
     dmxtrans_s(dmx,100);
    end;
    delay(100);
  end;
  dmxstop;
end.
```

3.3 Pascal-Routinen in Hochsprache

Listing der Unit DMX512.PAS

Diese Unit enthält das assemblierte Betriebssystem aus Kapitel 2.5 (bitte beachten: nur für Karten mit 12 MHz Quartz! Für Karten mit 16 MHz Quartz lesen Sie die Datei SLHDMX16.BIN aus unseren Demo-Programmen und übertragen deren Inhalt Byte für Byte in die Karte).

```
unit dmx512;
interface const b_sys:array[1..168] of byte=(

  $79,$ff,$90,$04,$00,$78,$17,$c2,
  $92,$00,$00,$d8,$fc,$e5,$00,$c2,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$00,$00,$d2,$92,$00,$00,
  $00,$00,$00,$00,$00,$00,$00,$e5,
  $00,$c2,$92,$00,$13,$92,$92,$00,
  $13,$92,$92,$00,$13,$92,$92,$00,
  $13,$92,$92,$00,$13,$92,$92,$00,
  $13,$92,$92,$00,$13,$92,$92,$00,
  $13,$92,$92,$00,$00,$00,$d2,$92,
  $00,$00,$00,$00,$00,$00,$00,$00,
  $00,$e0,$a3,$c2,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$13,$92,
  $92,$00,$13,$92,$92,$00,$00,$00,
  $d2,$92,$00,$00,$00,$00,$00,$00,
  $00,$00,$00,$d9,$cc,$02,$00,$00);

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

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

Implementation
function find_card ;
{**********************************
Sucht die Adresse der DMX-Karte
gibt die Adresse als Funktionswert zurück
Wird keine Karte gefunden, ist der Wert 0
**********************************}
var
i, portreceive:byte;
   portadr:word;
begin
 portadr:=0;

 for i:= 0 to 3 do
```

```
begin
 port[$100+$20*i] := 0;
 port[$101+$20*i] := 0;
 port[$102+$20*i] := $AA;
end;

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

```
procedure dmxinit;
{**************************************
 Überträgt die 8031 Betriebssoftware
 **************************************}
var
i :integer;
begin
 for i:= 1 to 168 do
 begin                                          {Übertrage Betriebssystem}
  port[cardadr]   := lo(i);                        {Adresse lowbyte}
  port[cardadr+1] := hi(i);                        {Adresse highbyte}
  port[cardadr+2] := b_sys[i];                     {Datenwert}
 end;

 for i:= $0 to $1ff do
 begin                                          {Setze alle Kanäle =0}
  port[cardadr]   := lo(i+$400);
  port[cardadr+1] := hi(i+$400);
  port[cardadr+2] := 0;
 end;
end;

procedure dmxstart;
{**************************************
 Startet die DMX-Übertragung
 **************************************}
var
dummy : word;
begin
 dummy := port[cardadr +3];
end;

procedure dmxstop;
{**************************************
 Stoppt die DMX-Übertragung
 **************************************}
begin
 port[cardadr] := 0;
end;

procedure dmxtrans(kanaladress:word;
  data:byte);
{**************************************
 Überträgt einen DMX-Datenwert in die
 entspr. Speicherstelle des DMX-RAM
 Beispiel: Kanal 3: -> RAMadresse $402
 **************************************}
begin
 port[cardadr]   := lo($400+kanaladress-1);
 port[cardadr+1] := hi($400+kanaladress-1);
 port[cardadr+2] := data;
end;

procedure dmxtrans512;
{**************************************
 Überträgt 512 DMX-Datenwerte aus dem
 Array d_sys in das DMX-RAM von 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;
{*************************************
  Liest ein Byte aus den DMX-RAM aus
*************************************}
begin
 port[cardadr]   := lo(adresse);
 port[cardadr+1] := hi(adresse);
 dmxread         := port[cardadr+2];
end;
end.
```

```
Program DMXDEMO;
{**************************************
 Demoprogramm zeigt das Initialisieren
 der Karte, das Schreiben und Auslesen
 von Daten des DMX-RAM
 **************************************}
uses crt, dos, dmx512;
var i, data : byte;
    j : word;
begin
 cardadr := find_card;                          {Finde Basisadresse}
 dmxinit;                                       {Kopiere Betriebssystem ins RAM}
 clrscr;
 for j:= 0 to $5ff do
                                                {Lies DMX-RAM vom 0 - 5FFh aus
                                                 und zeige auf Bildschirm}
 begin
  data:=dmxread(j);
  writeln(j:4,'  ', data:4);
  delay(10);
 end;

 for i:= 0 to 255 do
                                                {Dimme die Kanäle 1- 4 langsam hoch}
 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);                                 {Kanal 1 - 4 ausschalten}
 dmxtrans(2,0);
 dmxtrans(3,0);
 dmxtrans(4,0);
 dmxstart;
 delay(10);
 dmxstop;
 readln;
end.
```

3.4 Bedienen und Erkennen der Karte mit eigenen Programmen

Wie Sie oben gesehen haben, wird das Betriebssystem der Karte als Binärdatei in die Karte geladen und dann dort gestartet. Es besteht also die Möglichkeit, die Funktionen der Karte sehr einfach zu verändern oder zu erweitern, indem das Kartenbetriebssystem -die Binärdatei also- verändert oder ausgetauscht wird. Mit unseren Demo-Applikationen liefern wir Ihnen zwei Kartenbetriebssystemversionen mit, die zwei Kartenausführungen unterstützen:

SLHDMX12.BIN für Karten 1512A oder für Karten 1512B, die mit einem 12 MHz Quarz bestückt sind,

SLHDMX16.BIN für Karten 1512B, die mit einem 16 MHz Quarz bestückt sind.

So initialisieren Sie die Karte:

- stellen Sie die Kartenadresse fest. Bedienen Sie sich dazu **einer Routine entsprechend Beispiel** *find_card* aus Abschnitt 3.3:
  (1) Schreiben Sie $00 in Adresse $0100
  (2) Schreiben Sie $00 in Adresse $0101 (0100 +1)
  (3) Schreiben Sie $AA in Adresse $0102 (0100 +2)
  (4) Lesen Sie Adresse 0102 (0100+2)
  (5) Wenn der Rückgabewert $AA ist, wurde die Karte gefunden.
  (6) Wenn nicht, wiederholen Sie (1)-(5) für die Adressen $0120, $0140, $0160

- **Übertragen Sie das Betriebssystem in die Karte**, damit die Karte funktional wird. Gehen Sie zunächst davon aus, eine Karte 1512A oder eine 12 MHz-Karte 1512B vorzufinden. Dazu
  (1) öffnen Sie die Datei SLHDMX12.BIN.
  Hinweis: Schließen Sie die Datei zunächst, bevor sie Sie öffnen. Dann sind Sie sicher, niemals einen "File open" Error zu bekommen. Nch dem Öffnen befindet sich der Zeiger am Anfang der Datei. Je nach verwendeter Sprache öffen Sie die Datei gegebenenfalls als BINARY oder als RANDOM, da alle Bytes, inklusive Nullbytes, gelesen werden müssen. VB4-Programmierer bitte aufpassen: ein einfaches OPEN reicht nicht!

  Anschließend wird die Datei in die Karte übertragen. Dabei muß für die Zugriff auf die Karte die Kartenadresse bekannt sein (zuvor ermittelt), die RAM-Adresse bezeichnet die Position innerhalb des Daten- und Programm-RAM auf der Karte. Eine Übersicht über die RAM-Belegung der Karte ist nachfolgend zu finden.
  (2) Setzen Sie RAM-Adresse = $0000
  (3) Schreiben Sie das LowByte der RAM-Adresse in die Kartenadresse +0
  (4) Schreiben Sie das HiByte der RAM-Adresse in die Kartenadresse +1
  (5) Lesen Sie das nächste Byte aus der Datei und schreiben Sie es in die Kartenadresse +2
  (6) Wiederholen Sie (3) bis (5) bis EOF

  Nunmehr ist das Betriebssystem komplett übertragen und der Typ der installierten Karte kann festgestellt werden. Dazu wird der Kartenprozessor gestartet und der von ihm generierte Rückgabewert geprüft.

  (7) Lesen Sie die Kartenadresse +3, um die Karte zu starten und warten Sie dann mindestens 50us, um das Ergebnis auszulesen
  (8) Schreiben Sie $F9 in die Kartenadresse +0
  (9) Schreiben Sie $03 in die Kartenadresse +1
  (10) Lesen Sie die Kartenadresse +2
  Rückgabewert$80: 12 MHz Karte vorhanden
  Rückgabewert$01: 16 MHz Karte vorhanden

  Wenn Ihnen nun eine 12 MHz Karte gemeldet wird, ist die Initialisierung soweit ok. Falls hingegen eine 16 MHz Karte gemeldet wird, dann müssen Sie die gesamte Initialisierung mit der Datei SLHDMX16.BIN wiederholen. Danach muß dann erneut eine 16 MHz Karte als vorhanden gemeldet werden.

Wird ein anderer Rückgabewert gemeldet, ist entweder die Karte nicht in Ordnung, oder die Übertragung des Betriebssystems war fehlerhaft. Die Rückmeldecodes $02 bis $1F sind zudem für zukünftige SLH-Produkte reserviert.

- Übertragen Sie die Default-Parameter in die Karte.

Die beiden Betriebssystem gestatten die Einstellung des Kartentimings, der Kanalzahl und weiterer Parameter. Sind keine Parameter gesetzt oder Parameter außerhalb der zulässigen Grenzen angegeben, korrigiert die Karte auf gültige Einstellungen. Generell beachten Sie bitte, daß die Timing-Einstellungen für die 12 MHz-Karte definiert sind. Sofern Sie eine 16 MHz-Karte bedienen müssen, sollte das Timing angepaßt werden (Zeitwerte *16/12)

Speicheraufteilung der PC-DMX-Karte:

$0000-$03EF	Betriebssystem, max. 1000 Bytes
$03F0-$03FF	Betriebssystemparameter dmxdef()
$0400-$05FF	Sendedaten für 512 Kanäle

Karten-RAM

Adresse	Parameter	Default
$03F0	Start-Sync-Dauer Wert * 2us	dmxdef(0) = 44
$03F1	StartByte-Wert	dmxdef(1) = 0
$03F2	Ende-Leerlaufzeit Wert * 2us + 40us	dmxdef(2) = 30
$03F3		
$03F4	Die folgenden beiden Bytes bestimmen den Kanalzähler. Achtung:	

```
LO: 001  HI: 000       1 gesendeter Kanal
LO: 002  HI: 000       2 gesendete  Kanäle
LO: 003  HI: 000       3 gesendete  Kanäle
...
LO: 000  HI: 000     256 gesendete  Kanäle
LO: 001  HI: 001     257 gesendete  Kanäle
LO: 002  HI: 001     258 gesendete  Kanäle
...
LO: 255  HI: 001     511 gesendete  Kanäle
LO: 000  HI: 001     512 gesendete  Kanäle
```

$03F3	LowByte Kanalzähler	dmxdef(3) = 50
$03F4	HiByte Kanalzähler	dmxdef(4) = 0
$03F5	Sendewiederholung 0: einmal senden >0: endlos senden	dmxdef(5) = 0
$03F6	Interdigit Zeit Pause zwischen Bytes 0...44 us	dmxdef(6) = 0

Übertragen Sie nunmehr die Defaults in das Karten-RAM:

(1) Setze n=0
(2) Schreibe $F0 + n in Kartenadresse +0
(3) Schreibe $03 in Kartenadresse +1
(4) Schreibe dmxdef(n) in Kartenadresse +2
(5) n= n+1
(6) Wiederhole (2) bis (5) solange n<7

Damit ist die Karte betriebsbereit. Als zusätzliche Maßnahme kann sich empfehlen, beim Start das Daten-RAM der Karte abzulöschen. Schreiben Sie dazu -analog zur obigen Routine- jeweils eine Null in die 512 Adressen des Karten-Daten-RAM.

So bedienen Sie die Karte

Nach der Initialisierung, dem Setzen der Betriebssystem-Parameter (Defaultwerte) und ggfs. dem Löschen des Daten-RAM steht die Karte für Ihre Übertragung bereit. Bitte bedenken Sie dabei, daß jeder Schreibzugriff auf die Karte eine augenblicklich laufende Übertragung automatisch beendet. Nach den Kartenzugriffen muß daher die Übertragung jeweils wieder gestartet werden. Das erfolgt sehr einfach dadurch, indem Sie

(1) Lesen aus Kartenadresse +3: startet DMX-Übertragung

Es empfiehlt sich, das Karten-Update timergesteuert vorzunehmen. Richten Sie sich dazu einen Timer ein, der alle x ms folgen Aufgaben durchführt:

(1) Schreiben Sie das LowByte(DMX-Kanal) in Kartenadresse +0
(2) Schreiben Sie das HiByte(DMX-Kanal) OR $04 in Kartenadresse +1
(3) Schreiben Sie den Datenwert(DMX-Kanal) in Kartenadresse +2
(4) Wiederholen Sie (1) bis (3) für alle DMX-Kanäle, die ein Update benötigen
(5) Lesen Sie Kartenadresse +3, um die DMX-Übertragung zu starten.

Die mögliche Updaterate und damit die Timerzeit ist von der DMX-Übertragungslänge abhängig. Je weniger DMX-Kanäle übertragen werden müssen, desto schneller kann der Timer gesetzt werden. Wir empfehlen Ihnen jedoch, keine Timer-Einstellung unterhalb 10 ms zu wählen.

Die DMX Gesamt-Übertragungsdauer bestimmt sich aus:

Startbytedauer	typ. 88 us	= 88
+ Mark-after-Break	typ. 8 us	= 8
+ Anzahl Kanäle * 44 us	typ. 512*44	= 22528
+ Anzahl Kanäle * Inter-digitzeit	typ.512*0	= 0
+ Ende-Leerlaufzeit	typ.100us	=100
	Summe us	22724

4. Bemerkungen und Ausblicke

Keine Software ist jemals fertig. Selbstverständlich werden alle Anwenderprogramme für unsere DMX-Interfaces ständig weiterentwickelt, und deshalb sind wir für jede, möglichst natürlich konstruktive, Kritik dankbar.

Bitte bedenken Sie für eigene Implementationen:
Da per Definition durch das USITT-Protokoll DMX-512 zwar 512 Kanale unterstützt werden, aber nicht alle Kanäle übertragen werden müssen, ist es möglich, Wiederholraten von weniger als 22,6 Millisekunden zu realisieren. Werden z.B. nur 65 Kanäle übertragen, so beträgt die Übermittlungszeit hierfür nur noch ca. 3 Millisekunden. Es liegt auf der Hand, daß in diesem Fall schnellere Aktionen programmierbar sind. Andererseits ermöglicht Ihnen unsere Karte auf Wunsch aber auch, weit mehr als 512 Kanäle zu übertragen.

Die erweiterten Betriebssysteme SLHDMX.BIN und Ressourcen SLHDMX1.DLL ermöglichen Ihnen diese Programmierungen. Bitte lesen Sie die Dateien READ.ME bzw. LIES.DAS in den betreffenden Verzeichnissen.

Neue Anwenderprogramme und zusätzliche Utilities finden Sie, sobald verfügbar, im internet auf unserer Domain

http://www.soundlight.de

Die Benutzung der SOUNDLIGHT internet Domain ist kostenlos. Sie finden hier verschiedene Rubriken, z.B. zu den Themen DMX-512, zur Hardware, zu Programmupdates und daneben auch einen Programmierkurs speziell für die PC-Karte 1512B. Hier wird in mehreren Lektionen das Erstellen eigener Anwendungen unter verschiedenen Programmierumgebungen gezeigt. Quellcodes und kompilierte Anwendungen zum sofortigen Ausprobieren lassen sich direkt aus dem internet laden.