

# **SPC PCI DLL für Windows NT Handbuch**

***Für künftige Verwendung aufbewahren !***

Ausgabe-/Rev.-Datum: 17.03.2003  
Dokument-/Rev.-Nr.:  
Softstand:  
Dateiname:  
Verfasser: SIS

**TRSystemtechnik GmbH  
Eglishalde 6  
D-78647 Trossingen**

Tel. 07425 / 228-0  
Fax 07425 / 228-34

## Impressum

### **TRSystemtechnik GmbH**

D-78647 Trossingen  
Eglishalde 16  
Tel.: (++49) 07425/228-0  
Fax: (++49) 07425/228-34

© Copyright 1997 TRSystemtechnik

## **Änderungsvorbehalt**

Änderungen der in diesem Dokument enthaltenen Informationen, die aus unserem stetigen Bestreben zur Verbesserung unserer Produkte resultieren, behalten wir uns jederzeit vor.

## **Druck**

Dieses Handbuch wurde mit einer Textformatierungssoftware auf einem DOS-Personal-Computer erstellt. Der Text wurde in *Arial* gedruckt.

## **Schreibweisen**

*Kursive* oder **fette** Schreibweise steht für den Titel eines Dokuments oder wird zur Hervorhebung benutzt.

*Courier*-Schrift zeigt Text an, der auf dem Bildschirm / Display sichtbar ist und Menüauswahlen von Software.

" < > " weist auf Tasten der Tastatur Ihres Computers hin (wie etwa <RETURN>).

## **Hinweis**

Meldungen, die nach dem Symbol "HINWEIS" erscheinen, markieren wichtige Merkmale des verwendeten Produkts.

## **Hinweise zu Urheberrechten (Copyright ©)**

MS-DOS ist ein eingetragenes Warenzeichen der Microsoft AG.

## **Literatur**

## Änderungs-Index

### i

#### **Hinweis**

Auf dem Deckblatt dieses Dokumentes ist der aktuelle Revisionsstand mit dem dazugehörigen Datum vermerkt. Da jedes einzelne Blatt in der Fußzeile mit einem eigenen Revisionsstand und Datum versehen ist, kann es vorkommen, daß sich unterschiedliche Revisionsstände innerhalb des Dokumentes ergeben.

Zeichnungen, die sich im Anhang befinden, sind mit einem eigenen Änderungs-Index versehen.

Dokumenterstellung:

17.03.2003

Änderung	Datum

## **Inhaltsverzeichnis**

<b>Konfiguration .....</b>	<b>5</b>
1.1 SPC .....	5
1.2 PC .....	5
<b>2 Programmierung.....</b>	<b>7</b>
2.1 Operationen .....	7
<b>3 Beispiel.....</b>	<b>19</b>

## Konfiguration

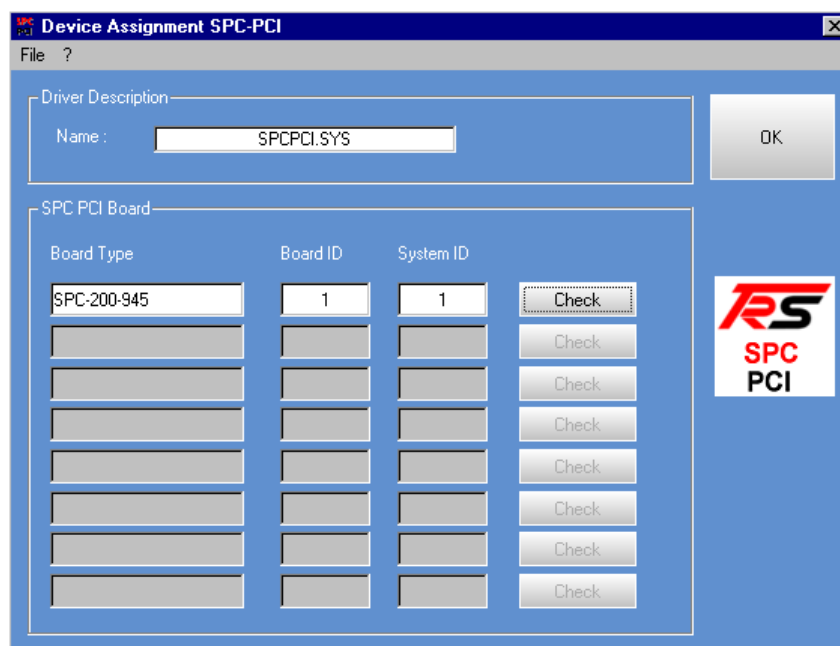
In einem Rechner können bis zu 8 SPC-PCI-Baugruppen konfiguriert werden. Zur Identifikation dieser einzelnen Baugruppen muss daher jeder Baugruppe eine Board ID vergeben werden. Über diese ID kann dann über den Treiber jede Karte angesprochen werden.

### 1.1 SPC

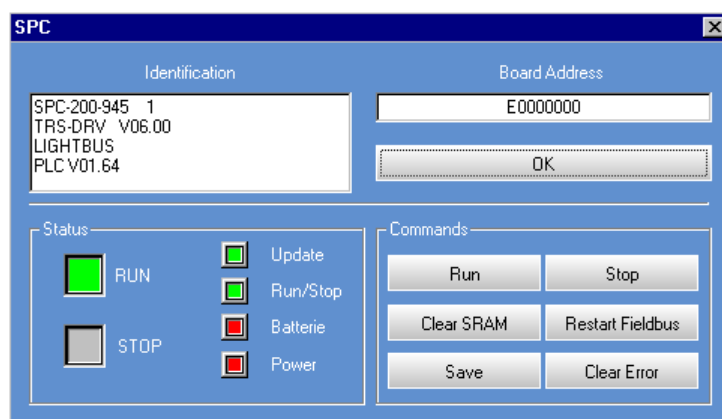
Die Daten zur statischen Konfiguration der Realtime-PLC sind beim SPC-200 in der Datei LWLDRV.INI bei den SPC-210, SPC-300, SPC-410 und SPC-510 in der Datei SPC945.INI gespeichert und werden beim Starten des Programms geladen. Diese Dateien sind ASCII-Datei und können mit jedem beliebigen Editor editiert werden. Die Parameter in eckigen Klammern und die Parameter links vom "=" Zeichen dürfen nicht verändert werden. Im Abschnitt [MISC] dieser Dateien ist im Eintrag BOARDNR die Identifikation der Baugruppe einzutragen. Zulässig sind Werte von 1 bis 8. Es ist darauf zu achten, dass jede Baugruppe in einem PC eine andere Identifikation erhält.

### 1.2 PC

Zur Überprüfung der Konfiguration wird bei der Installation des Treibers in der Systemsteuerung das Programm 'SPC PCI Configuration' installiert. Nach dem Start des Programms erscheint folgende Benutzeroberfläche:



Im Abschnitt SPC-PCI Board werden die im PC gefundenen Baugruppen aufgelistet. Das Feld BoardID zeigt dabei die unter Abschnitt 1.1 beschriebene Board-Nummer der Baugruppe an. Unter dieser ID kann die Baugruppe durch den Treiber abgesprochen werden. Das Feld SystemID zeigt eine Interne Variable an die durch das System zur Verwaltung der Boards verwendet wird. Über den Button Check lässt sich das folgende Fenster öffnen:



Folgende Informationen werden angezeigt:

### 1.2.1 Informationen

#### Identification

Zeile	Information
1	SPC-xxx-945 (Baugruppen ID "xxx" steht für die Feldbusvariante
2	TRS-DRV V06.00 (Softwareversionsnummer des TRS-Treibers)
3	INTERBUS V02.039 (Feldbuskarte und deren Firmwareversionsnummer)
4	PLC V 01.64 (Softwareversionsnummer der PLC Software)

#### Board Address

Startadresse der Baugruppe auf dem PCI-Bus des PC.

#### Status

RUN	Feld ist grün, wenn SPS im Zustand RUN
STOP	Feld ist rot, wenn SPS im Zustand STOP
Update	Updateschalter ein/aus
Run/Stop	Run/Stop Schalter ein/aus
Batterie	Lithium Batteriespannung
Power	Netzspannung

### 1.2.2 Commands

Folgende Funktionen können ausgeführt werden

Run	Run PLC
Stop	Stop PLC
Clear SRAM	Sram löschen
Restart Fieldbus	Feldbus neu starten
Save	PLC-Programm speichern
Clear Error	Fehlermeldung rücksetzen

## 2 Programmierung

Bei der Installation des Windows NT Boardtreibers wird das Device \\.\SpcPCI angelegt. Die Kommunikation zwischen Anwenderprogramm und Baugruppe erfolgt über die vorliegende DLL "SPCDLL.DLL". Um irgendeine der im folgenden Kapitel aufgezeigten Funktionen verwenden zu können muß zunächst die Funktion SPC\_Init mit der entsprechenden BoardID für jede SPC-Baugruppe aufgerufen werden. Nach erfolgreicher Initialisierung können alle weiteren Funktionen verwendet werden. Es werden bis 8 SPC-Baugruppen unterstützt.

### 2.1 Operationen

Es folgt eine Übersicht über alle implementierten Funktionen der DLL

SPC_Init	DLL initialisieren
SPC_Exit	Kommunikation mit einer SPC beenden
SPC_Close	Kommunikation mit Devicetreiber beenden
SPC_ReadByte	Ein Byte aus dem DPR einer SPC lesen
SPC_ReadWord	Ein Wort aus dem DPR einer SPC lesen
SPC_ReadSPSWord	Ein SPS-Wort aus einer SPC lesen. Beim Lesen wird das Motorola-Format der SPS in das Intel-Format des PC konvertiert
SPC_WriteByte	Ein Byte in das DPR einer SPC schreiben
SPC_WriteWord	Ein Wort in das DPR einer SPC schreiben
SPC_WriteSPSWord	Ein SPS-Wort in das DPR einer SPC schreiben. Beim Schreiben wird das Intel-Format des PC in das Motorola-Format der SPS konvertiert
SPC_ReadZeiten	Einen Timer aus einer SPC lesen
SPC_ReadZaehler	Einen Zähler aus einer SPC lesen
SPC_ReadMerkerWort	Ein Merkerwort aus einer SPC lesen
SPC_ReadMerkerByte	Ein Merkerbyte aus einer SPC lesen
SPC_ReadPAEWort	Ein Wort aus dem Prozessabbild Eingänge einer SPC lesen
SPC_ReadPAEByte	Ein Byte aus dem Prozessabbild Eingänge einer SPC lesen
SPC_ReadPAAWort	Ein Wort aus dem Prozessabbild Ausgänge einer SPC lesen
SPC_ReadPAAByte	Ein Byte aus dem Prozessabbild Ausgänge einer SPC lesen
SPC_ReadPBEWort	Ein Wort aus dem Peripheriebereich Eingänge einer SPC lesen
SPC_ReadPBEByte	Ein Byte aus dem Peripheriebereich Eingänge einer SPC lesen
SPC_ReadPBAWort	Ein Wort aus dem Peripheriebereich Ausgänge einer SPC lesen
SPC_ReadPBAByte	Ein Byte aus dem Peripheriebereich Ausgänge einer SPC lesen
SPC_ReadDB	Einen Datenbaustein einer SPC lesen
SPC_WriteDB	Einen Datenbaustein einer SPC schreiben
SPC_ReadFastDB	Mehrere Datenbausteine einer Baugruppe lesen
SPC_WriteFastDB	Mehrere Datenbausteine einer Baugruppe schreiben
SPC_ReadKME	Koppelmerker Eingänge einer SPC lesen
SPC_WriteKMA	Koppelmerker Ausgänge einer SPC schreiben
SPC_GetSPSStatus	Status einer SPC lesen
SPC_READDB_NI	Einen Datenbaustein einer SPC lesen
SPC_WRITEDB_NI	Einen Datenbaustein einer SPC schreiben
SPC_GetDLLError	Liest den letzten in der DLL aufgetretenen Fehler
SPC_ClearDLLError	Löscht den in der DLL aufgetretenen Fehler

### 2.1.1 BOOL \_stdcall SPC\_Init(short iBoardID, short iDummy);

SPC\_Init initialisiert die DLL.

iBoardID :	Bord ID einer SPC
iDummy:	Wird nicht verwendet

Returnwerte:

Bei erfolgreicher Durchführung wird True zurückgegeben.

### 2.1.2 BOOL \_stdcall SPC\_Exit(short iBoardID);

SPC\_Exit beendet die Kommunikation mit einer SPC. Die für die Kommunikation mit dieser SPC benötigten Ressourcen werden freigegeben.

iBoardID :	BoardID einer SPC
------------	-------------------

Returnwerte:

Bei erfolgreicher Durchführung wird True zurückgegeben.

### 2.1.3 BOOL \_stdcall SPC\_Close();

SPC\_Close beendet die Kommunikation mit dem Devicetreiber. Die für die Kommunikation benötigten Ressourcen werden freigegeben.

Returnwerte:

Bei erfolgreicher Durchführung wird True zurückgegeben.

### 2.1.4 short \_stdcall SPC\_ReadByte(short iBoardID,short Offset,unsigned char \*Value)

SPC\_ReadByte liest ein Byte aus dem DPR der SPC

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR von der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.5 short \_stdcall SPC\_ReadSPSWord(short iBoardID,short Offset,short \*Value)**

Liest ein Wort aus dem DPR der SPC. Dabei wird das Motorola-Format der SPS in das Intel-Format des PC konvertiert.

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR von der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.6 short \_stdcall SPC\_ReadWord(short iBoardID,short Offset, short \*Value);**

Liest ein Wort aus dem DPR der SPC.

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR von der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.7 short \_stdcall SPC\_WriteByte(short iBoardID ,short Offset,unsigned char Value);**

Schreibt ein Byte in das DPR der SPC.

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR an die geschrieben werden soll
Value :	Wert der geschrieben werden soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.8 short \_stdcall SPC\_WriteWord(short iBoardID ,short Offset,short Value);**

Schreibt ein Byte in das DPR der SPC.

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR an die geschrieben werden soll
Value :	Wert der geschrieben werden soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.9 short \_stdcall SPC\_WriteSPSWord(short iBoardID ,short Offset,short Value);**

Schreibt ein Wort in das DPR der SPC. Dabei wird das Intel-Format des PC in das Motorola-Format der SPS konvertiert.

iBoardID :	BoardID einer SPC
Offset:	Offsetadresse im DPR an die geschrieben werden soll
Value :	Wert der geschrieben werden soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.10 short \_stdcall SPC\_ReadZeiten(short iBoardID ,short Nr, short \*Value);**

Liest einen Timer aus der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Timer der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.11 short \_stdcall SPC\_ReadZaehler(short iBoardID ,short Nr, short \*Value);**

Liest einen Zähler aus der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Zählers der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.12 short \_stdcall SPC\_ReadMerkerWort(short iBoardID ,short Nr, short \*Value);**

Liest ein Merkerwort aus der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Merkers der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.13 short \_stdcall SPC\_ReadMerkerByte(short iBoardID ,short Nr, unsigned char \*Value);**

Liest ein Merkerbyte aus der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Merkers der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.14 short \_stdcall SPC\_ReadPAEWort(short iBoardID ,short Nr, short \*Value);**

Liest ein Wort aus dem Prozesabbild Eingänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Eingangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.15 short \_stdcall SPC\_ReadPAEByte(short iBoardID ,short Nr, unsigned char \*Value);**

Liest ein Byte aus dem Prozesabbild Eingänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Eingangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.16 short \_stdcall SPC\_ReadPAAWort(short iBoardID ,short Nr, short \*Value);**

Liest ein Wort aus dem Prozesabbild Ausgänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Ausgangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.17 short \_stdcall SPC\_ReadPAAByte(short iBoardID ,short Nr, unsigned char \*Value);**

Liest ein Byte aus dem Prozesabbild Ausgänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Ausgangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.18 short \_stdcall SPC\_ReadPBEWort(short iBoardID ,short Nr, short \*Value);**

Liest ein Wort aus dem Peripheriebereich Eingänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Eingangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.19 short \_stdcall SPC\_ReadPBEByte(short iBoardID ,short Nr, unsigned char \*Value);**

Liest ein Byte aus dem Peripheriebereich Eingänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Eingangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.20 short \_stdcall SPC\_ReadPBAWort(short iBoardID ,short Nr, short \*Value);**

Liest ein Wort aus dem Peripheriebereich Ausgänge der SPS

iBoardID :	BoardID einer SPC
Nr:	Nummer des Ausgangs der gelesen werden soll
Value :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.21 short \_stdcall SPC\_ReadPBAByte(short iBoardID ,short Nr, unsigned char \*Value);**

Liest ein Byte aus dem Peripheriebereich Ausgänge der SPS

<i>iBoardID</i> :	BoardID einer SPC
<i>Nr</i> :	Nummer des Ausgangs der gelesen werden soll
<i>Value</i> :	Zeiger auf Variable die gelesenen Wert aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.22 short \_stdcall SPC\_ReadDB (short iBoardID ,short iDBNr,short iOffset, short iAnzahl,short \*pValue);**

Liest Worte aus einem Datenbaustein der SPS

<i>iBoardID</i> :	BoardID einer SPC
<i>iDBNr</i> :	Nummer des Datenbausteins der gelesen werden soll
<i>iOffset</i> :	Offsetangabe im Datenbaustein ab der gelesen werden soll
<i>iAnzahl</i> :	Anzahl Worte die gelesen werden sollen
<i>pValue</i> :	Zeiger auf Puffer der die gelesenen Daten aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.23 short \_stdcall SPC\_ReadDB\_NI(short iBoardID ,short iDBNr,short iOffset, short iAnzahl,short \*pValue);**

SPC\_READDB\_NI hat die selbe Funktionsweise wie SPC\_READDB. Es wird lediglich Low- und Highbyte der Datenworte nicht getauscht.

#### 2.1.24 short \_stdcall SPC\_ReadFastDB(short iBoardID,SPCDBFAST stDB)

SPC\_ReadFastDB liest bis zu 5 Datenbausteine innerhalb eines SPS-Zyklus. Alle notwendigen Daten werden in einer Struktur vom Typ SPCDBFAST übergeben. Dabei wird für maximal 5 Datenbausteine jeweils die Datenbaustein-Nummer, der Offset und die Anzahl zu lesender Worte, sowie jeweils ein Zeiger auf einen Puffer der die zu lesenden Daten aufnimmt übergeben. Sollen weniger als 5 Datenbausteine gelesen werden, wird das Feld iDBNr[x] auf -1 gesetzt.

```
typedef struct SPCDBFASTTYPE
{
    short iBoardID;
    short iDBNr[5];
    short iOffset[5];
    short iAnzahl[5];
    char *pData[5];
} SPCDBFAST;
```

iBoardID :	BoardID einer SPC
iDBNr:	Nummer der Datenbausteine die gelesen werden sollen
iOffset:	Offsetangabe im Datenbaustein ab der gelesen werden soll
iAnzahl:	Anzahl Worte die gelesen werden sollen
pValue:	Zeiger auf Puffer der die gelesenen Daten aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

#### 2.1.25 short \_stdcall SPC\_ReadKME(short iBoardID ,short iOffset,short iAnzahl,char \*pValue);

Liest Koppelmerker der SPS

iBoardID :	BoardID einer SPC
iOffset:	Offsetangabe der Koppelmerker ab der gelesen werden soll
iAnzahl:	Anzahl Merker die gelesen werden sollen
pValue:	Zeiger auf Puffer der die gelesenen Daten aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.26 short \_stdcall SPC\_ReadDX (short iBoardID ,short iDXNr,short iOffset,  
short iAnzahl,short \*pValue);**

Liest Worte aus einem DX-Baustein der SPS

iBoardID :	BoardID einer SPC
iDXNr:	Nummer des DX-Bausteins der gelesen werden soll
iOffset:	Offsetangabe im Datenbaustein ab der gelesen werden soll
iAnzahl:	Anzahl Worte die gelesen werden sollen
pValue:	Zeiger auf Puffer der die gelesenen Daten aufnehmen soll

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.27 short \_stdcall SPC\_WriteDB (short iBoardID ,short iDBNr,short iOffset,  
short iAnzahl,short \*pValue);**

Schreibt Worte in einen Datenbaustein der SPS

iBoardID :	BoardID einer SPC
iDBNr:	Nummer des Datenbausteins der beschrieben werden soll
iOffset:	Offsetangabe im Datenbaustein ab der geschrieben werden soll
iAnzahl:	Anzahl Worte die geschrieben werden sollen
pValue:	Zeiger auf Puffer mit Daten

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.28 short \_stdcall SPC\_WriteDB\_NI(short iBoardID ,short iDBNr,short iOffset,  
short iAnzahl,short \*pValue);**

SPC\_WRITEDB\_NI hat die selbe Funktionsweise wie SPC\_WRITEDB. Es wird lediglich Low- und Highbyte der Datenworte nicht getauscht.

### 2.1.29 short \_stdcall SPC\_WriteFastDB(short iBoardID,SPCDBFAST stDB)

SPC\_WriteFastDB schreibt bis zu 5 Datenbausteine innerhalb eines SPS-Zyklus. Alle notwendigen Daten werden in einer Struktur vom Typ SPCDBFAST übergeben. Dabei wird für maximal 5 Datenbausteine jeweils die Datenbaustein-Nummer, der Offset und die Anzahl zu schreibender Worte, sowie jeweils ein Zeiger auf einen Puffer der die zu schreibenden Daten enthält übergeben. Sollen weniger als 5 Datenbausteine geschrieben werden, wird das Feld iDBNr[x] auf -1 gesetzt.

```
typedef struct SPCDBFASTTYPE
{
    short iBoardID;
    short iDBNr[5];
    short iOffset[5];
    short iAnzahl[5];
    char *pData[5];
} SPCDBFAST;
```

iBoardID :	BoardID einer SPC
iDBNr:	Nummer der Datenbausteine die geschrieben werden sollen
iOffset:	Offsetangabe im Datenbaustein ab dem geschrieben werden soll
iAnzahl:	Anzahl Worte die geschrieben werden sollen
pValue:	Zeiger auf Puffer der die zu schreibenden Daten enthält

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

### 2.1.30 short \_stdcall SPC\_WriteKMA(short iBoardID ,short iOffset,short iAnzahl,char \*pValue);

Schreibt Koppelmerker in die SPS

iBoardID :	BoardID einer SPC
iOffset:	Offsetangabe der Koppelmerker ab der geschrieben werden soll
iAnzahl:	Anzahl Merker die geschrieben werden sollen
pValue:	Zeiger auf Puffer mit Daten

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.31 short \_stdcall SPC\_WriteDX (short iBoardID ,short iDXNr,short iOffset, short iAnzahl,short \*pValue);**

Schreibt Worte in einen Datenbaustein der SPS

iBoardID :	BoardID einer SPC
iDXNr:	Nummer des DX-Bausteins der beschrieben werden soll
iOffset:	Offsetangabe im Datenbaustein ab der geschrieben werden soll
iAnzahl:	Anzahl Worte die geschrieben werden sollen
pValue:	Zeiger auf Puffer mit Daten

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.32 short \_stdcall SPC\_GetSPSStatus(short iBoardID);**

Liest den Status der SPS

iBoardID :	BoardID einer SPC
------------	-------------------

Returnwerte:

SPS-Status

0 : SPS ist in STOP

1 : SPS ist in RUN

**2.1.33 short \_stdcall SPC\_MemCs16ON(short iBoardID);**

Einige Rechner mit Pentium II – Prozessoren booten nicht, wenn das MemCs16-Signal aus der Adresse erzeugt wird.

Es wurde deshalb eine Funktion eingefügt, die das Signal MemCs16 sperrt bzw freigibt. Nach Hardware-Reset ist dann MemCs16 gesperrt. Aus Kompatibilitätsgründen wurde die Dekodierung der 8 definierten Adressräume der SPC unverändert erhalten. Die o.g. Funktion ist nur dann aktiv, wenn die Schalterstellung der SPC auf **8 .. F** steht.

Wird diese spezielle Dekodierung benutzt, muß die Funktion SPC\_MEMCS16ON aufgerufen werden, bevor eine andere Funktion zum Zugriff auf die SPC genutzt werden kann.

iBoardID :	BoardID einer SPC
------------	-------------------

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.34 short \_stdcall SPC\_MemCs16OFF(short iBoardID);**

SPC\_MEMCS16OFF schaltet die Dekodierung wie unter SPC\_MEMCS16ON beschrieben aus.

iBoardID : BoardID einer SPC

Returnwerte:

Bei erfolgreicher Durchführung wird 1 zurückgegeben sonst 0.

**2.1.35 short \_stdcall SPC\_GetDLLError();**

Liest den letzten in der DLL aufgetretenen Fehler.

Returnwerte:

Letzter aufgetretener Fehler.

- 1 : Timeout
- 2 : SPS ist in STOP
- 3 : SPC-XXX.SYS nicht gefunden
- 4 : Adressfehler
- 5 : Datenbaustein-Fehler

**2.1.36 void \_stdcall SPC\_ClearDLLError();**

Löscht den in der DLL aufgetretenen Fehler.

### 3 Beispiel

Eingesetzt wird ein PC mit 3 SPC-Baugruppen.

Folgendes Programm zeigt den Zugriff auf alle 3 SPC

```
#include <spcdll.h>

void main()
{
    /* Initialisieren aller SPC-Baugruppen */
    /* SPC1 erhält BoardID 1 */
    /* SPC2 erhält BoardID 2 */
    /* SPC3 erhält BoardID 3 */

    SPC_Init(1,0x0);
    SPC_Init(2,0x0);
    SPC_Init(3,0x0);

    /* Von SPC1 sollen aus Datenbaustein 90 */
    /* ab Datenwort 3 10 Worte gelesen werden */
    SPC_ReadDB(1,90,3,10,&iData[0]);

    /* Von SPC2 soll Merkerwort 10 und Ausgangsbyte */
    /* 5 gelesen werden */
    SPC_ReadMerkerWort(2,10,&iMW10);
    SPC_ReadPAAByte(2,5,&iPAB5);

    /* In SPC3 sollen in Datenbaustein 5 */
    /* ab Datenwort 1 3 Worte geschrieben werden */
    SPC_WriteDB(3,5,1,3,&iOutData[0]);

    /* Kommunikation mit Devicetreiber beenden */
    SPC_Close();
}
```