

Lenze

Handbuch

IEC 61131-3

inside

***Global Drive
PLC Developer Studio***



Global Drive

*Funktionsbibliothek
LenzeFpiDrv.lib*

Die Funktionsbibliothek **LenzeFpiDrv.lib** kann für folgende Lenze PLC's verwendet werden:

	Typenbezeichnung	ab Hardwarestand	ab Softwarestand
9300 Servo PLC	EVS93XX- <i>xl</i>	2K	2.0
9300 Servo PLC	EVS93XX- <i>xT</i>	2K	2.0
Drive PLC	EPL10200	VC	2.0
ECSxA	ECSxAxxx	1C	7.0

Wichtiger Hinweis:

Die Software wird dem Benutzer in der vorliegenden Form zur Verfügung gestellt. Alle Risiken hinsichtlich der Qualität und der durch ihren Einsatz ermittelten Ergebnisse verbleiben beim Benutzer. Entsprechende Sicherheitsvorkehrungen gegen eventuelle Fehlbedienungen sind vom Benutzer vorzusehen.

Wir übernehmen keine Verantwortung für direkt oder indirekt entstandene Schäden, z. B. Gewinnverluste, Auftragsverluste oder geschäftliche Beeinträchtigungen jeglicher Art.

© 2001 Lenze GmbH & Co KG

Ohne besondere schriftliche Genehmigung von Lenze GmbH & Co KG darf kein Teil dieser Dokumentation vervielfältigt oder Dritten zugänglich gemacht werden.

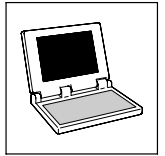
Wir haben alle Angaben in dieser Dokumentation mit größter Sorgfalt zusammengestellt und auf Übereinstimmung mit der beschriebenen Hard- und Software geprüft. Trotzdem können wir Abweichungen nicht ganz ausschließen. Wir übernehmen keine juristische Verantwortung oder Haftung für Schäden, die dadurch eventuell entstehen. Notwendige Korrekturen werden wir in die nachfolgenden Auflagen einarbeiten.

Windows, Windows NT und MS-DOS sind entweder eingetragene Warenzeichen oder Warenzeichen der Microsoft Corporation in den U.S.A und/oder anderen Ländern.

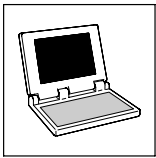
IBM und VGA sind eingetragene Warenzeichen von International Business Machines, Inc.

Alle anderen Markennamen sind Warenzeichen ihrer jeweiligen Besitzer.

Version 1.1 03/2005 - TD05

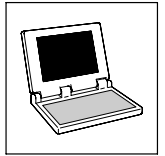


1 Vorwort und Allgemeines	1-1
1.1 Über dieses Handbuch	1-1
1.1.1 Konventionen in diesem Handbuch	1-1
1.1.2 Aufbau der Funktionsbeschreibungen	1-2
1.1.3 Piktogramme in diesem Handbuch	1-2
1.1.4 Verwendete Begriffe	1-2
1.2 Versionskennung der Funktionsbibliothek	1-3
2 Allgemeine Informationen zur RS-232C-Schnittstelle	2-1
2.1 Mechanische Eigenschaften	2-1
2.1.1 Steckverbinder-Typen	2-1
2.1.2 Anschlußbelegung beim FP-Interface	2-1
2.2 Elektrische Eigenschaften	2-2
2.2.1 Spannungspegel	2-2
2.2.2 Baudrate/Datenübertragungsrate	2-2
2.3 Synchroner/asynchroner Übertragungsmodus	2-3
2.4 Datenflußkontrolle (Handshake)	2-3
2.5 Übertragungsparameter	2-3
2.6 ASCII-Zeichensatz	2-5
2.6.1 Standard	2-5
2.6.2 Erweitert, Latin-1	2-6
2.6.3 Bedeutung der Steuerzeichen	2-7
3 Funktionen	3-1
3.1 Initialisierung (L_Rs232Open)	3-1
3.2 Deaktivierung (L_Rs232Close)	3-3
3.3 Daten senden (L_Rs232SendData)	3-4
3.4 Sendestatus (L_Rs232GetSendState)	3-6
3.5 Daten empfangen (L_Rs232ReceiveData)	3-7
3.6 Empfangsstatus (L_Rs232GetReceiveState)	3-10



Funktionsbibliothek LenzeFpiDrv.lib

Inhalt



1 Vorwort und Allgemeines

1.1 Über dieses Handbuch

Dieses Handbuch enthält Informationen zur Funktionsbibliothek **LenzeFpiDrv.lib** für das **Drive PLC Developer Studio**.

- In der Funktionsbibliothek **LenzeFpiDrv.lib** sind Funktionen zur Realisierung einer frei programmierbaren RS-232C-Schnittstelle mit dem **Kommunikationsmodul 2103 FP-Interface** enthalten.
- Mit Hilfe des **Kommunikationsmoduls 2103 FP-Interface** können Sie das Automatisierungsinterface (AIF) der **9300 Servo PLC/Drive PLC** als frei programmierbare RS-232C-Schnittstelle verwenden, um dadurch Drucker, Modems, Meßgeräte und andere Komponenten anzusteuern, die über eine serielle RS-232C-Schnittstelle verfügen.



Tip!

Informationen zur Installation sowie zu den technischen Daten des **Kommunikationsmoduls 2103 FP-Interface** entnehmen Sie bitte der dem Kommunikationsmodul beiliegenden Montageanleitung.

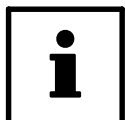
1.1.1 Konventionen in diesem Handbuch

Dieses Handbuch verwendet folgende Konventionen zur Unterscheidung von verschiedenen Arten von Informationen:

Variablenbezeichner

sind im erklärenden Text in Kursivschrift dargestellt:

- “Mit *byEndChar1*...”



Tip!

Informationen zu den Konventionen, die für die Variablenbezeichner von Lenze Systembausteinen, Funktionsblöcken sowie Funktionen verwendet werden, um eine einheitliche und durchgängige Benennung zu gewährleisten und dadurch die Lesbarkeit von SPS-Programmen zu fördern, finden Sie im Anhang der DDS-Online-Dokumentation “Einführung in die IEC1131-3 Programmierung”.

Funktionen

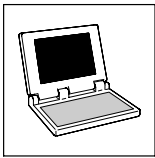
erkennen Sie am Namen. Sie beginnen immer mit einem “L_”:

- “Die Funktion **L_Rs232Open**...”

Programmlistings

sind in der Schriftart “Courier” aufgeführt, wobei Schlüsselwörter fett ausgezeichnet sind:

- “**IF** (ReturnValue < 0) **THEN**...”



Funktionsbibliothek LenzeFpiDrv.lib

Vorwort und Allgemeines

1.1.2 Aufbau der Funktionsbeschreibungen

Die Beschreibungen der einzelnen Funktionen in diesem Handbuch sind einheitlich nach folgendem Schema aufgebaut:

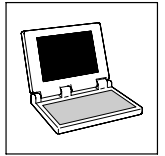
	① Überschrift mit Funktion und Funktionsbezeichner ② Deklaration der Funktion: • Datentyp des Rückgabewertes • Funktionsbezeichner • Liste der Übergabeparameter ③ Kurzbeschreibung der Funktion mit den wichtigsten Eigenschaften ④ Grafische Darstellung der Funktion mit den zugehörigen Variablen (Übergabeparameter und Rückgabewert) ⑤ Tabelle mit Informationen zu den Übergabeparametern: • Bezeichner • Datentyp • Einstellmöglichkeiten • Info ⑥ Tabelle mit Informationen zum Rückgabewert: • Datentyp des Rückgabewertes • Mögliche Rückgabewerte und ihre Bedeutung Hinweis Konnte eine Funktion nicht fehlerfrei ausgeführt werden, wird ein negativer Rückgabewert zurückgegeben, der eine Fehlernummer darstellt. • Jeder Fehlernummer ist eine entsprechende Fehlerursache in der Spalte "Bedeutung" zugeordnet. • Sind verschiedene Fehlernummern (-1, -2, ...) möglich, ist jeder Fehlernummer zusätzlich eine bestimmte Priorität (1, 2, ...) zugeordnet. – Je kleiner die Zahl, desto höher die Priorität, d. h. die Zahl "1" bedeutet die höchste Priorität. – Liegen beim Ausführen einer Funktion mehrere Fehlerursachen zugleich vor, wird von der Funktion immer die Fehlernummer mit der höchsten Priorität zurückgegeben. ⑦ Weitere Informationen zur Funktion (Anmerkungen, Tips, Anwendungsbeispiele, etc.)
--	--

1.1.3 Piktogramme in diesem Handbuch

	verwendete Piktogramme	Signalwörter	
Warnung vor Sachschäden		Stop!	Warnt vor möglichen Sachschäden . Mögliche Folgen bei Mißachtung: Beschädigung des Antriebsreglers/Antriebssystems oder seiner Umgebung.
Sonstige Hinweise		Tip! Hinweis!	Kennzeichnet einen Tip bzw. Hinweis.

1.1.4 Verwendete Begriffe

Begriff	Im folgenden Text verwendet für
DDS	Drive PLC Developer Studio
FB	Funktionsblock
FP-Interface	Lenze Kommunikationsmodul 2103 FP-Interface
GDC	Global Drive Control (Parametrier-Programm von Lenze)
Parameter-Codestellen	Codestellen zum Einstellen der Funktionalität eines Funktionsblocks
SB	Systembaustein



1.2 Versionskennung der Funktionsbibliothek

Die Version der Funktionsbibliothek können Sie über die globale Konstante `C_w[Funktionsbibliotheksname]Version` abfragen.

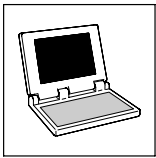
Versionskennung ab PLC Software-Stand 7.x:

Konstante	Bedeutung	Beispiel Wert
<code>C_w[Funktionsbibliotheksname]VersionER</code>	Externes Release	01
<code>C_w[Funktionsbibliotheksname]VersionEL</code>	Externes Level	05
<code>C_w[Funktionsbibliotheksname]VersionIR</code>	Internes Release	00
<code>C_w[Funktionsbibliotheksname]VersionBN</code>	Build-Nummer	00

Version: 01 05 00 00

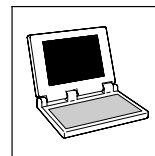
Der Wert dieser Konstanten ist hexadezimal-kodiert.

- Im Beispiel bedeutet der Wert "01050000" Version "1.05".



Funktionsbibliothek LenzeFpiDrv.lib

Vorwort und Allgemeines



2 Allgemeine Informationen zur RS-232C-Schnittstelle

RS-232C steht für *Recommended Standard-232C*, einer amerikanischen Norm der EIA (*Electronic Industries Association*), die die serielle Verbindung zwischen einem Datenendgerät (DTE = *Data Terminal Equipment*) und einer Datenübertragungseinrichtung (DCE = *Data Communications Equipment*) mit ihren elektrischen und mechanischen Eigenschaften beschreibt.

- Statt RS-232C werden für den Schnittstellenstandard oftmals auch die Bezeichnungen V.24/V.28 verwendet, hierbei handelt es sich um entsprechende internationale Empfehlungen des CCITT (heute bekannt als ITU - *International Telecommunication Union*), wobei in V.24 die mechanischen Eigenschaften und in V.28 die elektrischen Eigenschaften spezifiziert sind.

Obwohl die Norm RS-232C lediglich die Verbindung zwischen einem Datenendgerät und einer Datenübertragungseinrichtung definiert, hat sich die RS-232C-Schnittstelle als genereller Standard für serielle Datenübertragungen über kurze Distanzen etabliert.

In den folgenden Unterkapiteln erhalten Sie Informationen zu den mechanischen und elektrischen Eigenschaften der RS-232C-Schnittstelle sowie zur seriellen Datenübertragung.

2.1 Mechanische Eigenschaften

2.1.1 Steckverbinder-Typen

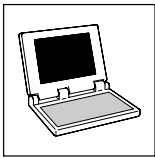
In der Regel werden als Schnittstellenverbindung Subminiatur-D-Steckverbinder in 9- oder 25-poliger Ausführung verwendet, wobei meist wie folgt unterschieden wird:

Master-fähiges Gerät (z. B. Lenze PLC mit FP-Interface)	Ausschließlich Slave-fähiges Gerät
9-pol. SubD-Stecker (male) oder 25-pol. SubD-Buchse (female)	9-pol. SubD-Buchse (female)

Für das FP-Interface ist ein Verbindungskabel mit einem 9-pol. SubD-Stecker (male) erforderlich.

2.1.2 Anschlußbelegung beim FP-Interface

9polige SubD-Buchse für RS-232C-Schnittstelle			
Pin	Bezeichnung	Ein-/Ausgang	Erläuterung
1	-	-	unbenutzt
2	RxD	Eingang	Empfangsdaten
3	TxD	Ausgang	Sendedaten
4	DTR	Ausgang	keine Funktion (logisch "0")
5	GND	-	Bezugspotential
6	DSR	Eingang	keine Funktion
7	-	-	unbenutzt
8	-	-	unbenutzt
9	+ 5V	-	max. Ausgangsstrom 50 mA



Funktionsbibliothek LenzeFpiDrv.lib

Allgemeine Informationen zur RS-232C-Schnittstelle

2.2 Elektrische Eigenschaften

2.2.1 Spannungspegel

Die logischen Zustände der einzelnen Bits werden als Spannungszustände über die Sende- bzw. Empfangsleitung übertragen:

logischer Zustand	Spannungspegel	Mindest-Spannungspegel Sender*
1	-3 ... -15 V	-5 V
0	+3 ... +15 V	+5 V

* bei zulässiger ohmscher Last > 3 kΩ

Die erzielbare Entfernung zwischen zwei RS-232C-Geräten ist wie bei allen seriellen Übertragungsverfahren vom verwendeten Kabel und der Baudrate abhängig. Als Richtmaß sollte bei einer Übertragungsrate von 9600 Baud eine Distanz von 15 bis 30 Metern nicht überschritten werden.

2.2.2 Baudrate/Datenübertragungsrate

Die Baudrate gibt an, wieviele Bit pro Sekunde übertragen werden. Aus dieser Angabe und den weiteren Übertragungsparametern (Anzahl Stoppbits/Datenbits sowie Paritätsbit) kann dann die Datenübertragungsrate ermittelt werden, die meistens in Zeichen pro Sekunde angegeben wird:

$$\text{Übertragungsrate} = \frac{\text{Baudrate}}{\text{Gesamtanzahl übertragener Bit pro Zeichen}}$$

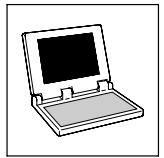
Beispiel

$$\text{Übertragungsrate} = \frac{9600}{1 (\text{Startbit}) + 8 (\text{Datenbits}) + 1 (\text{Paritätsbit}) + 1 (\text{Stoppbit})} = 872 \text{ Zeichen/s}$$

Die folgende Tabelle listet die mit dem FP-Interface möglichen Datenübertragungsraten für eine Übertragung mit 8 Datenbits, 1 Stoppbit sowie gerader Parität ("Even") auf:

Baudrate* (Bit/s)	Übertragungsdauer pro Zeichen (ms)	Datenübertragungsrate (Zeichen/s)
600	18,33	54
1200	9,17	109
2400	4,58	218
4800	2,29	436
9600	1,15	872
19200	0,573	1745
38400	0,286	3490

* Mit dem FP-Interface kann systembedingt alle 250 µs ein Zeichen gesendet werden, so daß Baudraten größer 38400 Baud mit dem FP-Interface nicht sinnvoll sind.



2.3 Synchroner/asynchroner Übertragungsmodus

Bei der seriellen Datenübertragung wird zwischen synchronen und asynchronen Übertragungsmodi unterschieden:

- **Synchroner Übertragungsmodus:**
Die Datenübertragung zwischen Sender und Empfänger wird mit Hilfe eines Takts synchronisiert. Dadurch wird gewährleistet, daß der Sender die Daten nicht schneller senden kann, als der Empfänger die Daten empfangen und verarbeiten kann.
- **Asynchroner Übertragungsmodus:**
Die Datenübertragung zwischen Sender und Empfänger erfolgt ohne Taktsynchronisation, stattdessen wird ein sogenanntes Handshake-Verfahren eingesetzt, um Datenverluste bei der Übertragung zu vermeiden.

Das FP-Interface unterstützt nur den asynchronen Übertragungsmodus!

2.4 Datenflußkontrolle (Handshake)

Bei einer Datenübertragung im asynchronen Übertragungsmodus muß der Empfänger zur Vermeidung von Datenverlusten dem Sender mitteilen können, wenn er keine weiteren Daten mehr verarbeiten kann. Diese Datenflußkontrolle bezeichnet man als Handshake, wobei zwischen folgenden zwei Verfahren unterschieden wird:

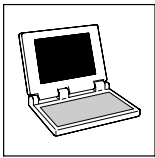
- **Hardware-Handshake:**
Der Empfänger steuert über Hardware-Leitungen die Eingänge CTS und/oder DSR des Senders mit seinem Ausgang DTR und/oder RTS.
 - Vorteil: Vollständig transparente Datenübertragung.
 - Nachteil: Es sind zusätzliche Leitungen zur Übertragung der Handshake-Signale erforderlich.
- **Software-Handshake:**
Der Empfänger sendet zur Steuerung des Datenflusses spezielle Zeichen an den Sender, z. B. XON/XOFF. (☐ 2-5, ASCII-Zeichensatz)
 - Vorteil: Zur Übertragung sind ausschließlich die Daten-Leitungen erforderlich.
 - Nachteil: Die zur Steuerung des Datenflusses verwendeten Zeichen dürfen nicht in den Nutzdaten enthalten sein.

Das FP-Interface unterstützt nur den Software-Handshake!

2.5 Übertragungsparameter

Bei einer seriellen, asynchronen Datenübertragung wird das zu übertragene Zeichen als Folge von Bits über die Datenleitung übertragen:

- Den Start der Bitfolge für ein Zeichen kennzeichnet das sogenannte Startbit, das immer als logische "0" gesendet wird.
- Anschließend wird das eigentliche Zeichen als Folge von wahlweise 7 oder 8 Bits, beginnend mit dem niederwertigsten Bit (LSB), übertragen.
- Der Bitfolge für das Zeichen kann ein Paritätsbit folgen, das zur Erkennung von Übertragungsfehlern verwendet werden kann.
- Das Ende der Bitfolge wird durch wahlweise 1 oder 2 Stoppbits gekennzeichnet, die immer als logische "1" gesendet werden.



Funktionsbibliothek LenzeFpiDrv.lib

Allgemeine Informationen zur RS-232C-Schnittstelle

Beispiel: Bitfolge/Spannungspegel für ein zu übertragenes Zeichen

Zeichen:	"A" (ASCII) 01000001 (bin) 65 (dez) 41 (hex)	
Parität*:	gerade ("Even")	
Stoppbits:	2	
Datenbits:	7	

| *Bei der Verwendung von 7 Datenbits wird immer ein Paritätsbit erzeugt, so daß in diesem Fall die Einstellung "keine Parität" nicht zulässig ist! | | |

Zeichen:	"A" (ASCII) 01000001 (bin) 65 (dez) 41 (hex)	
Parität:	gerade ("Even")	
Stoppbits:	2	
Datenbits:	8	

Damit eine erfolgreiche Kommunikation zwischen Sender und Empfänger zustande kommen kann, müssen für beide Teilnehmer die Übertragungsparameter (Baudrate, Datenbits, Paritätsbit, Stoppbits) identisch eingestellt werden.

- Die Übertragungsparameter des FP-Interface konfigurieren Sie mit Hilfe der Funktion **L_Rs232Open**. (3-1)

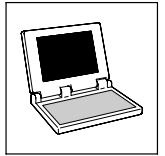
Paritätsbit

Das Paritätsbit dient als Kontroll-Bit bei der Datenübertragung, ähnlich einer Checksumme. Hierbei wird zwischen gerader und ungerader Parität ("Even"/"Odd") unterschieden, ebenso kann das Paritätsbit deaktiviert werden (Parität "None"):

- Gerade Parität ("Even"): Der Sender überträgt als Paritätsbit eine "0", wenn eine gerade Anzahl von Datenbits mit Wertigkeit "1" übertragen wurde.
- Ungerade Parität ("Odd"): Der Sender überträgt als Paritätsbit eine "0", wenn eine ungerade Anzahl von Datenbits mit Wertigkeit "1" übertragen wurde.
- Keine Parität ("None"): Es wird kein Paritätsbit übertragen.

Das Paritätsbit kann zur "Paritätsprüfung" verwendet werden:

Bei der Datenübertragung wird vom Sender die Parität der Datenbits bestimmt und über das Paritätsbit dem Empfänger mitgeteilt. Der Empfänger berechnet seinerseits die Parität der empfangenen Datenbits und vergleicht sie mit dem empfangenen Paritätsbit. Auf diese Weise können Übertragungsfehler in einem Bit (1-Bit-Fehler) festgestellt werden. Nicht feststellen lassen sich dagegen z. B. zwei falsch übertragene Datenbits (2-Bit-Fehler), da sich in diesem Fall die Parität nicht verändert.

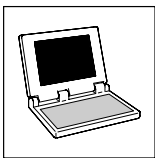


2.6 ASCII-Zeichensatz

2.6.1 Standard

Die folgende Tabelle enthält die Belegung des Standard-ASCII-Zeichensatzes (Zeichen 0 - 127):

HEX	DEC	CHAR	CTRL	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR
00	0	NUL	^@	20	32	SP	40	64	@	60	96	
01	1	SOH	^A	21	33	!	41	65	A	61	97	a
02	2	STX	^B	22	34	"	42	66	B	62	98	b
03	3	ETX	^C	23	35	#	43	67	C	63	99	c
04	4	EOT	^D	24	36	\$	44	68	D	64	100	d
05	5	ENQ	^E	25	37	%	45	69	E	65	101	e
06	6	ACK	^F	26	38	&	46	70	F	66	102	f
07	7	BEL	^G	27	39	'	47	71	G	67	103	g
08	8	BS	^H	28	40	(	48	72	H	68	104	h
09	9	HT	^I	29	41	)	49	73	I	69	105	i
0A	10	LF	^J	2A	42	*	4A	74	J	6A	106	j
0B	11	VT	^K	2B	43	+	4B	75	K	6B	107	k
0C	12	FF	^L	2C	44	,	4C	76	L	6C	108	l
0D	13	CR	^M	2D	45	-	4D	77	M	6D	109	m
0E	14	SO	^N	2E	46	.	4E	78	N	6E	110	n
0F	15	SI	^O	2F	47	/	4F	79	O	6F	111	o
10	16	DLE	^P	30	48	0	50	80	P	70	112	p
11	17	DC1	^Q	31	49	1	51	81	Q	71	113	q
12	18	DC2	^R	32	50	2	52	82	R	72	114	r
13	19	DC3	^S	33	51	3	53	83	S	73	115	s
14	20	DC4	^T	34	52	4	54	84	T	74	116	t
15	21	NAK	^U	35	53	5	55	85	U	75	117	u
16	22	SYN	^V	36	54	6	56	86	V	76	118	v
17	23	ETB	^W	37	55	7	57	87	W	77	119	w
18	24	CAN	^X	38	56	8	58	88	X	78	120	x
19	25	EM	^Y	39	57	9	59	89	Y	79	121	y
1A	26	SUB	^Z	3A	58	:	5A	90	Z	7A	122	z
1B	27	ESC	^[	3B	59	;	5B	91	[	7B	123	{
1C	28	FS	^\	3C	60	<	5C	92	\	7C	124	
1D	29	GS	^]	3D	61	=	5D	93	]	7D	125	}
1E	30	RS	^^	3E	62	>	5E	94	^	7E	126	~
1F	31	US	^_	3F	63	?	5F	95	_	7F	127	DEL



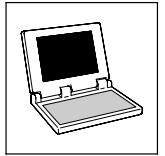
Funktionsbibliothek LenzeFpiDrv.lib

Allgemeine Informationen zur RS-232C-Schnittstelle

2.6.2 Erweitert, Latin-1

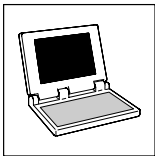
Die folgende Tabelle enthält die Belegung des erweiterten ASCII-Zeichensatzes (Latin-1, Zeichen 128 - 255):

HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR	HEX	DEC	CHAR
80	128	PAD	A0	160	NBS	C0	192	À	E0	224	à
81	129	HOP	A1	161	ı	C1	193	Á	E1	225	á
82	130	BPH	A2	162	¢	C2	194	Â	E2	226	â
83	131	NBH	A3	163	£	C3	195	Ã	E3	227	ã
84	132	IND	A4	164	¤	C4	196	Ä	E4	228	ä
85	133	NEL	A5	165	¥	C5	197	Å	E5	229	å
86	134	SSA	A6	166	¦	C6	198	Æ	E6	230	æ
87	135	ESA	A7	167	§	C7	199	Ç	E7	231	ç
88	136	HTS	A8	168	¨	C8	200	È	E8	232	è
89	137	HTJ	A9	169	©	C9	201	É	E9	233	é
8A	138	VTS	AA	170	ª	CA	202	Ê	EA	234	ê
8B	139	PLD	AB	171	«	CB	203	Ë	EB	235	ë
8C	140	PLU	AC	172	¬	CC	204	Ì	EC	236	ì
8D	141	RI	AD	173	-	CD	205	Í	ED	237	í
8E	142	SS2	AE	174	®	CE	206	Î	EE	238	î
8F	143	SS3	AF	175	¯	CF	207	Ï	EF	239	ï
90	144	DCS	B0	176	°	D0	208	Ð	F0	240	ð
91	145	PU1	B1	177	±	D1	209	Ñ	F1	241	ñ
92	146	PU2	B2	178	²	D2	210	Ò	F2	242	ò
93	147	STS	B3	179	³	D3	211	Ó	F3	243	ó
94	148	CCH	B4	180	´	D4	212	Ô	F4	244	ô
95	149	MW	B5	181	µ	D5	213	Ù	F5	245	õ
96	150	SPA	B6	182	¶	D6	214	Ö	F6	246	ö
97	151	EPA	B7	183	·	D7	215	×	F7	247	÷
98	152	SOS	B8	184	¸	D8	216	Ø	F8	248	ø
99	153	SGCI	B9	185	¹	D9	217	Ù	F9	249	ù
9A	154	SCI	BA	186	º	DA	218	Ú	FA	250	ú
9B	155	CSI	BB	187	»	DB	219	Û	FB	251	û
9C	156	ST	BC	188	¼	DC	220	Ü	FC	252	ü
9D	157	OSC	BD	189	½	DD	221	Ý	FD	253	ý
9E	158	PM	BE	190	¾	DE	222	Þ	FE	254	þ
9F	159	APC	BF	191	¿	DF	223	ß	FF	255	ÿ



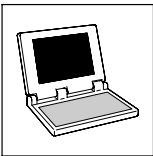
2.6.3 Bedeutung der Steuerzeichen

HEX	Name	Bedeutung	HEX	Name	Bedeutung
00	<nul>	Null	11	<dc1>	Device control 1 / XON
01	<soh>	Start of header	12	<dc2>	Device control 2
02	<stx>	Start of text	13	<dc3>	Device control 3 / XOFF
03	<etx>	End of text	14	<dc4>	Device control 4
04	<eot>	End of transmission	15	<nak>	Negative acknowledgment
05	<enq>	Enquiry	16	<syn>	Synchronous idle
06	<ack>	Acknowledgment	17	<etb>	End of transmission block
07	<bel>	Bell	18	<can>	Cancel
08	<bs>	Backspace	19		End of medium
09	<ht>	Horizontal tab	1A	<sub>	Substitute character
0A	<lf>	Line feed	1B	<esc>	Escape
0B	<vt>	Vertical tab	1C	<fs>	File separator
0C	<ff>	Form feed	1D	<gs>	Group separator
0D	<cr>	Carriage return	1E	<rs>	Record separator
0E	<so>	Shift out	1F	<us>	Unit separator
0F	<si>	Shift in			
10	<dle>	Data link escape	7F		Delete



Funktionsbibliothek LenzeFpiDrv.lib

Allgemeine Informationen zur RS-232C-Schnittstelle



3 Funktionen

3.1 Initialisierung (L_Rs232Open)

```
DWORD L_Rs232Open (wDrvNr, dwBaudrate, byNumberOfDataBits,  
byNumberOfStopBits, byParity)
```

Bevor mit dem FP-Interface gearbeitet werden kann, muß mit dieser Funktion eine Initialisierung der Schnittstelle mit den gewünschten Übertragungsparametern erfolgen.



Hinweise!

- Nach erfolgreicher Initialisierung arbeitet das Automatisierungsinterface (AIF) **ausschließlich** mit dem FP-Interface.
- Andere AIF-Aufsteckmodule (Keypad, INTERBUS-S, PROFIBUS-DP usw.) funktionieren erst wieder nach Aufruf der Funktion **L_Rs232Close**.
- Durch erneuten Aufruf dieser Funktion kann die Schnittstelle im initialisierten Zustand auf andere Übertragungsparameter umgeschaltet werden, dies hat aber zugleich zur Folge, daß bereits anstehende Sende- und Empfangsaufträge abgebrochen werden!

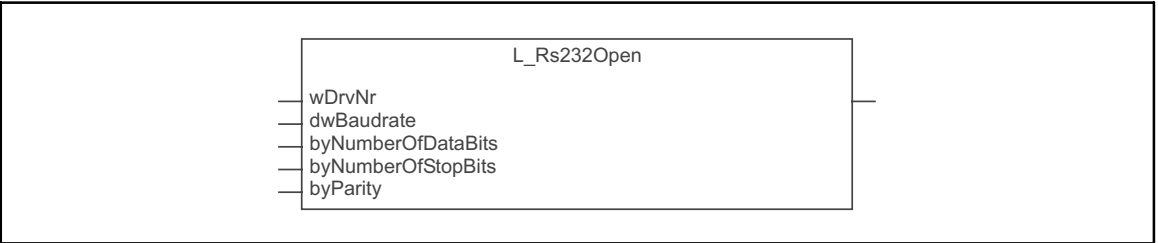
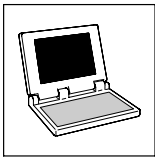


Abb. 3-1 Initialisierung (L_Rs232Open)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface
dwBaudrate	Double Word	600, 1200, 2400, 4800 9600, 19200, 38400	Baudrate
byNumberOfDataBits	Byte	7 oder 8	Anzahl der Datenbits*
byNumberOfStopBits	Byte	1 oder 2	Anzahl der Stoppbits
byParity	Byte	0: Keine 1: Even (gerade) 2: Odd (ungerade)	Parität*
* Bei der Verwendung von 7 Datenbits wird immer ein Paritätsbit erzeugt, so daß in diesem Fall die Einstellung "keine Parität" (byParity = 0) nicht zulässig ist!			



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Initialisierung (L_Rs232Open)

Rückgabewert

Datentyp	Bit	Wert	Bedeutung	Priorität
Double Word	0	0	Treiber ist initialisiert.	-
		1	Fehler beim Initialisieren.	
	1	0	Einstellung Treiber Ok.	
		1	Angewählter Treiber (<i>wDrvN</i>) ist auf dem Zielsystem nicht verfügbar.	
	2	0	Einstellung Baudrate Ok.	
		1	Angewählte Baudrate (<i>dwBaudrate</i>) steht nicht zur Verfügung.	
	3	0	Einstellung Datenbits Ok.	
		1	Falsche Anzahl Datenbits (<i>byNumberOfDataBits</i>).	
	4	0	Einstellung Stoppbits Ok.	
		1	Falsche Anzahl Stoppbits (<i>byNumberOfStopBits</i>).	
	5	0	Einstellung Parität Ok.	
		1	Falsche Einstellung der Parität (<i>byParity</i>).	
	6-15		Reserviert für zukünftige Erweiterungen (Bits sind auf 0 gesetzt).	
	16-31		Version des FPI-Treibers Format: Hauptversion/Unterversion (z. B. 0103hex = Version 1.03)	

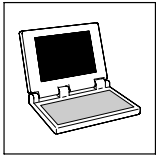
Beispiel

Aufruf der Funktion in ST mit Dekodierung des Rückgabewertes:

```
(* open FPI - returns g_dwOpenStatus *)

g_dwOpenStatus := L_Rs232Open(30, 4800, 8, 1, 1);

g_bInitOK      := NOT DWORD_TO_BOOL(g_dwOpenStatus AND 16#0000_00001);
g_bDriverFail  := DWORD_TO_BOOL(SHR(g_dwOpenStatus,1) AND 16#0000_0001);
g_bBaudrateFail:= DWORD_TO_BOOL(SHR(g_dwOpenStatus,2) AND 16#0000_0001);
g_bDatabitsFail:= DWORD_TO_BOOL(SHR(g_dwOpenStatus,3) AND 16#0000_0001);
g_bStopbitsFail:= DWORD_TO_BOOL(SHR(g_dwOpenStatus,4) AND 16#0000_0001);
g_bParityFail  := DWORD_TO_BOOL(SHR(g_dwOpenStatus,5) AND 16#0000_0001);
g_wVersion     := DWORD_TO_WORD(SHR(g_dwOpenStatus,16) AND 16#FFFF);
```



3.2 Deaktivierung (L_Rs232Close)

BOOL L_Rs232Close (wDrvNr)

Mit dieser Funktion wird das FP-Interface deaktiviert, so daß an der AIF-Schnittstelle wieder andere AIF-Aufsteckmodule (Keypad, INTERBUS-S, PROFIBUS-DP usw.) verwendet werden können.

- Noch anstehende Sende- und Empfangsaufträge werden nach Aufruf dieser Funktion abgebrochen.

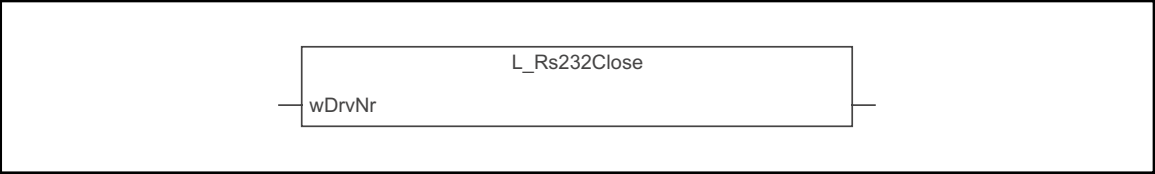


Abb. 3-2 Deaktivierung (L_Rs232Close)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface

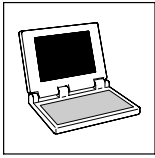
Rückgabewert

Datentyp	Wert	Bedeutung	Priorität
Bool	FALSE	Fehler beim Deaktivieren des Treibers: Falsche Treibernummer (<i>wDrvNr</i>) oder Treiber war nicht initialisiert.	-
	TRUE	Treiber wurde deaktiviert.	

Beispiel

Aufruf der Funktion in ST:

```
(* close FPI - returns g_bFPICloseState *)  
  
g_bFPICloseState := L_Rs232Close(30)
```



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Daten senden (L_Rs232SendData)

3.3 Daten senden (L_Rs232SendData)

INT **L_Rs232SendData** (wDrvNr, pabySendDataMemory, wNumberOfBytes)

Mit dieser Funktion können Daten über das FP-Interface gesendet werden.

- Das Senden von Daten übernimmt das Betriebssystem des Antriebsreglers parallel zum Ablauf des SPS-Programms, wodurch max. alle 250 µs ein Byte gesendet werden kann.
- Es können mit einem Funktionsaufruf max. 32767 Zeichen gesendet werden.
- Ein neuer Aufruf kann erst erfolgen, nachdem der vorherige Aufruf abgearbeitet wurde, d. h. nachdem alle Daten gesendet wurden.
- Ein aktiver Sendevorgang kann abgebrochen werden, indem die Funktion **L_Rs232Open** erneut aufgerufen wird. Daraufhin kann ein neuer Sendeauftrag mit den evtl. veränderten Sendeparametern erfolgen.

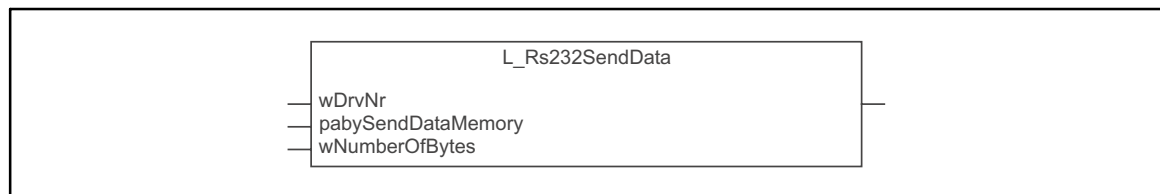


Abb. 3-3

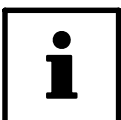
Daten senden (L_Rs232SendData)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface
pabySendDataMemory	Pointer to Array of Byte	Zeiger auf die Adresse im Speicher, ab der sich die zu sendenden Datenbytes befinden.	Die Adresse einer Variablen (z. B. eines Datenarrays) können Sie mit der Adressfunktion ADR ermitteln.
wNumberOfBytes	Word	1 ... 32767	Anzahl der zu sendenden Datenbytes

Rückgabewert

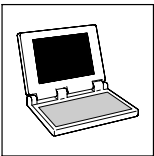
Datentyp	Wert	Bedeutung	Priorität
Integer	-4	Zeiger auf Sendespeicher zeigt nicht auf SPS-RAM.	4
	-3	Anzahl der zu sendenden Bytes außerhalb der Grenzen (1 ... 32767).	3
	-2	Sendeauftrag wurde nicht an das Betriebssystem übergeben, da der vorherige Sendeauftrag noch nicht abgeschlossen ist.	2
	-1	Falsche Treibernummer (wDrvNr) oder Treiber war nicht initialisiert.	1
	Hinweis: Beim Vorhandensein mehrerer Fehlerursachen wird immer der zur Fehlerursache mit der höchsten Priorität zugehörige Rückgabewert zurückgegeben.		
	0	Sendeauftrag wurde erfolgreich an das Betriebssystem übergeben.	



Tip!

Der Sendeprozess erfolgt parallel zum Ablauf des SPS-Programms.

Achten Sie daher darauf, daß vor dem Aufruf dieser Funktion der letzte Sendeauftrag abgeschlossen ist, da sonst ein Fehler auftritt (Rückgabewert = -2).



Beispiel 1

Aufruf der Funktion in ST für einen einmaligen Sendevorgang:

```
g_nSendStatus := L_Rs232SendData(30,
                                   ADR(g_abySendData),
                                   g_wNumberOfBytesToSend);
```

Beispiel 2

Aufruf der Funktion in ST mit Überwachung des letzten Sendevorgangs:

```
(* read state of sendprocess and send new data *)
(* if last sendprocess is finished *)

IF L_Rs232GetSendState(30) = 0 THEN

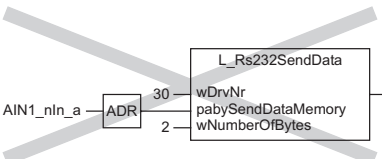
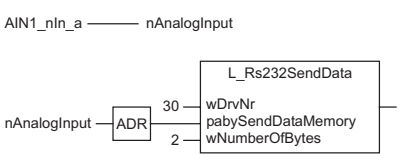
    g_nSendStatus := L_Rs232SendData(30,
                                       ADR(g_abySendData),
                                       g_wLength);

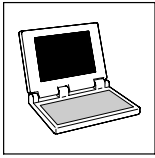
END_IF
```



Hinweis!

Die Verwendung einer Adresse, die im Speicherbereich des Betriebssystems liegt, ist bei den Funktionen **L_Rs232SendData** und **L_Rs232ReceiveData** nicht erlaubt!

Problem:	Abhilfe:
Auch die Adressen der Systemvariablen (Ein-/Ausgänge von Systembausteinen) liegen im Speicherbereich des Betriebssystems, in der folgenden Abbildung die Systemvariable <i>AIN1_nln_a</i> .	Verwenden Sie statt der Adresse der Systemvariable die Adresse einer Zwischenvariable, in die der Wert der entsprechenden Systemvariable kopiert wird:
	



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Sendestatus (L_Rs232GetSendState)

3.4 Sendestatus (L_Rs232GetSendState)

INT **L_Rs232GetSendState** (wDrvNr)

Mit dieser Funktion kann der momentane Sendestatus eines zuvor abgegebenen Sendeauftrags ermittelt werden.

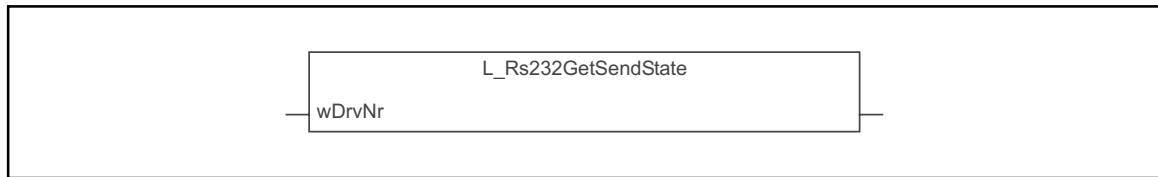


Abb. 3-4

Sendestatus (L_Rs232GetSendState)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface

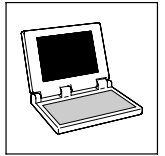
Rückgabewert

Datentyp	Wert	Bedeutung	Priorität
Integer	-1	Falsche Treibernummer (<i>wDrvNr</i>) oder Treiber war nicht initialisiert.	-
	0	Alle Datenbytes wurden gesendet, der Sendespeicher ist frei für weitere Aufträge.	
	1 ... 32767	Anzahl der noch zu sendenden Datenbytes bei aktivem Sendeauftrag.	

Beispiel

Aufruf der Funktion in ST:

```
g_dnStatusOfSend := L_Rs232GetSendState (30);
```



3.5 Daten empfangen (L_Rs232ReceiveData)

```
INT L_Rs232ReceiveData (wDrvNr, pabyReceiveDataMemory, wBufferSize,
                        wTimeOut, wNumberOfBytes,
                        byEndChar1, byEndChar2, wControl)
```

Mit dieser Funktion können Daten über das FP-Interface empfangen werden.

- Das Empfangen von Daten übernimmt das Betriebssystem des Antriebsreglers parallel zum Ablauf des SPS-Programms.
- Es können mit einem Funktionsaufruf max. 32767 Zeichen empfangen werden.
- Ein neuer Aufruf kann erst erfolgen, nachdem der vorherige Aufruf abgearbeitet wurde, d. h. nachdem alle Daten empfangen bzw. ein Empfangsende erkannt wurde.
- Ein aktiver Empfangsvorgang kann abgebrochen werden, indem die Funktion **L_Rs232Open** erneut aufgerufen wird. Daraufhin kann ein neuer Empfangsauftrag mit den evtl. veränderten Empfangsparametern erfolgen.

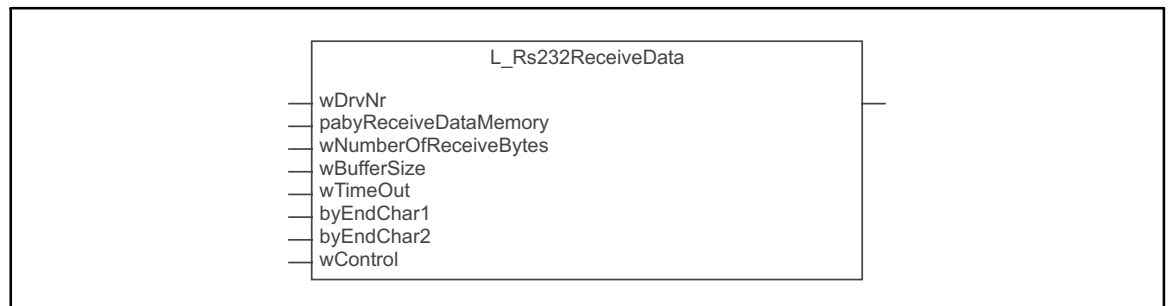
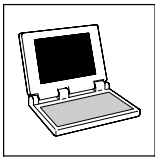


Abb. 3-5 Daten empfangen (L_Rs232ReceiveData)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface
pabyReceiveDataMemory	Pointer to Array of Byte	Zeiger auf die Adresse im Speicher, ab der die empfangenen Datenbytes gespeichert werden.	Die Adresse einer Variablen (z. B. eines Datenarrays) können Sie mit der Adressfunktion ADR ermitteln.
wBufferSize	Word	2 ... 32767	Größe des Empfangsspeichers in Byte • Der Empfangsspeicher hat eine Mindestgröße von zwei Byte.
wNumberOfReceiveBytes	Word	1 ... 32767	Anzahl der zu empfangenen Datenbytes • Dieser Parameter kann zur Empfangsendekontrolle verwendet werden (siehe <i>wControl</i>).
wTimeOut	Word	2 ... 60000	Timeout-Zeit (in ms) für den Datenempfang. • Dieser Parameter kann zur Empfangsendekontrolle verwendet werden (siehe <i>wControl</i>).
byEndChar1	Byte	0 ... 255	Zeichen, das das Ende der zu empfangenen Datenbytes definiert. • Dieser Parameter kann zur Empfangsendekontrolle verwendet werden (siehe <i>wControl</i>).
byEndChar2	Byte	0 ... 255	Zeichen, das das Ende der zu empfangenen Datenbytes definiert. • Dieser Parameter kann zur Empfangsendekontrolle verwendet werden (siehe <i>wControl</i>).



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Daten empfangen (L_Rs232ReceiveData)

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wControl	Word	Empfangsende nach: Bit 0: <i>wTimeout</i> Bit 1: <i>wNumberOfBytes</i> Bit 2: <i>byEndChar1</i> Bit 3: <i>byEndChar2</i>	Empfangsendekontrolle 0: inaktiv 1: aktiv 0: inaktiv 1: aktiv 0: inaktiv 1: aktiv 0: inaktiv 1: aktiv Bit 4-15 sind reserviert für spätere Erweiterungen

Rückgabewert

Datentyp	Wert	Bedeutung	Priorität
Integer	-6	<i>wControl</i> ist ungültig, da reservierte Bits beschrieben wurden.	6
	-5	Timeout-Zeit außerhalb der Grenzen (2 ... 60000 ms).	5
	-4	Anzahl zu empfangener Bytes > Empfangsspeicher (1 ... 32767) oder Anzahl zu empfangener Bytes = 0.	4
	-3	Empfangsspeichergröße außerhalb der Grenzen (2 ... 32767).	3
	-2	Zeiger auf Empfangsspeicher zeigt nicht auf SPS-RAM.	2
	-1	Falsche Treibernummer (<i>wDrvNr</i>) oder Treiber war nicht initialisiert.	1
	Hinweis: Beim Vorhandensein mehrerer Fehlerursachen wird immer der zur Fehlerursache mit der höchsten Priorität zugehörige Rückgabewert zurückgegeben.		
	0	Empfangsauftrag wurde erfolgreich an das Betriebssystem übergeben.	

Empfangsendekontrolle

Die Art und Weise, wie das Empfangsende erkannt und dadurch die Empfangsbereitschaft beendet wird, kann mit Hilfe von *wControl* konfiguriert werden:

Empfangsende, wenn	wControl	Priorität
der Empfangsspeicher voll ist. (Diese Endekennung ist immer aktiv.)		1
nach einer Wartezeit <i>wTimeout</i> kein Byte empfangen wurde.	0000 0000 0000 0001	2
die Anzahl von <i>wNumberOfReceiveBytes</i> Bytes empfangen wurde.	0000 0000 0000 001x	3
das Zeichen <i>byEndChar1</i> empfangen wurde.	0000 0000 0000 010x	4
das Zeichen <i>byEndChar2</i> empfangen wurde.	0000 0000 0000 100x	5
das Zeichen <i>byEndChar1</i> und <i>byEndChar2</i> in Folge empfangen wurde.	0000 0000 0000 110x	6
(x: Zustand beliebig)		

Beispiel:

$wControl = 0x0009_{\text{hex}} = 0000\ 0000\ 0000\ 1001_{\text{bin}}$

Empfangsende *wTimeout* ist aktiv

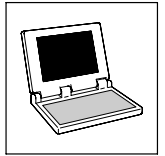
Empfangsende *byEndChar2* ist aktiv



Achtung!

Achten Sie darauf, daß die Größe des angelegten Empfangsspeichers (z. B. ein Array) nicht die in *wBufferSize* angegebene Größe unterschreitet, da sonst die Speichergrenzenprüfung nicht funktioniert.

Dadurch können Speicherbereiche u. U. unerlaubt überschrieben werden, was schwerwiegende Fehler zur Folge haben kann!



Empfangsspeicher voll

Wenn beim Empfangen der Daten erkannt wird, daß der Empfangsspeicher voll ist, so wird der Empfangsprozess sofort abgebrochen. Dadurch wird verhindert, daß unerlaubte Speicherbereiche mit Empfangsdaten überschrieben werden, was schwerwiegende Fehler zur Folge haben kann.

TimeOut

Mit dem Aufruf der Funktion wird ein TimeOut-Timer gestartet. Dieser Timer wird jedesmal zurückgesetzt, wenn ein Byte empfangen wurde. Wurden innerhalb der eingestellten Zeit *wTimeOut* keine Daten empfangen, so wird ein TimeOut-Fehler ausgelöst und die Empfangsbereitschaft abgebrochen.

NumberOfReceiveBytes

Wenn *wNumberOfReceiveBytes* Bytes empfangen wurden, wird der Empfangsprozess sofort beendet.

Endezeichen (byEndChar1/byEndChar2)

Zur Datenflußkontrolle bei der asynchronen Datenübertragung wird zwischen Sender und Empfänger oft ein Start- und ein Endezeichen definiert. Das Endezeichen kennzeichnet das Ende eines jeden gesendeten Datenpaketes und muß auf der Empfängerseite entsprechend erkannt werden. Manchmal wird als Endezeichen auch ein Zeichenpaar definiert.

Mit *byEndChar1* und *byEndChar2* können diese Endezeichen festgelegt werden. Der Empfangsprozess wird beendet, sobald die definierten Endezeichen empfangen wurden. Die Endezeichen werden ebenfalls im Empfangsspeicher abgelegt.

Beispiel

Aufruf der Funktion in ST:

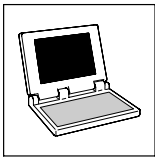
```
g_nReceiveStatus:=L_Rs232ReceiveData(30,  
                                      ADR(g_abyReceiveData),  
                                      g_wLength,  
                                      g_wReceiveBufferSize,  
                                      g_wTimeOut,  
                                      g_byEndChar1,  
                                      g_byEndChar2,  
                                      g_wControl);
```



Hinweis!

Die Verwendung einer Adresse, die im Speicherbereich des Betriebssystems liegt, ist bei den Funktionen **L_Rs232SendData** und **L_Rs232ReceiveData** nicht erlaubt!

- Siehe auch Hinweis zur Funktion **L_Rs232SendData**. (□ 3-4)



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Empfangsstatus (L_Rs232GetReceiveState)

3.6 Empfangsstatus (L_Rs232GetReceiveState)

INT **L_Rs232GetReceiveState** (wDrvNr, bStateControl)

Mit dieser Funktion kann der momentane Status eines zuvor abgegebenen Empfangsauftrags ermittelt und in Abhängigkeit von *bStateControl* auf unterschiedliche Weise ausgegeben werden:

Ist *bStateControl*

- **FALSE**, so wird im Falle eines aufgetretenen Fehlers beim Empfang ein Fehlercode zurückgegeben, ansonsten die Anzahl der empfangenen Bytes.
- **TRUE**, so wird - unabhängig von evtl. aufgetretenen Fehlern beim Empfang - die Anzahl der bereits empfangenen Bytes zurückgegeben.

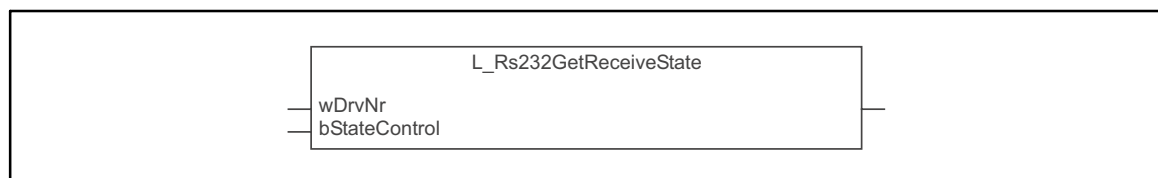


Abb. 3-6 Empfangsstatus (L_Rs232GetReceiveState)

Übergabeparameter

Bezeichner	Datentyp	Einstellmöglichkeiten	Info
wDrvNr	Word	30	AIF-Interface
bStateControl	Bool	FALSE	Rückgabewert ist Empfangsstatus
		TRUE	Rückgabewert ist Anzahl der empfangenen Bytes

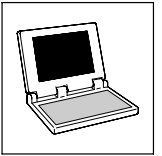
Rückgabewert

bStateControl = **FALSE**

Datentyp	Wert	Bedeutung	Priorität
Integer	-7	Framefehler der seriellen Schnittstelle (falsche Einstellung der Anzahl Stopbits).	2
	-6	Overrun-Fehler der seriellen Schnittstelle (falsche Einstellung der Baudrate).	3
	-5	Parity-Fehler der seriellen Schnittstelle (falsche Einstellung der Parität).	4
	-4	Empfang abgebrochen, da Empfangsspeicher voll.	5
	-3	TimeOut-Zeit wurde überschritten oder innerhalb der TimeOut-Zeit wurden keine Zeichen empfangen.	6
	-2	Letzter Empfangsauftrag noch nicht abgeschlossen.	7
	-1	Falsche Treibernummer (<i>wDrvNr</i>) oder Treiber war nicht initialisiert.	1
	Hinweis: Beim Vorhandensein mehrerer Fehlerursachen wird immer der zur Fehlerursache mit der höchsten Priorität zugehörige Rückgabewert zurückgegeben. Auch bei negativen Rückgabewert befinden sich die bereits empfangenen Daten im Empfangsspeicher.		
	0 ... 32767	Anzahl der empfangenen Datenbytes, wenn Empfangsende erkannt wurde.	

bStateControl = **TRUE**

Datentyp	Wert	Bedeutung	Priorität
Double Integer	-1	Falsche Treibernummer (<i>wDrvNr</i>) oder Treiber war nicht initialisiert.	-
	0 ... 32767	Anzahl der empfangenen Datenbytes (unabhängig von Empfangsende oder Fehlern).	



Beispiel

Aufruf der Funktion in ST:

```
(* read receive state *)

ReturnValue := L_Rs232GetReceiveState(30, FALSE);

(* Receive Error? *)

IF (ReturnValue < 0) THEN

    (* Receive Error *)
    (* get error code and number *)
    (* of received bytes in case of error *)

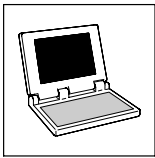
    g_nReceiveErrorCode := ReturnValue;
    g_wNumberOfReceivedBytes_Error := L_Rs232GetReceiveState(30, TRUE);

ELSE

    (* No Receive Error *)
    (* get number of received bytes *)

    g_wNumberOfReceivedBytes_NoError := ReturnValue;

END_IF
```



Funktionsbibliothek LenzeFpiDrv.lib

Funktionen

Empfangsstatus (L_Rs232GetReceiveState)